The Survivable Network Design Problems with High Node-Connectivity Constraints: Polyhedra and Algorithms

Meriem Mahjoub

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée en cotutelle à l'Université Paris-Dauphine et l'Université de Tunis El Manar

The Survivable Network Design Problem with High Node-Connectivity Constraints: Polyhedra and Algorithms

École doctorale de Dauphine – ED 543

Spécialité INFORMATIQUE

COMPOSITION DU JURY :

M. Ali Ridha MAHJOUB
Université Paris-Dauphine
Directeur de thèse

M. Faouzi BEN CHARRADA
Université de Tunis El Manar
Directeur de thèse

M. Ibrahima DIARRASSOUBA
Université du Havre
Co-encadrant

M. Imed KACEM
Université de Lorraine
Rapporteur

M. Pierre FOUILHOUX
Université Pierre et Marie Curie
Examinateur

M. Vangelis PASCHOS
Université Paris-Dauphine
Président du jury

Soutenue par
Meriem MAHJOUB
le **13.12.2017**

Dirigée par **A. Ridha MAHJOUB**
et **Faouzi BEN CHARRADA**

Remerciements

Je tiens à remercier Mr. A. Ridha Mahjoub, Professeur à l'Université Paris Dauphine, de m'avoir permis d'effectuer cette thèse sous sa direction. Je voudrais lui exprimer ma profonde gratitude pour la confiance qu'il m'a accordée en me permettant d'effectuer mon stage de P.F.E. puis une thèse sous sa direction. Je le remercie pour sa disponibilité, ses conseils, et son soutien. J'ai découvert grâce à lui l'univers de la recherche et de l'enseignement. J'ai apprécié les longues heures passées à travailler ensemble, à travers lesquelles j'ai beaucoup appris. Je le remercie de m'avoir laissée profiter de sa grande expérience et de sa rigueur scientifique. Je lui en serai toujours reconnaissante.

Je remercie Mr. Faouzi Ben Charrada, Professeur à l'Université de Tunis El Manar, d'avoir accepté de m'encadrer dans cette thèse. Je lui témoigne ma sincère reconnaissance pour sa constante disponibilité et sa présence dans les moments difficiles.

Je tiens à exprimer ma profonde reconnaissance et ma gratitude à Mr Ibrahima Diarrassouba, Maître de Conférences à l'Université du Havre, de m'avoir encadrée dans cette thèse. J'ai découvert en lui une personne impressionnante, tant au niveau humain que professionnel. Je le remercie pour toutes ces journées de travail à Paris et au Havre. Il a su me transmettre ses connaissances et sa passion pour la recherche. Pour tout cela, je lui témoigne ma plus sincère reconnaissance.

J'ai été honorée que Mr Imed Kacem, Professeur à l'Université de Lorraine, ait accepté de rapporter ma thèse. Je lui exprime mes plus sincères remerciements pour l'intérêt qu'il a bien voulu porter à ce travail, pour les commentaires pertinents qu'il a apportés, et pour son agréable contact.
Je remercie également Mr Mohamed Didi Biha, Professeur à l'Université de Caen, de m'avoir fait l'honneur d'accepter la charge de rapporteur, pour sa lecture précise de ce manuscrit ainsi que pour l'intérêt qu'il a porté à mes travaux.
Je remercie Mr Pierre Fouilhoux, Maître de Conférences à l'Université Pierre et Marie Curie, pour l'intérêt qu'il a bien voulu porter à ma thèse et pour avoir accepté de participer à mon jury.

Je suis également très reconnaissante envers Mr Vangelis Paschos, Professeur à l'Université Paris Dauphine, d'avoir bien voulu examiner mes travaux et de m'avoir fait le plaisir de présider le jury. Je le remercie aussi pour les deux années d'enseignements que j'ai eues avec lui et pour son contact charmant et naturel.

Mes remerciements vont ensuite à mes collègues du LAMSADE. D'abord à mes voisins de bureau, sans qui cette thèse n'aurait pas été la même au C605. A Ian, pour son esprit, sa fascinante intelligence, et pour m'avoir prouvé que l'humour français pouvait être délectable. A Thomas, pour son originalité, ses débats critiques et son grand cœur. A Yassine, mon voisin et compatriote, pour son agréable compagnie et son énorme générosité. A Ioannis, $\alpha\gamma\alpha\pi\eta\tau$ο $\mu\sigma\upsilon$ $\varepsilon\lambda\lambda\eta\nu\iota\kappa$ο, dont la présence suffit pour donner le sourire. A Youcef, mon frère de thèse, pour son soutien et son amitié. A Khalil, pour sa compagnie pendant ces étés passés au Havre. A Céline, ma demi-compatriote, pour son esprit joyeux et sa spontanéité. A Boris, pour sa discrétion et sa simplicité. Je remercie également Olivier, mon ingénieur préféré, pour ses débats existentiels, pour sa bienveillance, et pour avoir été un si bon ami durant cette thèse. Je remercie ma petite Diana, pour son humour, son doux accent libanais, et pour cette nouvelle brise qu'elle a apportée. Merci à Fabien pour sa bonne humeur et sa nature bienveillante, à Anaëlle pour sa charmante présence et ses discussions joyeuses, et à Marcel pour les échanges agréables durant l'année où je l'ai connu. Je remercie aussi Pedro, Manel, Hiba, Satya, Amin, Justin, Zahra, Saeed, Mehdi, Maude et Hossein, en leur souhaitant beaucoup de courage dans leurs thèses respectives. Un grand merci à Dr. Amel Benhamiche, avec qui les échanges ont toujours été intéressants et plaisants, merci à Dr. Raouia Taktak, d'avoir été mon amie et co-auteur, et merci à Dr. Lyes Belhoul, parti trop tôt, pour sa personnalité attachante. Je remercie aussi mon amie Dr. Hager et mon pote Achraf, de m'avoir adoptée et de m'avoir soutenue durant toute cette thèse.
Mes remerciements vont aussi à toute ma famille et mes amis. Je dédie spécialement cette thèse à mes tantes Lilia, Basma et Lamia, mes oncles Lotfi, Jalel et Ramzi, et à ma grand-mère Halima, leur présence et leur soutien ont été une véritable source d'énergie aux moments les plus difficiles. Un petit merci à mon petit Seif et à ma petite Yasmine, pour la gaieté qu'ils m'ont apportée par leur simple existance.

Je dédie particulièrement cette thèse à mon père Mohamed, mon premier sponsor, sans qui cette thèse n'aurait jamais vu le jour, merci de m'avoir fait confiance et de m'avoir soutenue. Je dédie enfin cette thèse à ma mère Nadia, mon éternelle supportrice, qui est sûrement fière de voir la fin de toutes ces années d'études, merci d'avoir été là, de m'avoir toujours soutenue et aimée.

Abstract

This thesis presents a polyhedral study of survivable network design problems with high connectivity requirement. These problems have applications in telecommunications. In particular, the k-node-connected subgraph and the k-node-connected hop-constrained network design problems when $k \geq 3$ are investigated.

We first consider the k-node-connected subgraph problem. Given a weighted undirected graph $G = (V, E)$ and a positive integer k, the k-node-connected subgraph problem is to find a minimum weight subgraph of G which contains k-node-disjoint paths between every pair of nodes of V. We investigate the polytope associated with that problem when $k \geq 3$. We introduce new classes of valid inequalities and discuss their facial aspect. We also devise separation routines, investigate structural properties of the linear relaxation and discuss some reduction operations that can be used in a preprocessing phase. Using these results, we devise a Branch-and-Cut algorithm and present some computational results. Then we present a new extended formulation for the the k-node-connected subgraph problem, which holds a polynomial number of constraints and an exponential number of variables. We use this formulation to develop a Branch-and-Cut-and-Price algorithm for the problem. We also provide some computational results and a comparative study between the two formulations we have introduced for the problem.

In a second part of the thesis, we investigate the hop-constrained version of the problem. Given a graph $G = (V, E)$ with weights on the edges, a set of origin and destination pairs of nodes $D \subseteq V \times V$, and two positive integers $L \geq 2$ and $k \geq 2$, the k node-disjoint hop-constrained network design problem is to find a minimum weight subgraph of G such that between every origin and destination there exist at least k node-disjoint paths of length at most L. We investigate the structure of the associated polytope when $k \geq 2$ and $L \in \{2, 3\}$. We propose an integer linear programming formulation for the problem for $L = 2, 3$ and investigate the associated polytope. We introduce valid inequalities for the problem, and give necessary and sufficient conditions for these inequalities to be facet defining. We also devise separation algorithms, and

using these results, we propose a Branch-and-Cut algorithm for solving the problem along with some computational results for $k \geq 3$ and $L = 3$, and for $k = 2$ and $L = 4$.

Key words : k-node-connected graph, k-node-disjoint hop-constrained paths, survivable network, polytope, facet, separation, Branch-and-Cut, Branch-and-Cut-and-Price.

Résumé long

La conception des réseaux fiables est un large domaine de recherche qui est question importante dans les télécommunications. L'objectif est de concevoir des réseaux efficaces, fiables et à coût réduit, avec des caractéristiques et des exigences spécifiques sur la topologie. La fiabilité est généralement exprimée en termes de connectivité dans le réseau. Le niveau de connectivité dépend des exigences de chaque opérateur de télécommunication. Il peut être nécessaire de concevoir plusieurs chemins pour relier chaque paire de sommets pour assurer la transmission en cas de déconnexion ou de panne, tout cela au moindre coût possible.

Par conséquent, il devient nécessaire de proposer des solutions pour satisfaire la demande des utilisateurs et d'assurer une utilisation efficace des ressources du réseau.

La majorité des travaux proposés dans la littérature ont considéré uniquement les graphes k-arête-connexe. Pour plusieurs types de graphes, cette conception peut ne pas être très fiable. En effet, plusieurs nœuds ou routeurs dans le cas des réseaux de télécommunication peuvent tomber en panne et cela peut causer une mauvaise circulation et une perte des données. L'étude des graphes k-sommet-connexes devient donc un critère fondamental dans la mise en place d'une conception efficace des réseaux, pour éviter les pannes engendrant une interruption du service, et pour assurer l'acheminement permanent des données.

Un réseau peut être représenté par un graphe $G = (V, E)$ où V est l'ensemble de sommets et E, l'ensemble des arêtes. Plusieurs topologies ont été proposées pour la conception de réseaux. Cependant, la plus fréquente et la plus utile en pratique est la topologie uniforme. C'est à dire que tous les sommets du réseau ont la même importance, et il est requis que entre chaque paire de sommets il existe au moins k chemins arête-(sommet-)disjoint, où k est un entier donné tel que $k \geq 2$. Ainsi le réseau peut demeurer fonctionnel quand au plus $k - 1$ arêtes (sommets) tombent en panne. Le problème consiste à déterminer, compte tenu des poids sur les liens possibles du graphe, un sous-graphe de poids minimum satisfaisant l'arête ou la sommet connexité.

Cette thèse porte sur le problème de sommet connexité.

Étant donné un graphe, le problème du sous-graphe k-sommet-connexe est de trouver un sous-graphe de poids minimum, tel que entre chaque paire de sommets du graphe il existe k chemins sommet-disjoints entre ces deux sommets.

Cependant, cette contrainte de connectivité peut ne pas être suffisante pour garantir une grande fiabilité et une bonne qualité de routage. En fait, pour certains réseaux spéciaux tels que les VPN (réseaux privés virtuels), nous pouvons avoir besoin d'un degré de connectivité plus élevé. En outre, le chemin de routage alternatif du réseau peut être long et coûteux, ce qui peut entraîner une dégradation significative de la vitesse de transfert. Afin de limiter la longueur de réacheminement et de garantir une bonne qualité de service, il est généralement nécessaire d'assurer que la longueur (nombre d'arêtes) des chemins entre une paire origine-destination soit délimitée par un nombre donné L selon les paramètres technologiques.

Le problème est alors de déterminer, compte tenu des poids sur les liens possibles du réseau et des paires d'origine-destination, un sous-graphe de poids minimum contenant au moins k chemins sommet-disjoints entre chaque paire d'origine-destination de longueur au plus L.

Les méthodes d'optimisation combinatoire, en particulier l'approche dite polyèdrale ont montré leur efficacité pour traiter des problèmes difficiles et ayant une combinatoire importante. Initiée par Edmonds dans le cadre du problème du couplage [34], cette technique consiste à réduire la résolution d'un problème d'optimisation combinatoire à celle d'un où plusieurs programmes linéaires. Il s'agit notamment de donner, une description complète (ou partielle) du polytope des solutions du problème considéré avec un système d'inégalités linéaires. L'approche polyèdrale a montré son efficacité sur plusieurs problèmes d'optimisation combinatoire tels que le Problème du Voyageur de Commerce, le Problème de Conception de Réseau, ainsi que le Problème de la Coupe Maximum.

Dans cette thèse, nous étudions, dans un contexte polyédral, les deux problèmes de conception de réseaux fiables, le problème du sous-graphe k-sommet-connexe et le problème de conception de réseaux fiables k-sommet-connexes avec contraintes de borne. En particulier, nous examinons ces problèmes dans le cas ou un niveau élevé de connectivité est requis, c'est-à-dire lorsque $k \geq 3$. Ces deux problèmes sont NP-Difficile lorsque $k \geq 2$.

Le premier chapitre est consacré à l'introduction de quelques notions préliminaires concernant l'optimisation combinatoire, les méthodes exactes en général et l'approche

polyèdrale en particulier. Nous donnons notamment un aperçu des méthodes des plans sécants et de génération de colonnes, ainsi que des algorithmes de coupes et branchements, et de génération de colonnes et branchements. Nous donnons alors quelques définitions basiques sur la théorie des graphes et introduisons la terminologie et les notations utilisées dans ce manuscrit. Enfin, nous présentons un état de l'art sur les problèmes de conception de réseaux. Dans le chapitre suivant nous présentons le contexte pratique ainsi que les enjeux technologiques de ce travail.

Dans le deuxième chapitre, nous considérons le problème du sous-graphe k-sommet-connexe, avec $k \geq 3$. Nous nous intéressons au polyèdre associé à ce problème et à l'étude de sa dimension. Nous présentons certaines classes d'inégalités valides et nous décrivons certaines conditions pour que ces inégalités définissent des facettes pour le polytope associé. Nous étudions également les propriétés structurelles des points extrêmes de la relaxation linéaire du problème et présentons des opérations de réduction, qu'on utilise comme préprocessing avant la phase de séparation. Ces opérations permettent d'effectuer la séparation des inégalités valides dans un graphe réduit.

Le troisième chapitre porte sur l'algorithme de coupes et branchements que nous proposons pour la formulation du problème. Cet algorithme est basé sur les résultats théoriques introduits dans le chapitre précédent. Nous donnons d'abord un aperçu du fonctionnement de cet algorithme, puis nous détaillons les procédures de séparation utilisées. L'objectif de ce chapitre est de présenter la mise en oeuvre de l'approche proposée dans le chapitre précédent et de donner un aperçu de l'efficacité des contraintes utilisées en pratique. Nous présentons des résultats de calcul pour $k = 3, 4, 5$ (voir les Tableaux 1 et 2 pour $k = 3$). L'algorithme est testé sur des instances réelles issues de la SNDlib[1] et de la TSPlib[2]. Les expériences montrent en particulier l'efficacité des procédures de séparation qu'on utilise dans l'algorithme de Branch-and-Cut et l'amélioration significative fournie par les inégalités introduites pour renforcer la relaxation linéaire du problème. Les résultats de calcul qu'on obtient montrent aussi que les inégalités de F-sommet-partition, les inégalités de SP-sommet-partition et les inégalités de partition sont efficaces pour résoudre le problème. En outre, les opérations de réduction que nous utilisons se révèlent efficaces dans la phase de séparation de l'algorithme Branch-and-Cut. Les expériences montrent également que le problème devient plus facile lorsque k augmente, et est plus difficile lorsque k est impair que lorsque k est pair.

L'étude présentée dans ce chapitre montre l'efficacité de certaines inégalités valides, à savoir les F-sommet-partition, les SP-sommet-partition et les inégalités de partition, dans la résolution du problème. Il serait intéressant d'approfondir l'étude du polytope du problème et d'identifier les cas où ces inégalités définissent complètement le polytope

du problème.

Instance	#EC	#NC	#SPC	#FNPC	#NPC	COpt	Gap1	NSub1	CPU1
atlanta_15	15	606	1	17	1	3265	0.01	3	0:00:01
geant_22	72	1990	19	28	6	375	1.07	60	0:00:26
france_25	80	7500	15	36	7	3254	0.08	37	0:00:32
norway_27	68	4448	10	55	5	5730	0.76	15	0:00:43
sun_27	42	2582	8	28	0	4771	0.04	7	0:00:31
india_35	62	2231	5	26	6	452	0.33	8	0:00:53
cost266_37	135	10726	30	775	7	275	0.9	13	0:18:01
giul_39	62	2760	7	32	1	5878	0.03	5	0:02:19
pioro_40	11	2866	0	2	0	5637	0.00	1	0:00:09
germany_50	42	13094	5	14	2	112	0.01	4	0:02:37
ta2_65	124	7597	10	106	4	5334	0.07	9	0:43:55

Table 1: Résultats pour les instances de la SNDLIB avec $k = 3$.

Instance	#EC	#NC	#SPC	#FNPC	#NPC	COpt	Gap1	NSub1	CPU1
bays_29	74	3709	11	39	8	14815	1.01	19	0:01:10
dantzig_42	137	9156	12	32	16	1232	0.03	42	0:14:14
att_48	138	13995	14	47	10	17527	0.02	48	0:42:11
eil_51	55	4680	7	30	1	745	0.02	4	0:06:41
berlin_52	133	9518	26	95	10	12644	0.22	30	0:27:05
eil_76	80	15321	8	84	4	947	0.11	8	0:48:05
gr_96	174	330	19	6	0	915	0.6	2	2:03:11
rat_99	112	294	9	19	0	2105	0.3	32	2:02:26
kroA_100	169	305	24	13	1	36492	0.21	2	2:04:35
rd_100	186	303	21	3	0	13391	0.13	21	2:01:03
kroB_100	145	300	12	52	1	37341	1.6	12	2:03:58
lin_105	214	317	15	6	6	24870	2.4	35	2:01:44
gr_120	90	332	10	0	0	11562	0.6	2	2:26:57
bier_127	136	364	16	2	0	199863	3.2	23	2:42:41
pr_124	179	403	12	0	0	99696	0.29	3	2:28:01
ch_130	122	371	10	0	0	10571	7.1	12	2:48:25
kroA_150	130	415	1	2	0	44952	2.6	23	2:49:56
*u_159	112	429	3	7	0	71772	*8.9*	59	5:00:00

Table 2: Résultats pour les instances de la TSPLIB avec $k = 3$.

Dans le quatrième chapitre, nous présentons une formulation étendue pour le problème du sous-graphe k-sommet-connexe pour $k = 2$. Nous étudions un algorithme de génération de colonne et des heuristiques efficaces pour résoudre le problème. Et en utilisant ces résultats, nous mettons au point un algorithme Branch-and-Cut-and-Price pour résoudre le problème.

Les résultats de calcul montrent que l'algorithme de génération de colonnes est effi-

cace pour résoudre le problème et pour produire une bonne borne supérieure pour le problème (voir Tableaux 3 et 4). En outre, on démontre que l'algorithme de Branch-and-Cut est plus efficace pour résoudre le problème à l'optimalité. Nous observons également que lorsque l'on s'approche d'une valeur de LP optimale, l'algorithme de génération de colonnes a du mal à trouver la valeur optimale bien qu'elle soit proche. Ce phénomène est connu comme l'effet de "tailing off". Comme décrit dans [25], une explication principale pour l'effet de "tailing off" est que les variables convergent lentement vers leur valeur optimale respective et, d'une itération à l'autre, prennent des valeurs non liées et aléatoires.

Table 3: Résultats pour les instances de la SNDLIB.

Instance	$\|V\|$	Gen_Cuts	Gen_Cols	COpt	Gap	NSub	CPU
diyuan	11	2	0	989	0.00	1	0:00:02
pdh_12	11	19	0	18	0.00	1	0:00:03
abilene	12	19	24	121	0.04	15	0:00:18
atlanta	12	17	28	115	0.00	1	0:00:03
polska	12	26	4	27	0.05	11	0:00:01
nobel-us	14	32	0	117	0.00	5	0:00:02
newyork	16	24	56	2335	0.00	1	0:00:04
geant	22	58	80	226	0.02	15	0:00:20
ta1	24	45	27	1859	0.02	27	0:00:31
france	25	52	43	1985	0.01	7	0:00:48
janos-us	26	64	626	178	0.06	25	0:00:21
sun	27	50	22	2849	0.00	1	0:00:08
norway	27	91	1068	3847	0.08	139	0:01:19
janos-us-ca	37	85	14	201	0.06	27	0:00:24
cost266	37	118	7876	173	0.07	103	0:12:59
giul39	39	149	6576	3949	0.05	147	0:09:46
pioro	40	81	2168	3625	0.03	11	0:01:44
germany	50	67	4948	56	0.1	33	0:52:34
ta2	65	148	2439	3455	0.04	3	1:06:10

Comme que travail futur, nous pouvons considérer des solutions pour diminuer cet effet, ce qui pourra améliorer l'efficacité de la résolution en introduisant un algorithme qui stabilise et accélère le processus de solution tout en restant dans le cadre de programmation linéaire. L'algorithme de stabilisation peut être utilisé pour améliorer le temps de résolution des solutions pour les cas difficiles et pour résoudre des problèmes plus importants.

On peut également essayer d'étendre l'approche développée dans ce chapitre pour

Table 4: Résultats pour les instances de la TSPLIB.

Instance	$\|V\|$	Gen_Cuts	Gen_Cols	COpt	Gap	NSub	CPU
burma14	14	38	124	3841	0.06	29	0:00:08
ulysses22	22	56	868	8180	0.07	47	0:00:31
fri26	26	67	65	965	0.01	23	0:01:33
bays	29	84	1498	9884	0.05	55	0:03:41
dantzig42	42	82	1	699	0.00	1	0:00:20
eil51	51	114	6847	482	0.06	19	4:59:01
berlin52	52	141	127	8181	0.04	7	0:05:48
st70	70	191	2970	750	0.05	3	4:51:05
pr76	76	240	2595	130921	0.1	5	4:53:32
kroC100	100	376	2135	23295	0.05	46	2:45:27
lin105	105	420	8771	16766	0.07	36	4:41:23
ch130	130	331	3722	7129	0.07	48	4:55:44
gr137	137	363	5579	79460	0.06	25	5:00:00

étudier le problème pour $k \geq 3$ avec une formulation étendue utilisant des variables de chemin et concevoir des algorithmes efficaces pour le problème dans ce cas.

Dans le cinquième chapitre, nous étudions le problème de conception de réseaux fiables k-sommet-connexes avec contraintes de borne. Le problème consiste à trouver un sous-graphe de poids minimum, tel que entre chaque paire d'origine-destination donnée, il existe k chemins sommet-disjoints de longueur au plus L, où L est un entier donné. Nous étudions le problème d'un point de vue polyédral. Nous proposons une formulation linéaire en nombres entiers pour $L = 2, 3$. Nous étudions le polytope associé au problème et introduisons de nouvelles inégalités valides pour $L \in \{2, 3, 4\}$, ainsi que des conditions nécessaires et des conditions suffisantes sous lesquelles ces inégalités définissent des facettes.

Finalement, dans le sixième et dernier chapitre nous élaborons un algorithme Branch-and-Cut pour résoudre le problème présenté dans le chapitre précédent en utilisant les inégalités que nous avons présentées précédemment. En particulier, nous discutons le problème de séparation des inégalités de st-cut, 3-st-path-cut, st-node-cut, et de 3-st-node-cut. Enfin, nous présentons les résultats expérimentaux pour le problème lorsque $L = 3$ et $k = 3, 4, 5$ d'une part, et lorsque $L = 4$ et $k = 2$ d'autre part (voir Tableaux 5 et 6).

Table 5: Résultats pour $k = 3$, $L = 3$ et des demandes routées.

$\|V\|$	$\|D\|$	C-LPC	NC-NLPC	SP	DC	P	COpt	Gap	NSub	CPU
r 21	15	11775	74	10	0	0	5526	9.23	2265	00:04:46
r 21	17	22356	228	0	0	0	5939	9.4	4518	00:18:24
r 21	20	71354	116	0	0	0	6466	9.54	17673	03:10:39
r 30	15	15599	264	14	0	0	10109	6.87	1521	00:12:06
r 30	20	58659	1516	12	0	0	11376	8.41	15280	05:00:00
r 30	25	80999	615	18	0	0	12661	12.33	14281	05:00:00
r 48	20	51038	1632	26	0	0	18337	18.1	6133	05:00:00
r 48	30	66277	898	10	0	0	25437	28.68	5305	05:00:00
r 48	40	69242	257	2	0	0	31693	30.17	5628	05:00:00
r 52	20	49717	1674	22	0	0	11170	9.15	5707	05:00:00
r 52	30	62698	1692	18	0	0	14626	17.11	3845	05:00:00
r 52	40	68794	1024	16	0	0	17920	21.86	4953	05:00:00
r 52	50	77808	142	0	0	0	20873	24.49	4397	05:00:00

Table 6: Résultats pour $k = 3$, $L = 3$ et des demandes arbitraires.

$\|V\|$	$\|D\|$	C-LPC	NC-NLPC	SP	DC	P	COpt	Gap	NSub	CPU
a 21	10	56593	2	0	483	0	6680	8.66	9191	05:00:00
a 21	11	29325	2	0	375	1	6770	6.8	2614	00:57:32
a 30	10	44057	25	18	38	0	10354	6.64	13274	05:00:00
a 30	15	53545	86	0	462	0	13936	11.69	6399	05:00:00
a 48	15	34047	0	0	20	0	-	-	1119	05:00:00
a 48	20	28329	0	2	10	0	-	-	229	05:00:00
a 48	24	23975	0	0	11	0	-	-	103	05:00:00
a 52	20	30157	108	6	41	0	-	-	1735	05:00:00
a 52	26	24217	0	0	96	0	-	-	307	05:00:00

Les instances que nous avons testés dans ce chapitre montrent que l'algorithme de Branch-and-Cut est assez efficace pour résoudre le problème lorsque $L = 3$ et $k = 3, 4, 5$, et ceci, pour les deux ensembles d'instances, arbitraires et routées. Il est également à souligner que les instances de grande taille sont encore difficiles à résoudre dans les 5 heures de temps maximal, mais les gaps obtenus sont dans la plupart des cas sont assez intéressants. De plus, les expériences montrent l'importance des inégalités de double-cut et de Steiner-SP-partition, alors que les inégalités de partition semblent être moins efficaces.

Il convient également de noter que, contrairement au problème de conception de réseaux fiables sans contraintes de borne, nos expériences ne permettent pas de conclure sur l'impact d'une augmentation de la connectivité k sur la résolution du problème. En fait, les expériences antérieures faites pour le problème de conception de réseaux fiables ont conclu que le problème sans considérer les contraintes de borne semble devenir plus

facile lorsque k augmente. Dans notre cas, l'impact de la connectivité sur la résolution est moins clair. Il semble même, lors de la comparaison des résultats pour $L = 3$ et $L = 4$, que le problème devient plus difficile à résoudre lorsque L augmente.

L'étude du calcul souligne qu'un très grand nombre d'inégalités de st-cut et de 3-st-path-cut ont été générées lors de la résolution du problème. Cela peut être un problème puisqu'il oblige l'algorithme Branch-and-Cut de gérer un énorme pool de contraintes et peut impliquer une consommation de temps de CPU excessive pour la gestion de ces contraintes. Cela peut empêcher l'algorithme d'avoir une bonne exploration de l'arbre Branch-and-Cut.

Les expériences que nous avons effectuées pour le problème de conception de réseaux fiables k-sommet-connexes avec contraintes de borne pour $k = 3, 4, 5$ et $L = 3$ ont un temps CPU relativement élevé. Ces observations suggèrent qu'un algorithme efficace pour le problème nécessite une formulation plus étroite pour le problème. On peut mener une étude plus approfondie sur le polytope du problème afin de fournir plus d'inégalités définissant des facettes et de produire un algorithme efficace de coupe et branchement.

On peut également utiliser les graphes orientés appropriés et en exploitant les résultats connus sur les problèmes de chemins disjoints dans les graphes orientés. Cela peut aider à fournir de nouvelles facettes pour le problème. Il serait également intéressant, d'un point de vue algorithmique, d'améliorer les procédures de séparation prévues pour les différentes inégalités que nous avons introduites dans ce travail. Nous devons développer des heuristiques de séparation plus efficaces pour l'algorithme de Branch-and-Cut. Il sera également intéressant de se concentrer sur des méthodes de prétraitement plus sophistiquées afin de faciliter la résolution du problème. Il sera également intéressant d'effectuer les expérimentations pour $L = 4$ et tout k. Le même type d'étude peut être utilisé pour le problème lorsque $L \geq 5$. Si possible, cela peut fournir une formulation en nombres entiers pour le problème ainsi qu'un algorithme de Branch-and-Cut pour tout $k \geq 2$ et $L = 5$.

Il existe de nombreuses directions dans lesquelles la recherche dans cette thèse peut être poursuivie pour les deux problèmes considérés. À des fins théoriques, il devrait être intéressant d'étudier le polyope du problème dans certains cas spéciaux, comme par exemple lorsque le graphe est série-parallèle. En outre, on pourrait étudier le problème du point de vue de la répartition des demandes, car cela peut influencer la description

polyédrale des solutions du problème et probablement l'efficacité des algorithmes de résolution.

Une autre question intéressante serait de voir si l'on peut utiliser des modèles dirigés pour le problème. Cela peut fournir des formulations linéaire en nombres entiers plus forte. C'est l'une de nos lignes de recherche à l'avenir.

Contents

Introduction

The design of survivable networks is an important issue in telecommunications. The aim is to conceive cheap, efficient and reliable networks with specific characteristics and requirements on the topology. Survivability is generally expressed in terms of connectivity in the network. The level of connectivity depends on the need of each telecommunication operator. We may have to conceive several paths to link each pair of nodes to ensure the transmission in case of disconnection or breakdown, all this at the cheapest possible cost. Therefore, it becomes necessary to propose solutions to satisfy user demand, and ensure an efficient use of network resources.

A network can be represented by a graph $G = (V, E)$ where V is the set of nodes and E, the set of edges. Different topologies have been proposed to design survivable networks. However, as we can see in [54, 55], the most frequent and useful case in practice is the uniform topology. This means that the nodes of the network have all the same importance and it is required that between every pair of nodes there are at least k edge- (node-) disjoint paths, where k is a fixed integer such that $k \geq 2$. Thus the network will be still functional when at most $k-1$ edges (nodes) fail. The underlaying problem is to determine, given weights on the possible links of the network, a minimum weight network satisfying the edge or the node connectivity. This thesis deals with the node connectivity of the problem.

However this connectivity requirement may not unfortunately be sufficient to guarantee a high survivability and a routing quality. In fact, for some special networks such as VPN (Virtual Private Networks), we may need a higher degree of connectivity. Moreover, the alternative routing path in the network may be too long and costly and this may cause a significant degradation in the transfer speed. In order to limit the rerouting length and guarantee a high QoS, it is commonly required that the length (number of edges) of the paths between an origin-destination pair is bounded by a given number L depending on technological parameters. The problem is then to determine, given weights on the possible links of the network, and pairs of origin-destinations, a minimum weight network containing at least k edge (node) disjoint paths between each

pair of origin-destination of length no more than L. This thesis deals with the node connectivity case of the problem.

As the 2-connected topolgy ($k = 2$) provides an adequate level of survivability since most failure usually can be repaired relatively quickly, we will study the problem for $k = 2$, with a different, but related method to solve MIPs, namely the branch-and-price method and its extension, the branch-cut-and-price method. It relies on exploiting problem structures in a MIP via a decomposition. The problem is split into a coordinating problem and one or more typically well structured subproblems that can often be solved efficiently. For many difficult but well structured combinatorial optimization problems, this approach leads to a better performance than a branch-and-cut algorithm.

The survivable network design problem has been widely studied when the paths required between the nodes are edge-disjoint or when the connectivity requirement is low ($k = 2$). However the high node-connectivity requirement case ($k \geq 3$) has received a little attention. In this thesis, we study the survivable network design problem with high node-connectivity requirement. In particular, we focus on two variants of the problem: when k-node-disjoint paths are required between every pair of nodes (the k-node-connected subgraph problem) and when k-node-disjoint paths of length at most L are required between certain pairs of nodes (the k-node-disjoint hop-constrained network design problem). The study is led using the polyhedral approach and provides exact and efficient algorithms to solve these problems.

This thesis is organized as follows. In Chapter 1, we present the basic notions of combinatorial optimization and notations that will be used throughout this thesis. We laso present a state-of-the-art on survivable network design problems. Chapters 2 and 3 concern the first considered network design problem, that is the k-Node-Connected Subgraph Problem (kNCSP). We investigate the kNCSP polytope and present several classes of valid inequalities. Then we discuss the conditions under which these inequalities define facets of the polytope. We also consider the polytope associated with the linear relaxation of the problem and present some structural proporties as well as some reduction operations. Using these results we devise a Branch-and-Cut algorithm, and we give some experimental results. In Chapter 5 and 6 we discuss the hop-contrained version of the first problem, namely the k Node-Disjoint Hop-Constrained survivable network Problem(kNDHP). We introduce a new integer programming formulation for this problem and study the associated polytope. We devise a Branch-and-Cut algorithm for the problem and present extensive computational results. In Chapter 4 we devise a Branch-and-Cut-and-Price algorithm for the 2-node-connected subgraph problem and present extensive computational results.

Chapter 1

Preliminaries Notions and State-of-the-Art

Contents

This chapter is dedicated to the presentation of some preliminary notions concerning combinatorial optimization, exact approaches and polyhedra. In particular, we give an

overview of cutting planes methods as well as Branch-and-Cut algorithms. We then give some basic definitions in graph theory and introduce some notations and terminology that will be used throughout the dissertation. Finally, we give a state-of-the-art on network design problems.

1.1 Preliminaries Notions

1.1.1 Combinatorial optimization

Combinatorial optimization is the field of discrete optimization problems. In many applications, the most important decisions (control variables) are discrete in nature. Binary variables model on/off decisions to buy, invest, hire, send a vehicle, or enforce a precedence. Integer variables model indivisible quantities. Extra variables can represent continuous adjustments or amounts. This results in models known as mixed integer programs (MIP), where the relationships between variables and input parameters are expressed as linear constraints and the goal is defined as a linear objective function. MIPs are among the most widely used modeling tools. They allow a fair description of reality; they are versatile; they can handle many non-linearities (and even non-convexities) in the cost function and in the constraints; they are also well-suited for global optimization. However useful they may be, these models are notoriously difficult to solve: good quality estimations of the optimal value (bounds) are required to prune enumeration-based global-optimization algorithms whose complexity is exponential. In the standard approach to solving an MIP is so-called branch-and-bound algorithm : (i) one solves the linear programming (LP) relaxation using the simplex method; (ii) if the LP solution is not integer, one adds a disjunctive constraint on a factional component (rounding it up or down) that defines two sub-problems; (iii) one applies this procedure recursively, thus defining a binary enumeration tree that can be pruned by comparing the local LP bound to the best known integer solution. State-of-the-art MIP solvers, such as the commercial solvers CPLEX of Ilog or Dash-Optimization's Xpress-mp, are remarkably effective. But many real-life applications remain beyond their scope, and the scientific community is actively seeking to extend the capabilities of MIP solvers. Developments made in the context of specific applications often become generic tools over time and see their way into commercial software.

The most effective solution schemes are a complex blend of techniques: cutting planes to better approximate the convex hull of feasible (integer) solutions and hence provide better LP bounds, Lagrangian decomposition methods to produce alternative powerful

relaxations, constraint programming to actively reduce the solution domain through logical implications, heuristics and meta-heuristics (greedy, local improvement, or randomized partial search procedures) to produce good candidate solutions, and specialized branch-and-bound or dynamic programming enumeration schemes to find a global optimum. The real challenge is to integrate the most efficient methods into one global system. Another key to further progress is the development of stronger problem formulations whose relaxations provide approximations that enable enhanced truncation of enumerative solution schemes. Tighter formulations are also much more likely to yield good quality approximate solutions through rounding techniques. With properly chosen formulations, exact optimization tools can be competitive with other methods (such as meta-heuristics) in constructing good approximate solutions within limited computational time, and of course has the important advantage of being able to provide a performance guarantee through the relaxation bounds.

Combinatorial Optimization is a branch of operations research related to computer science and applied mathematics. Its purpose is the study of optimization problems where the set of feasible solutions is discrete or can be represented as a discrete one. Typically, the problems concerned with combinatorial optimization are those formulated as follows. Let $E = \{e_1, \ldots, e_n\}$ be a finite set called *basic set* where each element e_i is associated with a weight $c(e_i)$. Let $\mathcal{F}$ be a family of subsets of E. If $F \in \mathcal{F}$, then $c(F) = \sum_{e_i \in F} c(e_i)$ denotes the weight of F. The problem consists in identifying an element F^* of $\mathcal{F}$ whose weight is minimum or maximum. In other words,

$$\min(or \max)\{c(F) : F \in \mathcal{F}\}.$$

Such a problem is called *combinatorial optimization problem*. The set $\mathcal{F}$ represents the set of feasible solutions of the problem.

The term *combinatorial* refers to the discrete structure of $\mathcal{F}$. In general, this structure is represented by a graph. The term *optimization* signifies that we are looking for the best element in the set of feasible solutions. This set generally contains an exponential number of solutions, therefore, one can not expect to solve a combinatorial optimization problem by exhaustively enumerate all its solutions. Such a problem is then considered as a hard problem.

Various effective approaches have been developed to tackle combinatorial optimization problems. Some of these approaches are based on graph theory, while others use linear and non-linear programming, integer programming and polyhedral approach. Besides, several practical problems arising in real life, can be formulated as combinatorial optimization problems. Their applications are in fields as diverse as telecommunications, transport, industrial production planing or staffing and scheduling in airline

companies. Over the years, the discipline got thus enriched by the structural results related to these problems. And, conversely, the progress made in computed science have made combinatorial optimization approaches even more efficient on real-world problems.

In fact, combinatorial optimization is closely related to algorithm theory and computational complexity theory as well. The next section introduces computational issues of combinatorial optimization.

1.1.2 Computational complexity

Computational complexity theory is a branch of theoretical computer science and mathematics, whose study started with works of Cook [30], Edmonds [33] and Karp [53]. Its objective is to give a classify a given problem depending on its difficulty. A plentiful literature can be find on this topic, see for example [38] for a detailed presentation of NP-completeness theory.

A *problem* is a question having some input parameters, and to which we aim to find an answer. A problem is defined by giving a general description of its parameters, and by listing the properties that must be satisfied by a solution. An *instance* of the problem is obtained by giving a specific value to all its input parameters. An *algorithm* is a sequence of elementary operations that allows to solve the problem for a given instance. The number of input parameters necessary to describe an instance of a problem is the *size* of that problem.

An algorithm is said to be polynomial if the number of elementary operations necessary to solve an instance of size n is bounded by a polynomial function in n. We define the class P as the class gathering all the problems for which there exists some polynomial algorithm for each problem instance. A problem that belongs to the class P is said to be "easy" or "tractable".

A *decision problem* is a problem with a *yes* or *no* answer. Let $\mathcal{P}$ be a decision problem and $\mathcal{I}$ the set of instances such that their answer is yes. $\mathcal{P}$ belongs to the class *class NP* (Nondeterministic Polynomial) if there exists a polynomial algorithm allowing to check if the answer is yes for all the instances of $\mathcal{I}$. It is clear that a problem belonging to the class P is also in the class NP. Although the difference between P and NP has not been shown, it is a highly probable conjecture.

In the class NP, we distinguish some problems that may be harder to solve than others. This particular set of problems is called *NP-complete*. To determine whether

a problem is NP-complete, we need the notion of *polynomial reducibility*. A decision problem P_1 can be polynomially reduced (or transformed) into an other decision problem P_2, if there exists a polynomial function f such that for every instance I of P_1, the answer is "yes" if and only if the answer of $f(I)$ for P_2 is "yes". A problem $\mathcal{P}$ in NP is also NP-complete if every other problem in NP can be reduced into $\mathcal{P}$ in polynomial time. The Satisfiability Problem (SAT) is the first problem that was shown to be NP-complete (see [30]).

With every combinatorial optimization problem is associated a decision problem. Furthermore, each optimization problem whose decision problem is NP-complete is said to be *NP-hard*. Note that most of combinatorial optimization problems are NP-hard. One of the most efficient approaches developed to solve those problems is the so-called polyhedral approach.

1.1.3 Polyhedral approach and Branch-and-Cut

1.1.4 Elements of polyhedral theory

The polyhedral method was initiated by Edmonds in 1965 [34] for a matching problem. It consists in describing the convex hull of problem solutions by a system of linear inequalities. The problem reduces then to the resolution of a linear program. In most of the cases, it is not straightforward to obtain a complete characterization of the convex hull of the solutions for a combinatorial optimization problem. However, having a system of linear inequalities that partially describes the solutions polyhedron may often lead to solve the problem in polynomial time. This approach has been successfully applied to several combinatorial optimization problems. In this section, we present the basic notions of polyhedral theory. The reader is referred to works of Schrijver [63] and [56].

We shall first recall some definitions and properties related to polyhedral theory.

Let n be a positive integer and $x \in \mathbb{R}^n$. e say that x is a *linear combination* of x_1, x_2, ..., $x_m \in \mathbb{R}^n$ if there exist m scalar λ_1, λ_2, ..., λ_m such that $x = \sum_{i\in 1}^{m} \lambda_i x_i$. If $\sum_{i=1}^{m} \lambda_i = 1$, then x is said to be a *affine combination* of x_1, x_2, ..., x_m. Moreover, if $\lambda_i \geq 0$, for all $i \in \{1, \ldots, m\}$, we say that x is a *convex combination* of x_1, x_2, ..., x_m.

Given a set $S = \{x_1, \ldots, x_m\} \in \mathbb{R}^{n\times m}$, the *convex hull* of S is the set of points $x \in \mathbb{R}^n$ which are convex combination of x_1, ..., x_m (see Figure 1.1), that is

$$conv(S) = \{x \in \mathbb{R}^n | x \text{ is a convex combination of } x_1, \ldots, x_m\}.$$

The points $x_1, \ldots, x_m \in \mathbb{R}^n$ are *linearly independents* if the unique solution of the

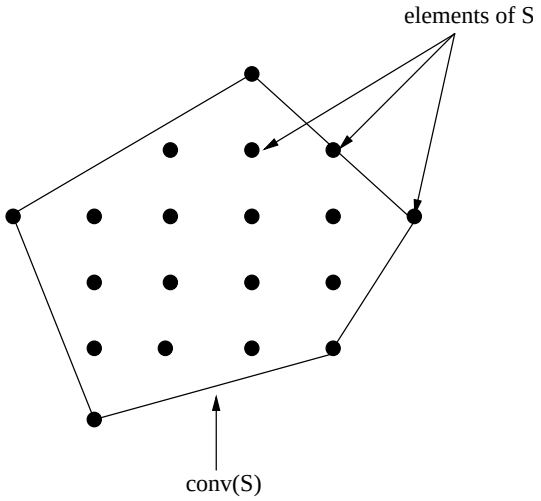

Figure 1.1: A convex hull

system $\sum_{i=1}^{m} \lambda_i x_i = 0$ is $\lambda_i = 0$, for all $i \in \{1, \ldots, m\}$. They are *affinely independent* if the unique solution of the system

$$\sum_{i=1}^{m} \lambda_i x_i = 0, \sum_{i=1}^{m} \lambda_i = 1,$$

is $\lambda_i = 0$, $i = 1, \ldots, m$.

A *polyhedron* P is the set of solutions of a linear system $Ax \leq b$, that is $P = \{x \in \mathbb{R}^n | Ax \leq b\}$, where A is a m-row n-columns matrix and $b \in \mathbb{R}^m$. A *polytope* is a bounded polyhedron. A point x of P will be also called a *solution* of P.

A polyhedron P is said to be of *dimension* p if it has at most $p+1$ affinely independent solutions. We denote it by $dim(P) = p$. We also have that $dim(P) = n$ - $rank(A^{=})$, where $A^{=}$ is the submatrix of A of *inequalities* that are satisfied with equality by all tje solutions of P (implicit equalities). The polyhedron P is full dimensional if $dim(P) = n$.

An inequality $ax \leq \alpha$ is *valid* for a polyhedron $P \subseteq \mathbb{R}^n$ if for every solution $\overline{x} \in P$, $a\overline{x} \leq \alpha$. This inequality is said to be *tight* for a solution $\overline{x} \in P$ if $a\overline{x} = \alpha$. The inequality $ax \leq \alpha$ is *violated* by $\overline{x} \in P$ if $a\overline{x} > \alpha$. Let $ax \leq \alpha$ be a valid inequality for the polyhedron P. $F = \{x \in P | ax = \alpha\}$ is called a *face* of P. We also say that F is a *face induced by* $ax \leq \alpha$. If $F \neq \emptyset$ and $F \neq P$, we say that F is a *proper* face of P. If F is a proper face and $dim(F) = dim(P) - 1$, then F is called a *facet* of P. We also say that $ax \leq \alpha$ induces a facet of P or is a *facet defining* inequality.

If P is full dimensional, then $ax \leq \alpha$ is a facet of P if and only if F is a proper face and there exists a facet of P induced by $bx \leq \beta$ and a scalar $\rho \neq 0$ such that $F \subseteq \{x \in P | bx = \beta\}$ and $b = \rho a$.

If P is not full dimensional, then $ax \leq \alpha$ is a facet of P if and only if F is a proper face and there exists a facet of P induced by $bx \leq \beta$, a scalar $\rho \neq 0$ and $\lambda \in \mathbb{R}^{q \times n}$ (where q is the number of lines of matrix $A^{=}$) such that $F \subseteq \{x \in P | bx = \beta\}$ and $b = \rho a + \lambda A^{=}$.

An inequality $ax \leq \alpha$ is *essential* if it defines a facet of P. It is *redundant* if the system $A'x \leq b'\}$ obtained by removing this inequality from $Ax \leq b$ defines the same polyhedron P. This is the case when $ax \leq \alpha$ can be written as a linear combination of inequalities of the system $A'x \leq b'$. A *complete minimal linear description* of a polyhedron consists of the system given by its facet defining inequalities and its implicit equalities.

A solution is an *extreme point* of a polyhedron P if and only if it cannot be written as the convex combination of two different solutions of P. It is equivalent to say that x induces a face of dimension 0. The polyhedron P can also be described by its extreme points. In fact, every solution of P can be written as a convex combination of some extreme points of P.

Figure 1.2 illustrates the main definitions given is this section.

1.1.5 Cutting plane method

Now let $\mathcal{P}$ be a combinatorial optimization problem, E its basic set, $c(.)$ the weight function associated with the variables of $\mathcal{P}$ and $\mathcal{S}$ the set of feasible solutions. Suppose that $\mathcal{P}$ consists in finding an element of $\mathcal{S}$ whose weight is maximum. If $F \subseteq E$, then 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 polyhedron $P(\mathcal{S}) = conv\{x^S | S \in \mathcal{S}\}$ is the *polyhedron of the solutions* of $\mathcal{P}$ or *polyhedron associated with* $\mathcal{P}$. $\mathcal{P}$ is thus equivalent to the linear program $max\{cx | x \in P(\mathcal{S})\}$. Notice that the polyhedron $P(\mathcal{S})$ can be described by a set of a facet defining inequalities. And when all the inequalities of this set are known, then solving $\mathcal{P}$ is equivalent to solve a linear program.

Recall that the objective of the polyhedral approach for combinatorial optimization problems is to reduce the resolution of $\mathcal{P}$ to that of a linear program. This reduction induces a deep investigation of the polyhedron associated with $\mathcal{P}$. It is generally not

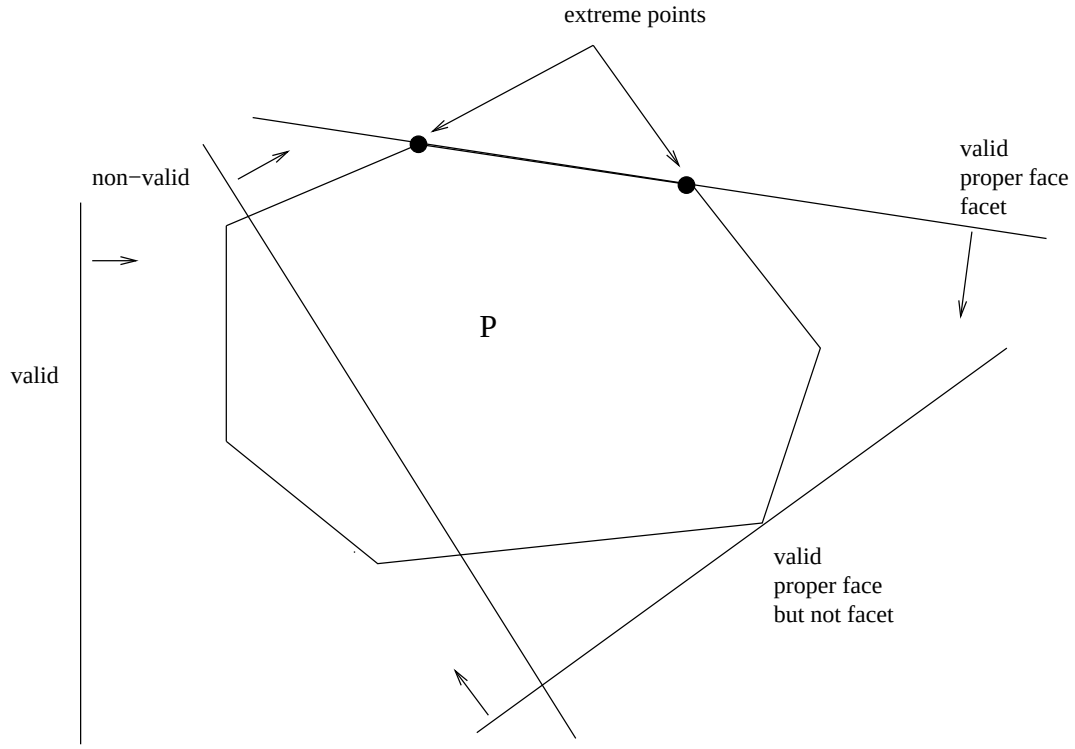

Figure 1.2: Valid inequality, facet and extreme points

easy to characterize the polyhedron of a combinatorial optimization problem by a system of linear inequalities. In particular, when the problem is NP-hard there is a very little hope to find such a characterization. Moreover, the number of inequalities describing this polyhedron is, most of the time, exponential. Therefore, even if we know the complete description of that polyhedron, its resolution remains in practice a hard task because of the large number of inequalities.

Fortunately, a technique called the *cutting plane method* can be used to overcome this difficulty. This method is described in what follows.

The cutting plane method is based on the so-called *separation problem*. This consists, given a polyhedron P of $\mathbb{R}^n$ and a point $x^* \in \mathbb{R}^n$, in verifying whether if x^* belongs to P, and if this is not the case, to identify an inequality $a^T x \leq b$, valid for P and violated by x^*. In the later case, we say that the hyperplane $a^T x = b$ separates P and x^* (see Figure).

Grötschel, Lovász and Schrijver [43] have established the close relationship between separation and optimization. In fact, they prove that optimizing a problem over a polyhedron P can be performed in polynomial time if and only if the separation problem

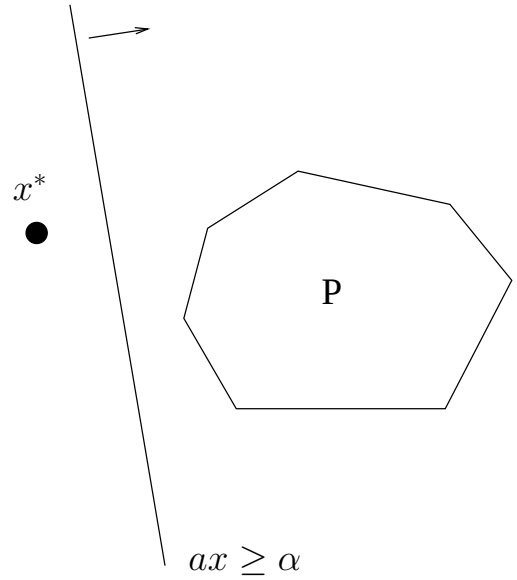

Figure 1.3: A hyperplan separating x^* and P

associated with P can be solved in polynomial time. This equivalence has permitted an important development of the polyhedral methods in general and the cutting plane method in particular. More precisely, the *cutting plane* method consists in solving successive linear programs, with possibly a large number of inequalities, by using the following steps. Let $LP = \max\{cx, Ax \leq b\}$ be a linear program and LP' a linear program obtained by considering a small number of inequalities among $Ax \leq b$. Let x^* be the optimal solution of the latter system. We solve the separation problem associated with $Ax \leq b$ and x^*. This phase is called the *separation phase*. If every inequality of $Ax \leq b$ is satisfied by x^*, then x^* is also optimal for LP. If not, let $ax \leq \alpha$ be an inequality violated by x^*. Then we add $ax \leq \alpha$ to LP' and repeat this process until an optimal solution is found. Algorithm 1 summarizes the different cutting plane steps.

Note that at the end, a cutting-plane algorithm may not succeed in providing an optimal solution for the underlying combinatorial optimization problem. In this case a *Branch-and-Bound algorithm* can be used to achieve the resolution of the problem, yielding to the so-called *Branch-and-Cut algorithm*.

1.1.6 Branch-and-Cut algorithm

Consider again a combinatorial optimization problem $\mathcal{P}$ and suppose that $\mathcal{P}$ is equivalent to $\max\{cx|Ax \leq b, x \in \{0,1\}^n\}$, where $Ax \leq b$ has a large number of inequali-

Algorithm 1: A cutting plane algorithm

Data: A linear program LP and its system of inequalities $Ax \leq b$
Result: Optimal solution x^* of LP
Consider a linear program LP' with a small number of inequalities of LP;
Solve LP' and let x^* be an optimal solution;
Solve the separation problem associated with $Ax \leq b$ and x^*;
if *an inequality* $ax \leq \alpha$ *of* LP *is violated by* x^* **then**
    Add $ax \leq \alpha$ to LP';
    Repeat step 2 ;
end
else
    x^* is optimal for LP;
    return x^*;
end

ties. A Branch-and-Cut algorithm starts by creating a Branch-and-Bound tree whose root node corresponds to a linear program $LP_0 = max\{cx|A_0x \leq b_0, x \in \mathbb{R}^n\}$, where $A_0x \leq b_0$ is a subsystem of $Ax \leq b$ having a small number of inequalities. Then we solve the linear relaxation of $\mathcal{P}$ that is $LP = \{cx|Ax \leq b, x \in \mathbb{R}^n\}$ using a cutting plane algorithm whose starting from LP_0. Let x_0^* denote its optimal solution and $A_0'x \leq b_0'$ the set of inequalities added to LP_0 at the end of the cutting plane phase. If x_0^* is integral, then it is optimal. If x_0^* is fractional, then we perform a *branching phase*. This step consists in choosing a variable, say x^1, with a fractional value and adding two nodes P_1 and P_2 in the Branch-and-Cut tree. The node P_1 corresponds to the linear program $LP_1 = \max\{cx|A_0x \leq b_0, A_0'x \leq b_0', x^1 = 0, x \in \mathbb{R}^n\}$ and $LP_2 = \max\{cx|A_0x \leq b_0, A_0'x \leq b_0', x^1 = 1, x \in \mathbb{R}^n\}$. We then solve the linear program $\overline{LP}_1 = \max\{cx|Ax \leq b, x^1 = 0, x \in \mathbb{R}^n\}$ (resp., $\overline{LP}_2 = \max\{cx|Ax \leq b, x^1 = 1, x \in \mathbb{R}^n\}$) by a cutting plane method, starting from LP_1 (resp. LP_2). If the optimal solution of $\overline{LP}_1$ (resp. $\overline{LP}_2$) is integral then, it is feasible for $\mathcal{P}$. Its value is then a lower bound of the optimal solution of $\mathcal{P}$, and the node P_1 (resp. P_2) becomes a leaf of the Branch-and-Cut tree. If the solution is fractional, then we select a variable with a fractional value and add two children to the node P_1 (resp. P_2), and so on.

Note that sequentially adding constraints of type $x^i = 0$ and $x^i = 1$, where x^i is a fractional variable, may lead to an infeasible linear program at a given node of the Branch-and-Cut tree. Or, if it is feasible, its optimal solution may be worse than the best known lower bound of the problem. In both cases, that node is pruned from the Branch-and-Cut tree. The algorithm ends when all nodes have been explored and the

optimal solution of $\mathcal{P}$ is the best feasible solution given by the Branch-and-Bound tree.

This algorithm can be improved by computing a good lower bound of the optimal solution of the problem before it starts. This lower bound can be used by the algorithm to prune the node which will not allow an improvement of this lower bound. This would permit to reduce the number of nodes generated in the Branch-and-Cut tree, and hence, reduce the time used by the algorithm. Furthermore, this lower bound may be improved by comparing at each node of the Branch-and-Cut tree a feasible solution when the solution obtained at the root node is fractional. Such a procedure is referred to as a *primal heuristic*. It aims to produce a feasible solution for $\mathcal{P}$ from the solution obtained at a given node of the Branch-and-Cut tree, when this later solution is fractional (and hence infeasible for $\mathcal{P}$). Moreover, the weight of this solution must be as best as possible. When the solution computed is better than the best known lower bound, it may significantly reduce the number of generated nodes, as well as the CPU time. Moreover, this guarantees to have an approximation of the optimal solution of $\mathcal{P}$ before visiting all the nodes of Branch-and-Cut tree, for example when a CPU time limit has been reached.

The Branch-and-Cut approach has shown a great efficiency to solve various problems of combinatorial optimization that are considered difficult to solve, such as the Travelling Salesman Problem [6]. Note a good knowledge of the polyhedron associated with the problem, together with efficient separation algorithms (exacts as well as heuristics), might help to improve the effectiveness of this approach. Besides, the cutting plane method is efficient when the number of variables is polynomial. However, when the number of variables is large (for example exponential), further methods, as column generation are more likely to be used. In what follows, we briefly introduce the outline of this method.

1.2 Column generation and Branch-and-Price

Compact formulations of combinatorial optimization problems often provide a weak linear relaxation. Those problems require then further formulations, whose linear relaxation is closer to the convex hull of feasible solutions. Those reformulations may have a huge number of variables, so that one can not consider them explicitly in the model. we describe a method that suits well to such reformulation, that is the so-called *column generation method*.

1.2.1 Column generation procedure

The column generation method is used to solve linear programs with a huge number of variables only by considering a few number among these variables. This method was pioneered by Dantzig and Wolfe in 1960 [24] in order to solve problems that could not be handled efficiently because of their size (CPU time and memory consumption). Column generation is generally used either for problems whose structure is suitable for a *Dantzig-Wolfe decomposition*, or for problems with a large number of variables. Gilmore and Gomory [16, 17] used this method to solve a *cutting stock problem* belonging to the later class.

The overall idea of column generation is to solve a sequence of linear programs with a restricted number of variables (also referred to as columns). The algorithm starts by solving a linear program having a small number of variables, and such that a feasible solution for the original problem may be identified using this basis. At each iteration of the algorithm, we solve the so-called *pricing problem* whose objective is to identify the variables which must enter the current basis. These variables are characterized by a negative reduced cost. The reduced cost associated with a variable is computed using the dual variables associated with the constraints of the problem. We then solve the linear program obtained by adding the generated variables, and repeat the procedure until no variable with reduced cost can be identified by the pricing problem. In this case, the solution obtained from the last restricted program is optimal for the original model. The main step of column generation procedure is summarized in Algorithm 2.

The column generation method can be seen as the dual of the cutting plane method since it adds columns (variables) instead of rows (inequalities) in the linear program. Furthermore, the pricing problem may be NP-hard. One can then use heuristic procedures to solve it. For more details on column generation algorithms, the reader is referred to [64, 52, 25].

1.2.2 Branch-and-Price algorithm

The solution obtained by a column generation procedure may not be integer. Therefore, to solve an integer programming problem, the column generation method has to be integrated within a Branch-and-Bound framework. This is known a *Branch-and-Price algorithm*. Branch-and-Price is similar to Branch-and-Cut approach, except that procedure focuses on column generation rather than row generation. In fact, generating variables (pricing) and adding inequalities (cutting plane) are complementary operations to strengthen the linear relaxation of a integer programming formulation.

Algorithm 2: A column generation algorithm

Data : A linear program MP (Master Problem) with a huge number of variables
Output : optimal solution x^* of MP

1: Consider a linear program RMP (Restricted Master Problem) including only a small subset of variables of the MP;
2: Solve RMP and let x^* be an optimal solution;
3: Solve the pricing problem associated with the dual variables obtained by the resolution of the RMP;
4: **If** there exists a variable x with a negative reduced cost then;
5: add x to RMP.
6: go to 2.
7: **else**
8: x^* is optimal for MP.
9: return x^*.

The Branch-and-Price procedure works as follows. Each node of the Branch-and-Bound tree is solved by column generation, so that variables may be added to improve the linear relaxation of the current LP. The branching phase occurs when no columns price out to enter the basis and the solution of the linear program is not integer.

Branch-and-Price approaches have been widely used in the literature to solve large scale integer programming problems. The applications are in various fields, and even real life problems such as Cutting stock problem [65], Generalized Assignment Problem (GAP) [62], Airline Crew Scheduling [36], Multi-commodity Flow Problems [8], etc.

Note that, at each node of the Branch-and-Price tree, column generation may be combined with cutting plane approach, to tighten the LP relaxation of the problem. In this case, the algorithm is called *Branch-and-Cut-and-Price algorithm*. Such a method can be difficult to handle, since adding valid inequalities to the initial model may change the structure and complexity of the pricing problem. However, some successful applications of this algorithm can be found in the literature (see [61], [8] for instance).

1.2.3 Graph theory

In this section we will introduce some basic definitions and notations of graph theory that will be used throughout the chapters of this dissertation. For more details, we refer the reader to [63].

The graphs we consider are undirected, finite and loopless.

An undirected graph is denoted by $G = (V, E)$ where V is the node set and E is the edge set. If $e \in E$ is an edge with endnodes u and v, we also write uv to denote e. Given a set of nodes $Z \subset V$, we denote by $G \setminus Z$ the subgraph obtained from G by deleting the nodes in Z and all their incident edges. For $W \subseteq V$, we let $\overline{W} = V \setminus W$. Given W and W', two disjoint subsets of V, $[W, W']$ denotes the set of edges of G having one endnode in W and the other one in W'. If $W' = \overline{W}$, then $[W, W']$ is called a *cut* of G and denoted by $\delta_G(W)$. We will write $\delta(W)$ if the meaning is clear from the context. For $W \subset V$, we denote by $E(W)$ the set of edges of G having both endnodes in W and by $G[W]$ the subgraph induced by W. If $\pi = (V_1, ..., V_p)$, $p \geq 2$, is a partition of V, then we denote by $\delta_G(\pi)$ the set of edges having their endnodes in different sets. We may also write $\delta_G(V_1, ..., V_p)$ for $\delta_G(\pi)$. Note that for $W \subset V$, $\delta_G(W) = \delta_G(W, \overline{W})$.

Let $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$ be a subgraph of G. If $w(.)$ is a weight function which associates with each edge $e \in E$ the weight $w(e)$, then the total weight of G' is $w(E') = \sum_{e \in E'} w(e)$.

Given an undirected graph $G = (V, E)$, for all $F \subseteq E$, $V(F)$ will denote the set of nodes incident to the edges of F. For $W \subseteq V$, we denote by $E(W)$ the set of edges of G having both endnodes in W and $G[W]$ the graph induced by W, that is the graph $(W, E(W))$. Given an edge $e = uv \in E$, *contracting* e consists in deleting e, identifying the nodes u and v and in preserving all the adjacencies. Contracting a node subset W consists in indentifying 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 $\delta(V_1, ..., V_p)$.

A path P of an undirected graph G is an alternate sequence of nodes and edges $(u_1, e_1, u_2, e_2, ..., u_{q-1}, e_{q-1}, u_q)$ where $e_i \in [u_i, u_{i+1}]$ for $i = 1, ..., q-1$. We will denote a path P either by its node sequence $(u_1, ..., u_p)$ or its edge sequence $(e_1, ..., e_{q-1})$. The nodes u_1 and u_q are called the *endnodes* of P, while its other nodes are said to be *internal*. A path is *simple* if it does not contain the same node twice. In the sequel, we will always consider that the paths are simple. A path whose endnodes are s and t will be called an *st-path*. A *cycle* in G is a path whose endnodes coincide, that is $u_1 = u_q$. Also, a cycle is simple if it does not contain twice the same node, excepted u_1. We call a *chord* an edge between two non-adjacent nodes of a path. A *matching* of G is a set of pairwise nonadjacent edges. Given a fixed integer $L \geq 1$ and a pair of nodes $\{s, t\} \in V \times V$, an *L-st-path* in G is a path between s and t whose length is at most L, where the *length* is the number of edges of that path. The number of edges

of a path is also called *hops* and we also speak of *L-hop-constrained* paths for paths whose length is at most L.

An undirected graph is *connected* if for every pair of nodes (u, v) there is at least one path between u and v. A connected graph which have no cycle is called a *spanning tree*. A *connected component* of a graph G is a connected subgraph of G which is maximal, that is adding a node or an edge to that subgraph gives a non-connected graph.

Given an undirected graph $G = (V, E)$, two st-paths are *edge-disjoint* if they have no edge in common. They are *node-disjoint* if they have no internal node in common. A graph is said to be *k-edge-connected* if it contains at least k edge-disjoint st-paths for all pair of nodes $\{s, t\} \in V \times V$. It is *k-node-connected* if it contains at least k node-disjoint st-paths for all pair of nodes $\{s, t\} \in V \times V$. The largest integer k such that the graph is k-node-connected is the *node-connectivity* of the graph. We say that a graph is *Steiner* k-edge-connected (k-node-connected) if it is k-edge-connected (k-node-connected) relatively to a certain pair of privileged nodes. We ommit the qualificative Steiner when the required connectivity is for every pair of nodes of the graph. The privileged nodes are called *terminal nodes* while non-privileged ones are called *Steiner nodes*.

Given an undirected graph $G = (V, E)$, a *demand set* $D \subseteq V \times V$ is a subset of pairs of nodes, called *demands*. For a demand $\{s, t\} \in D$, s is the *source* of the demand and t is the destination of that demand. If several demands $\{s, t_1\}$, ..., $\{s, t_d\}$ have the same node s as source node, then these demands are *rooted* in s. A node involved in at least one demand is said to be *terminal*. A node which does not belong to any demand is called a *Steiner node*.

A *complete graph* is a graph in which there is an edge between each node and the others. A complete graph with n nodes is denoted by K_n. A *bipartite* graph $G = (V, E)$ is an undirected graph such that $V = V_1 \cup V_2$ with $V_1 \cap V_2 = \emptyset$ and for every pair of nodes $u, v \in V_1$ (resp. $u, v \in V_2$), $[u, v] = \emptyset$. A *complete bipartite graph* is a bipartite graph where there is an edge between each node of V_1 and the nodes of V_2. A bipartite complete graph is denoted $K_{m,n}$ where $m = |V_1|$ and $n = |V_2|$. An undirected graph is *outerplanar* when it can be drawn in the plane as a cycle with non crossing chords. 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;

ii) subdivision od an edge.

Observe that a graph is series-parallel (outerplanar) if and only if it is not contractible to K_4 (K_4 and $K_{3,2}$). Therefore, an outerplanar graph is also series-parallel.

1.3 State-of-the-art on network design problems

Survivable network design problems have been widely studied. The aim of the first studies was to produce heuristics and approximation algorithms for these problems. In the last decades, studies starts focusing on exact algorithms with, in particular, the use of the polyhedral approach.

In this section we present the previous works in the litterature related to survivable network design problems. We first present the general survivable network design problem, the related works and main results on this problem. Then we discuss two variants of the problem, the k-node-connected subgraph problem and the k-node-disjoint hop-constrained network design problem. The first one will be studied in Chapters 2, 3 and 4 and the second one in the Chapters 5 and 6.

1.3.1 The general survivable network design problem

A network can be represented by a graph, directed or undirected, where each node of the network corresponds to a node of the graph and a link between two nodes of the network is represented by an edge or an arc of the graph.

Consider an undirected graph $G = (V, E)$ representing a telecommunication network and $w(.)$ a weight function which associates the weight $w(e)$ with an edge $e \in E$. Each node $v \in V$ is associated with an integer, denoted by $r(v)$ and called *connectivity type* of v, which can be seen as the minimum number of edges connecting v to the rest of the network. The vector $(r(v) \mid v \in V)$ is the connectivity type vector associated with the nodes of G. We say that a subgraph $H = (U, F)$, $U \subseteq V$ and $F \subseteq E$, satisfies the *edge-connectivity* (resp. *node-connectivity*) *requirement* if for every pair of nodes $(s, t) \in V \times V$, there exist at least

$$r(s, t) = min\{r(s), r(t)\}$$

edge-disjoint (resp. node-disjoint) paths between s and t. This condition ensures that the traffic will still be routed between s and t when at most $r(s, t) - 1$ links, in the

case of edge-connectivity, and at most $r(s,t)-1$ nodes, in the case of node-connectivity, fail.

Let $r_{max} = max\{r(v) \mid u \in V\}$. When $r_{max} \leq 2$ we speak of *low connectivity requirement* and of *high connectivity requirement* when $r_{max} \geq 3$.

Grötschel, Monma and Stoer [45] introduced the *general survivable network design problem* which consists in finding a minimum weight subgraph of G which satisfies the connectivity requirement. This problem is NP-hard as it contains the Steiner tree problem as a special case ($r(u) \in \{0,1\}$ for all $u \in V$) which is known to be NP-hard [37].

Menger [60] showed the relation between the number of edge-disjoint paths and the cardinality of cuts in the graph G. This relation is given in the theorem below.

Theorem 1 [60] *Let $G = (V, E)$ be an undirected graph and s, t two nodes of G. Then there exist at least k edge-disjoint paths between s and t if and only if every st-cut of G contains at least k edges.*

We will denote the edge version (resp. nodes version) of the survivable network design problem by ESNDP (resp. NSNDP). By Theorem 1, the ESNDP can be described as a linear integer program. To this end we introduce some notations.

$$\begin{array}{rcll} r(W) & = & max\{r(u) \mid u \in W\} & \text{for all } W \subseteq V, \\ con(W) & = & max\{r(u,v) \mid u \in W, v \in \overline{W}\} & \\ & = & min\{r(W), r(\overline{W})\} & \text{for all } W \subset V, \emptyset \neq W \neq V. \end{array}$$

The ESNDP is equivalent to the following linear integer program

$$\begin{array}{lll} Minimize \sum_{e \in E} c(e)x(e) & & \\ x(\delta(W)) \geq con(W), & \text{for all } W \subsetneq V, W \neq \emptyset, & (1.1) \\ x(e) \geq 0, & e \in E, & (1.2) \\ x(e) \leq 1, & e \in E, & (1.3) \\ x(e) \in \{0,1\}. & & (1.4) \end{array}$$

Grötschel and Monma [44] study the polyhedral aspects of that model. They discuss the dimension of the associated polytope and the facial aspect of the basic inequalities. In [45], Grötschel et al. study further polyhedral aspects of that model. They devise cutting plane algorithms and give computational results. In [39], Goemans and Bertsimas give an approximation algorithm based for the ESNDP based on a new analysis of a well-known algorithm for the Steiner tree problem.

1.3.2 The k-edge(node)-connected subgraph problem

The kNCSP has applications in communication and transportation networks ([9, 45, 44, 46, 47]). The edge version of the problem has been widely studied in the literature ([9, 19, 12, 45, 44, 46, 47, 5]).

In [19] Chopra studied the polyhedron defined by the convex hull of k-edge-connected spanning subgraphs of a given graph G where multiple copies of an edge are allowed. A complete inequality description of the polytope when k is odd and G is an outer planar graph is given. He described a family of facet-defining inequalities of the polytope that have the same support graph and coefficients that depend on the connectivity.

In [5], Mahjoub study the problem of finding a two-edge connected spanning subgraph of minimum weight. This problem is closely related to the widely studied traveling salesman problem and has applications to the design of reliable communication and transportation networks. He discusses the polytope associated with the solutions to this problem, and shows that when the graph is series-parallel, the polytope is completely described by the trivial constraints and the so-called cut constraints. He also gives some classes of facet defining inequalities of this polytope when the graph is general, and later in [12], Didi Biha and Mahjoub give a complete description of the k-edge connected spanning subgraph polytope, for all $k \geq 0$ on series-parallel graphs.

In [9], Bendali et al. consider the k-edge connected subgraph problem from a polyhedral point of view. They introduce further classes of valid inequalities for the associated polytope and describe sufficient conditions for these inequalities to be facet defining. They also devise separation routines for these inequalities and discuss some reduction operations that can be used in a preprocessing phase for the separation. Using these results, they develop a Branch-and-Cut algorithm and present some computational results.

The kNCSP has been particulary considered for $k = 2$ (see [28, 57]). A little attention has been given for the high connectivity case where $k \geq 3$. The kNCSP has been

studied by Grötschel et al. ([45, 44, 46, 47]) within a more general survivability model. Grötschel et al. study the model from a polyhedral point of vue and propose cutting plane algorithms.

In [57], Mahjoub and Nocq discuss the linear relaxation of the 2NCSP(G). They describe some structral properties and characterize which they called extreme points of rank 1. They introduce an ordering on the fractional extreme points of the polytope and give a characterization of the minimal extreme points with respect to that ordering. This yields a polynomial method to separate a minimal extreme point of the polytope from the 2-node connected subgraph polytope. It also provides a new class of facet defining inequalities for this polytope.

1.3.3 The k-edge(node)-connected hop-constrained network design problem

The problem is to determine, given weights on the possible links of the network, and pairs of origin-destinations, a minimum weight network containing at least k edge (node) disjoint paths between each pair of origin-destination of length no more than L. This thesis deals with the node connectivity case of the problem.

Consider an undirected graph $G = (V, E)$ with weights $c(e)$, $e \in E$, on the edges, an integer $L \geq 2$, and a set of demands $D \subset V \times V$. Each demand is an ordered pair (s, t) of nodes, with $s \neq t$. Node s is called the *source* (or *origin*) of the demand and t its *destination*. The *k-Node-Disjoint Hop-Constrained Network Design Problem* (kNDHP for short) is to find a minimum weight subgraph of G containing at least k node-disjoint L-st-paths, that is, paths from s to t with at most L edges (also called hops), between each pair of nodes $(s, t) \in D$. The edge version of the problem has been widely studied in the literature. However, the kNDHP has been only considered for $k = 2$.

Node version with bounds

The edge version of the problem has been widely studied in the literature. However, the kNDHP has been only considered for $k = 2$. In [28], Diarrassouba et al. consider the kNDHP for $k = 2$. Here it is supposed that the two paths are node disjoint and each path does not exceed L edges for a fixed integer $L \geq 1$. They investigate the structure of the associated polytope and describe several classes of valid inequalities

when $L \leq 3$. Based on this, they devise a Branch-and-Cut algorithm. Huygens and Mahjoub [50] study the problem when $L = 4$ and $k = 2$. They show that the so-called cut and L-path-cut inequalities suffice for formulating the problem in this case. In [18], Chimani et al. consider $\{0, 1, 2\}$-Survivable Network Design problems with node-connectivity constraints. Given an edge-weighted graph and two customer sets $\mathscr{R}_1$ and $\mathscr{R}_2$, they look for a minimum cost subgraph that connects all customers, and guarantees two-node-connectivity for the $\mathscr{R}_2$ customers. They give a graph characterization of 2-node-connected graphs via orientation properties. Using this, they propose integer programming formulations based on directed graphs.

Edge version with bounds

The edge version of the problem has also been studied by several authors when $L = 2, 3$. In particular, in [51] Huygens et al. give a complete and minimal linear description of the corresponding polytope when $L = 2, 3$ and $|D| = 1$. In [49], Huygens et al. consider the problem when $|D| \geq 2$ and two edge disjoint paths are required for each demand. They show that the problem is strongly NP-hard even when the demands in D are rooted at some node s and the costs are unitary. However, if the graph is complete, they prove that the problem in this case can be solved in polynomial time. They give an integer programming formulation of the problem in the space of the design variables when $L = 2, 3$, and they study the associated polytope. Moreover, they describe several classes of valid inequalities along with necessary and/or sufficient conditions to be facet defining, and propose a Branch-and-Cut algorithm.

In [10] Bendali et al. consider the more general k edge-disjoint hop-constrained problem (kEDHP) when k edge disjoint paths are required. They discuss a Branch-and-Cut algorithm for the problem when $L = 2, 3$. Huygens and Mahjoub [50] study the kEDHP when $L = 4$ and $k = 2$. They introduce a new general class of valid inequalities. Using this, they give an integer programming formulation of the problem in the natural space of variables. In [21], Dahl considers the hop-constrained path problem, that is the problem of finding between two distinguished nodes s and t a minimum cost path with no more than L edges when L is fixed. He gives a complete description of the dominant of the associated polytope when $L \leq 3$ and a class of facet defining inequalities for $k \geq 4$. Dahl and Gouveia [23] consider the directed hop-constrained shortest path problem. They describe valid inequalities and characterize the associated polytope when $L = 2, 3$. A related problem is considered in Dahl et al. [22], the hop-constrained walk problem. The authors discuss the associated polytope in directed graphs when $L = 4$.

In [42], Gouveia and Leitner consider the Network Design Problem with Vulnerability Constraints. The solutions to the problem are subgraphs containing a path of length at most H_{st} for each commodity $\{s,t\}$ and a path of length at most H'_{st} between s and t after at most $k-1$ edge failures. They give characterizations of feasible solutions and propose integer programming formulations. In [41] Gouveia et al. consider the problem with bounded lengths in the context of an MPLS (Multi-Protocol Label Switching) network design model. They discuss two models involving one set of variables associated to each path between each pair of demand nodes (a standard network flow model with additional cardinality constraints and a model with hop-indexed variables) and a third model involving one single set of hop-indexed variables for each demand pair. They show that the aggregated more compact hop-indexed model produces the same linear programming bound as the multi-path hop-indexed model.

Extended formulations for the edge version with bounds

In [15] Botton et al. consider the hop-constrained survivable network design problem with reliable edges, i.e., edges that are not subject to failure. They study two variants, a static problem where the reliability of the edges is given, and an upgrading problem where edges can be upgraded to the reliable status at a given cost. They adapt for the two variants an extended formulation proposed in Botton et al. [14] for the case without reliable edges. They use Benders decomposition to accelerate the solution process. Their computational results indicate that these two variants appear to be more difficult to solve than the original problem (without reliable edges). In [58] Mahjoub et al. propose an extended formulation for the rooted case, when all the demands have a common vertex, called hop-level multicommodity flow formulation, inspired from the formulation given in [14]. The authors introduce the concept of solution level. To each solution of the problem, a partition of the node set into $L+2$ levels can be associated according to the distance to the root in the solution. Then they reduce the problem to a specific multicommodity flow problem in an auxiliary layered directed graph.

In Table 1.1 a summary of the previously studied hop constrained network design problems is presented.

Edge and node versions without bounds

The k-node-connected subgraph problem without bounds on the paths has been considered in the literature. In [57], Mahjoub and Nocq discuss structural properties of

Connectivity	Type of paths	Reference	Results
$k = 1$	-	**G. Dahl and L. Gouveia [23]**	Valid inequalities for the directed hop-constrained shortest path problem. Complete linear characterizations of the hop-constrained path polytope when $L = 2, 3$.
$k = 2$	Edge/Node disjoint	**Huygens and Mahjoub [50]**	IPF in the space of the design variables, for the node case when $L \leq 4$.
$k = 2$	Edge/Node disjoint	**Huygens et al. [49]**	IPF, valid inequalities and Branch-and-Cut algorithm for $L = 2, 3$.
$k \geq 1$	Edge disjoint	**Bendali et al. [10]**	Caracterization of the associated polytope for $L = 3$ and $\|D\| = 1$.
$k \geq 1$	Edge disjoint	**Diarrassouba et al. [27]**	Valid inequalities and Branch-and-Cut and Branch-and-Cut-and-Price algorithms for $L = 2, 3$.
$k = 2$	Node disjoint	**Diarrassouba et al. [28]**	Valid inequalities and Branch-and-Cut algorithm for $L = 3$

Table 1.1: State of the art of the hop-constrained survivable network design problem.

the 2-node-connected polytope (see also [7]). Grötschel et al. ([45, 44, 46, 47]) study the problem within a more general survivability model. In [44] Grötschel et al. introduce the concept of connectivity types. With each node $s \in V$ of G it is associated a nonnegative integer r_s, called the type of s. A subgraph of G is said to be survivable if for each pair of distinct nodes $s, t \in V$, the subgraph contains at least $r_{st} = min\{r_s, r_t\}$ edge (node) disjoint (s, t)-paths. Grötschel et al. study the problem from a polyhedral point of view, and propose cutting plane algorithms [44, 46, 47]. In [48], Kerivin et al. propose Branch & Cut algorithms for both versions of the $\{1, 2\}$ survivable network design problem. Here the type of each node is either 1 or 2. In [31], Mahjoub et al. consider the k-node-connected subgraph problem. They give valid inequalities and propose a Branch & Cut algorithm.

The uniform edge case without hop constraints has been widely investigated. The reader can be referred to [9, 12, 13, 45, 44, 46, 47] for more details.

In Table 1.2, we show the studied survivable network models with node versus edge connectivity.

As indicated in Table 1.2, the kNDHP has not been considered for $k \geq 3$.

Connectivity	Bound	Edge case results	Node case results
$k = 2$	$L = \infty$	ILP formulation, valid inequalities, separation, Branch & Cut, polytope characterization [7, 54, 45, 46, 47, 5]	ILP formulation, valid inequalities, separation, Branch & Cut [54, 57, 45, 46, 47]
$k \geq 3$	$L = \infty$	ILP formulation, valid inequalities, separation, Branch & Cut, polytope characterization [7, 54, 45, 47, 5]	ILP formulation, separation valid inequalities, Branch & Cut [9, 22, 45, 47]
$k = 2$	$L = 2, 3$	ILP formulation, valid inequalities, separation, Branch & Cut [49, 51]	ILP formulation, valid inequalities polyhedral study, Branch & Cut [10, 22, 45, 47]
$k = 2$	$L = 4$	ILP formulation, valid inequalities, separation, Branch & Cut [50, 49]	ILP formulation, valid inequalities Branch & Cut [50]
$k \geq 3$	$L = 2, 3$	ILP formulation, valid inequalities, separation, Branch & Cut, extended formulation [10, 15, 14, 21, 22, 23, 27]	Considered in this thesis

Table 1.2: Models of survivable networks with node versus edge connectivity.

Chapter 2

The k-node-connected subgraph problem

Contents

In this chapter we consider the k-Node-Connected Subgraph Problem (kNCSP). We investigate the kNCSP polytope and present several classes of valid inequalities. Then we discuss the conditions under which these inequalities define facets of the polytope. We also we consider the polytope associated with the linear relaxation of the problem and present some structural proporties as well as some reduction operations. Using these results we devise a Branch-and-Cut algorithm, and we give some experimental results. This work has led to a work published in the proceedings of the International Conference CIE45 [29] and an article published in Computers and Industrial Engineering [59].

2.1 Formulation

A graph $G = (V, E)$ is called *k-node* (resp. *k-edge*) *connected* ($k \geq 0$) if for every pair of nodes $i, j \in V$, there are at least k node-disjoint (resp. edge-disjoint) paths between i and j. Given a graph $G = (V, E)$ and a weight function c on E that associates with an edge $e \in E$ a weight $c(e) \in \mathbb{R}$, the *k-node-connected subgraph problem* (kNCSP for short) is to find a k-node-connected spanning subgraph $H = (V, F)$ of G such that $\sum_{e \in F} c(e)$ is minimum.

Let F be an edge subset of E, then the *incidence vector* of F, denoted by x^F, is the $0-1$ vector defined by

$$x^F(e) = \begin{cases} 1 & \text{if } e \in F \\ 0 & \text{otherwise.} \end{cases}$$

Let $F \subseteq E$ be an edge subset of G. Then F induces a solution of the kNCSP for G, that is, the subgraph of G induced by F is k-node-connected, if x^F satisfies the following inequalities

$$x(e) \geq 0, \qquad e \in E, \tag{2.1}$$

$$x(e) \leq 1, \qquad e \in E, \tag{2.2}$$

$$x(\delta_G(W)) \geq k, \qquad \text{for all } W \subsetneq V \text{ with } W \neq \emptyset, \tag{2.3}$$

$$x(\delta_{G \setminus Z}(W)) \geq k - |Z|, \qquad \text{for all } Z \subseteq V \text{ such that } 1 \leq |Z| \leq k-1, \text{ and all } W \subsetneq V \setminus Z \text{ with } W \neq \emptyset. \tag{2.4}$$

Conversely, any integer solution of the system above is the incidence vector of the edge set of a k-node-connected subgraph of G. Hence, the kNCSP is equivalent to

$$\min\{cx \mid x \text{ satisfies } (2.1) - (2.4) \text{ and } x \in \mathbb{Z}_+^E\}. \tag{2.5}$$

Constraints (2.3) and (2.4) are called *cut* and *node-cut* inequalities, respectively. The convex hull of all integer solutions of (2.1)-(2.4), denoted by kNCSP(G), will be called kNCSP(G) *the k-node-connected subgraph problem polytope.*

We will also denote by $P(G, k)$ the polytope described by constraints $(2.1) - (2.4)$.

In what follows we give an alternative formulation for the problem. This consists in restricting the node-cut inequalities (2.4) to the node sets $Z \subset V$ such that $|Z| = k-1$. We hence consider the following set of inequalities

$$x(\delta_{G\setminus Z}(W)) \geq 1, \qquad \begin{array}{l} \emptyset \neq Z \subseteq V, |Z| = k-1, \\ \emptyset \neq W \subset V \setminus Z. \end{array} \tag{2.6}$$

Theorem 2 *The kNCSP is equivalent to*

$$\min\{cx \mid x \text{ satisfies } (2.1)-(2.3),\ (2.6) \text{ and } x \in \mathbb{Z}_+^E\}. \tag{2.7}$$

Proof. It suffices to show that any integer solution $\overline{x}$ of (2.1)-(2.3),(2.6) also satisfies (2.4). For this we will show that if $\overline{x}$ satisfies all inequalities $x(\delta_{G\setminus Z}(W)) \geq k - |Z|$ with $|Z| = t$ for some $t \in \{k-1, ..., 2\}$, then $\overline{x}$ satisfies $x(\delta_{G\setminus Z'}(W')) \geq k - |Z'|$ for all $Z' \subset V$ with $|Z'| = t-1$ and $W' \subset V \setminus Z'$. Indeed, first note that either $|V\setminus(W' \cup Z')| \geq 2$ or $|W'| \geq 2$ or both. In fact, if $|V\setminus(W' \cup Z')| = |W'| = 1$, then $|Z'| = n-2$ $(= t-1)$. But this implies that $t = n-1$, and, as $t \leq k-1$, it follows that $k \geq n$, which is impossible. In what follows we suppose, w.l.o.g., that $|V\setminus(W' \cup Z')| \geq 2$. We claim that there is at least one node, say u, in $V\setminus(W' \cup Z')$ such that $\overline{x}([u, W']) \geq 1$. In fact, let $u_0 \in V \setminus (W' \cup Z')$. Let $Z = Z' \cup \{u_0\}$. By our assumption, $\overline{x}(\delta_{G\setminus Z}(W')) \geq k - |Z| = k - t$. As $t \leq k-1$, it follows that $\overline{x}(\delta_{G\setminus Z}(W')) \geq 1$. Therefore there is a node u in $V \setminus (W' \cup Z' \cup \{u\})$ such that $\overline{x}([u, W']) \geq 1$.

Now let $u \in V\setminus(W' \cup Z')$ such that $\overline{x}([u, W']) \geq 1$, and let $Z^* = Z' \cup \{u\}$. We have $|Z^*| = t$. Again, by our assumption, we have that $\overline{x}(\delta_{G\setminus Z^*}(W')) = \overline{x}(\delta_{G\setminus Z'}(W')) - \overline{x}([u, W']) \geq k - |Z^*| = k - t$. As $\overline{x}([u, W']) \geq 1$, it then follows that $\overline{x}(\delta_{G\setminus Z'}(W')) = \overline{x}(\delta_{G\setminus Z^*}(W)) + \overline{x}([u, W']) \geq k - t + 1 = k - |Z'|$. □

As before, we will denote by $Q(G, k)$ the polytope associated with the linear relaxation of (2.7). Clearly, $P(G, k) \subseteq Q(G, k)$. Moreover, the two polytopes may be different, that is $P(G, k) \neq Q(G, k)$, for some graph G and connectivity k. For example, consider the graph and the solution of Figure 2.1 for $k = 3$. The solution satisfies the cut inequalities and the node-cut inequalities with $|Z| = k - 1 = 2$, and violates a node-cut inequality with $|Z| = k - 2 = 1$. Indeed, for $Z = \{v_7\}$ and $W = \{v_1, v_2, v_3\}$, $x(\delta_{G\setminus Z}(W)) < 2$. Thus, formulation (2.5) may produce a better linear relaxation than (2.7). We will hence consider formulation (2.5) for solving the kNCSP.

In the next sections, we investigate the polytope kNCSP(G) and describe some valid inequalities.

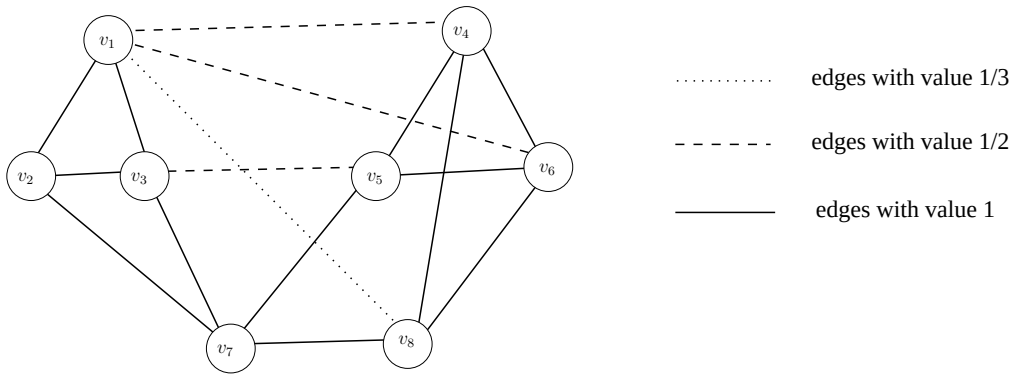

Figure 2.1: A solution of $Q(G,k) \setminus P(G,k)$ for $k = 3$.

2.2 Dimension and Valid inequalities

In this section, we will discuss the polytope kNCSP(G). We will establish its dimension and describe some classes of valid inequalities.

2.2.1 Dimension

Let $G = (V, E)$ be a graph. An edge e is said to be *essential* if the solutions set of kNCSP($G\backslash e$) is empty. Let E^* be the set of essential edges of kNCSP. We have the following result.

Theorem 3 *dim(kNCSP(G))*$= |E| - |E^*|$.

Proof. Let $e \in E^*$. Then, $x(e) = 1$ for every solution x of kNCSP(G). Then $dim(k\text{NCSP}(G)) \leq |E| - |E^*|$. Now, observe that the edge sets $S_e = E\backslash\{e\}$, $e \in E\backslash E^*$, and E form $|E| - |E^*| + 1$ solutions of the kNCSP. Moreover, the incidence vectors of these solutions are affinely independent. Therefore, $dim(k\text{NCSP}(G)) \geq |E| - |E^*|$. Thus, the result follows. □

Corollary 1 *kNCSP(G) is full-dimensional if and only if G is* $(k+1)$*-node-connected.*

Now we describe some classes of valid inequalities for kNCSP(G). One can easily see that any solution of the kNCSP on G is also solution of the kECSP on G. Thus, any valid inequality for the kECSP polytope on G is also valid for kNCSP(G).

In the following, we introduce a notation that will be used throughout the remainder of the chapter. 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 by contracting the sets V_i, $i = 1, ..., p$. We will denote by $\delta_G(V_1, ..., V_p)$ the set of edges of G_π, that is, the edges that have their endnodes in different elements of π.

2.2.2 Node-partition inequalities

In [45], Grötschel et al. introduce a class of valid inequalities for a more general version of the kNCSP as follows. Consider a subset $Z \subset V$, such that $|Z| \leq k-1$, and let $V_1, ..., V_p$, $p \geq 2$ be a partition of $V \backslash Z$. Then the inequality

$$x(\delta_{G\backslash Z}(V_1, ..., V_p)) \geq \begin{cases} \left\lceil \frac{p(k-|Z|)}{2} \right\rceil & \text{if} \quad |Z| \leq k-2, \\ p-1 & \text{if} \quad |Z| = k-1, \end{cases} \tag{2.8}$$

is valid for the kNCSP(G). Inequalities of type (2.8) are called *node-partition inequalities.*

2.2.3 SP-node-partition inequalities

Now we introduce a class of inequalities called SP-node-partition inequalities, which generalize the so-called SP-partition inequalities introduced by Didi Biha and Mahjoub [12] for the kECSP(G), where kECSP(G) is the convex hull of the solutions of the k-edge-connected subgraph problem. These latter inequalities are defined as follows. Let $\pi = (V_1, ..., V_p)$ be a partition of V such that the graph G_π is series-parallel. Recall that a graph is series-parallel if it is not contractible to K_4, the complete graph on four nodes. The SP-partition inequality associated with π is given by

$$x(\delta_G(V_1, ..., V_p)) \geq \left\lceil \frac{k}{2} \right\rceil p - 1. \tag{2.9}$$

Didi Biha and Mahjoub [12] showed that these inequalities are valid for the kECSP(G), for every $k \geq 1$.

For the kNCSP, we introduce a similar type of inequalities. Let $Z \subset V$ such that $|Z| \leq k-1$ and $k - |Z|$ is odd, and consider a partition $\pi = (V_1, ..., V_p)$ of $V \backslash Z$ such that $(G \backslash Z)_\pi$ is series-parallel. The *SP-node-partition inequality* associated with π is

$$x(\delta_{G\backslash Z}(\pi) \geq \left\lceil \frac{k-|Z|}{2} \right\rceil p - 1. \tag{2.10}$$

Theorem 4 *The SP-node-partition inequalities (2.10) are valid for kNCSP(G).*

Proof.

Let $x \in k\text{NCSP}(G)$ and consider x' the restriction of x on $G\backslash Z$. As $x' \in (k-|Z|)\text{ECSP}(G\backslash Z)$, and the SP-partition inequalities (2.9) are valid for $(k-|Z|)\text{ECSP}(G\backslash Z)$, we have

$$x(\delta_{G\backslash Z}(V_1, ..., V_p)) = x'(\delta_{G\backslash Z}(V_1, ..., V_p)) \geq \left\lceil \frac{k-|Z|}{2} \right\rceil p - 1,$$

which proves the result. □

Chopra [19] (see also Didi Biha and Mahjoub [12]) described a lifting procedure for inequalities (2.10). This can be easily extended to the SP-node-partition inequalities. Let $G = (V, E)$ be a graph and $k \geq 3$ an odd integer. Consider the graph $G' = (V, E \cup T)$ obtained from G by adding an edge set T. Let $Z \subset V$ and $\pi = (V_1, ..., V_p)$ be a partition of $V\backslash Z$ such that $(G\backslash Z)_\pi$ is series-parallel. Then the *lifted SP-node-partition inequality* induced by π is

$$x(\delta_{G\backslash Z}(V_1, ..., V_p)) + \sum_{e \in T \cap \delta_{G'}(V_1, ..., V_p)} a(e)x(e) \geq \left\lceil \frac{k-|Z|}{2} \right\rceil p - 1, \tag{2.11}$$

where $a(e)$ is the length, that is to say the number of edges, of the shortest path in $(G\backslash Z)_\pi$ between the endnodes of e, for all $e \in T \cap \delta_{G'}(V_1, ..., V_p)$.

Chopra [19] shows that inequalities (2.11) are valid for $k\text{ECSP}(G)$. Therefore they are also valid for $k\text{NCSP}(G)$.

2.2.4 F-node-partition inequalities

Let $G = (V, E)$ be a graph and Z a node subset of V. Let $\pi = (V_0, V_1, ..., V_p)$ be a partition of $V\backslash Z$ and F an edge subset of $\delta_{G\backslash Z}(V_0)$. Let $Z_i \subset Z$ be the set of nodes of

Z adjacent to the nodes in V_i, $i = 1, ..., p$. Suppose that $|Z_i| \leq k-1$, for $i = 1, ..., p$, and $Z = \bigcup_{i=1,...,p} Z_i$. The following inequality

$$x(\delta_{G\setminus Z}(\pi)\setminus F) \geq \left\lceil \frac{\sum_{i=1}^{p}(k - |Z_i|) - |F|}{2} \right\rceil \tag{2.12}$$

is called an *F-node-partition inequality.*

Theorem 5 *F-node partition inequalities are valid for the kNCSP(G).*

Proof. Consider the following valid inequalities

$$\begin{array}{rcll} x(\delta_{G\setminus Z_i}(V_i)) & \geq & k - |Z_i|, & \text{for all } i = 1, ..., p, \\ -x(e) & \geq & -1, & \text{for all } e \in F, \\ x(e) & \geq & 0, & \text{for all } e \in \delta_{G\setminus Z}(V_0)\setminus F. \end{array}$$

By summing these inequalities, we obtain

$$2x(\delta_{G\setminus Z}(\pi)\setminus F) \geq \sum_{i=1}^{p}(k - |Z_i|) - |F|.$$

By dividing by 2 and rounding up the right hand, we get inequality (2.12). □

2.3 Facial aspect

In this section, we discuss the facial aspect of the kNCSP polytope. Namely, we investigate the conditions under which the inequalities presented in the previous section define facets of kNCSP(G). In the following we assume that G is $(k+1)$-node-connected. By Corollary 1, kNCSP(G) is then full-dimensional.

In [44], Grötschel et al. characterize when the trivial inequalities define facets.

Theorem 6 [44]

1) Inequality (2.1) *defines a facet for kNCSP(G) if and only if e does not belong to a cut* $\delta_{G\setminus Z}(W)$ *for some* $Z \subset V$ *containing exactly* $k+1-|Z|$ *edges.*

2) Inequalities (2.2) *define facets for kNCSP(G) for every* $e \in E$.

The next theorem deals with conditions for the cut inequalities to define facets. Before, we give the following remark that will be helpful for proving the results below.

Remark 2.1 *Let* W *and* $\overline{W}$ *be a partition of* G *such that* $|W| \geq k$, $|\overline{W}| \geq k+1$ *and* $G[W]$ *and* $G[\overline{W}]$ *are both k-node-connected. Let* $\{e_1, ..., e_k\}$ *be edges of* $\delta_G(W)$ *forming a matching of* G. *Let* $S = E(W) \cup E(\overline{W}) \cup \{e_1, ..., e_k\}$. *Then* S *is a solution of kNCSP(G).*

Theorem 7 *The cut inequality* (2.3), *induced by a node set* $W \subset V$, *defines a facet for kNCSP(G) if the following hold.*

i) $G[W]$ *and* $G[\overline{W}]$ *are* $(k+1)$*-node-connected.*

ii) A maximum cardinality matching M *in* $\delta_G(W)$ *contains at least* $k+1$ *edges.*

Proof.

Let us denote by $ax \geq \alpha$ the cut inequality induced by W, and let $\mathscr{F} = \{x \in kNCSP(G) \mid ax = \alpha\}$. Suppose there exists a defining facet inequality $bx \geq \beta$ such that $\mathscr{F} \subseteq F = \{x \in kNCSP(G) \mid bx = \beta\}$. We will prove that there is a scalar ρ such that $b = \rho a$. By ii) there exists a matching $M = \{e_1, ..., e_p\}$, $p \geq k+1$, in $\delta_G(W)$ of p edges such that $e_i = u_i v_i$, $i = 1, ..., p$, with $u_i \in W$ and $v_i \in \overline{W}$. Let $U_1 = \{u_1, ..., u_p\}$ and $V_1 = \{v_1, ..., v_p\}$. Let $T_1 = E(W) \cup E(\overline{W}) \cup \{e_1, ..., e_k\}$. As by i) $G[W]$ and $G[\overline{W}]$ are $(k+1)$-node-connected, by Remark 2.1, T_1 is a solution of kNCSP(G). We will show in what follows that the coefficients b_e are equal for all $e \in \delta_G(W)$. First we show that $b_{e_i} = b_{e_j}$ for $i, j \in \{1, ..., p\}$. Let $T_2 = (T_1 \setminus \{e_1\}) \cup \{e_{k+1}\}$. Clearly T_2 is a solution of kNCSP. As $x^{T_1}, x^{T_2} \in \mathscr{F}$, we have that $bx^{T_1} = bx^{T_2} = \beta$. This implies that $b_{e_1} = b_{e_{k+1}} = \rho$ for some $\rho \in \mathbb{R}$. By symmetry, it follows that $b_{e_i} = \rho$ for all $i = 1, ..., p$. As M has a maximum cardinality, any edge $e \in \delta(W) \setminus M$ is adjacent to M. Consider an edge $f = u_i v \in [U_1, \overline{W} \setminus V_1]$, $i \in \{1, ..., p\}$. Let $T_3 = (T_1 \setminus \{e_i\}) \cup \{f\}$. As $x^{T_1}, x^{T_3} \in \mathscr{F} \subset F$, we have that $bx^{T_1} = bx^{T_3} = \beta$. This yields $b_f = b_{e_i} = \rho$. Thus $b_f = \rho$ for all $f \in [U_1, \overline{W} \setminus V_1]$. By symmetry we also have $b_f = \rho$ for all $f \in [V_1, W \setminus U_1]$.

Finally consider an edge $h = u_i v_j$, $i, j \in \{1, ..., p\}$, with $i \neq j$. W.l.o.g., we suppose $i, j \leq k$. Consider the subset $T_4 = (T_1 \backslash \{e_i, e_j\}) \cup \{h, e_{k+1}\}$. We have that T_4 is a solution of kNCSP(G), and $x^{T_4} \in \mathscr{F} \subseteq F$. Which implies that $b_{e_i} + b_{e_j} = b_h + b_{e_{k+1}}$. As $b_{e_i} = b_{e_j} = b_{e_{k+1}} = \rho$, it follows that $b_h = \rho$. Thus we obtain that $b_e = \rho$ for all $e \in \delta_G(W)$.
We will now show that $b_e = 0$ for all $e \in E \backslash \delta_G(W)$. As $G[W]$ and $G[\overline{W}]$ are $(k+1)$-node-connected, we have that $T_5 = T_1 \backslash \{e\}$ induces a k-node-connected graph for all edge $e \in E(W) \cup E(\overline{W})$. Moreover $x^{T_5} \in \mathscr{F} \subseteq F$. Hence $b_e = 0$. Consequently, we have that $b_e = \rho$ for all $e \in \delta_G(W)$, and $b_e = 0$ for all $e \in E \backslash \delta_G(W)$. Thus $b = \rho a$. □

Corollary 2 *If the graph G is complete, the cut inequality (2.3) induced by $W \subset V$ is facet-defining for kNCSP(G) if $|W| \geq k + 2$ and $|\overline{W}| \geq k + 2$.*

The following theorems give necessary conditions and sufficient conditions for the node-cut inequalities to be facet-defining.

Theorem 8 *The node-cut inequality (2.4), induced by a node-cut $\delta_{G \backslash Z}(W)$ for some node sets W and Z, defines a facet for kNCSP(G) only if $|[W, Z]| \geq |Z| + 1$ and $|[V \setminus (W \cup Z), Z]| \geq |Z| + 1$.*

Proof. Suppose for instance that $|[W, Z]| < |Z| + 1$, the case where $|[V \setminus (W \cup Z), Z]| < |Z| + 1$ is similar. Thus, if $|[W, Z]| < |Z| + 1$, then for any solution $x \in k$NCSP(G) we have that $-x([W, Z]) \geq -|Z|$, and $x(\delta_G(W)) \geq k$. Hence we obtain that $x(\delta_{G \backslash Z}(W)) = x(\delta_G(W)) - x([W, Z]) \geq k - |Z|$. In consequence, $x(\delta_{G \backslash Z}(W)) \geq k - |Z|$ is redundant with respect to the cut and trivial inequalities, and hence cannot define a facet. □

Theorem 9 *The node-cut inequality (2.4) defines a facet for kNCSP(G) if the following hold.*

i) $G[W]$ and $G[\overline{W}]$ are $(k+1)$-node-connected.

ii) A maximum cardinality matching C in $\delta_G(W)$ contains at least $k + 1$ edges such that $|C \cap [Z, W]| = |Z|$ and there exists a node in W which is not incident to the matching C and it is adjacent to all the nodes of Z.

Proof. Let us denote by $ax \geq \alpha$ the cut inequality induced by W, and let $\mathscr{F} = \{x \in kNCSP(G) \mid ax = \alpha\}$. Suppose there exists a defining facet inequality $bx \geq \beta$ such that $\mathscr{F} \subseteq F = \{x \in kNCSP(G) \mid bx = \beta\}$. We will prove that there is a scalar ρ such that $b = \rho a$. By ii) there exists a matching $C = \{e_1, ..., e_p\}$, $p \geq k+1$, in $\delta_G(W)$, such that $e_i = u_i v_i$, $i = 1, ..., p$, $u_i \in W$ and $v_i \in \overline{W}$, and $e_{k-t+j} \in [W, Z]$, $j = 1, ..., t$, with $|Z| = t$. Let $U_1 = \{u_1, ..., u_p\}$ and $V_1 = \{v_1, ..., v_p\}$. And let $T_1 = E(W) \cup E(\overline{W}) \cup \{e_1, ..., e_k\} \cup [W, Z]$. As by i) $G[W]$ and $G[\overline{W}]$ are $(k+1)$-node-connected, by Remark 2.1, T_1 is a solution of kNCSP(G). Hence $x^{T_1} \in \mathscr{F}$. Let $T_2 = (T_1 \setminus \{e_1\}) \cup \{e_{k+1}\}$ (Recall that $p \geq k+1$). Clearly, T_2 is a solution of kNCSP. As $x^{T_1}, x^{T_2} \in \mathscr{F}$, we have that $bx^{T_1} = bx^{T_2} = \beta$, implying that $b_{e_1} = b_{e_{k+1}} = \rho$ for some $\rho \in \mathbb{R}$. By symmetry, we obtain that $b_{e_i} = \rho$ for all $e_i \in C \setminus [W, Z]$. As C has a maximum cardinality, any edge $e \in \delta_{G \setminus Z}(W) \setminus C$ is adjacent to C. Consider an edge $f = u_i v \in [U_1, V \setminus (W \cup Z \cup V_1)]$. W.l.o.g., we suppose $i \in \{1, ..., k-t\}$. Let $T_3 = (T_1 \setminus \{e_i\}) \cup \{f\}$. Set T_3 is a solution of kNCSP(G). Moreover $x^{T_3} \in \mathscr{F} \subseteq F$. Hence $bx^{T_1} = bx^{T_3} = \beta$, implying that $b_f = b_{e_i} = \rho$. Thus $b_f = \rho$ for all $f \in [U_1, V \setminus (W \cup Z \cup V_1)]$. By symmetry, we also have that $b_f = \rho$, for all $f \in [V_1 \setminus Z, W \setminus U_1]$. Now let $h = u_i v_j$, $i \neq j$, $i, j \in \{1, ..., p\} \setminus \{k-t+1, ..., k\}$. Consider $T_4 = (T_1 \setminus \{e_i, e_j\}) \cup \{h, e_{k+1}\}$. Set T_4 is a solution of kNCSP(G). Moreover $x^{T_4} \in \mathscr{F} \subseteq F$. Hence $b_{e_i} + b_{e_j} = b_h + b_{e_{k+1}}$. As $b_{e_i} = b_{e_j} = b_{e_{k+1}} = \rho$, this implies that $b_h = \rho$. Therefore we obtain that $b_e = \rho$ for all $e \in \delta_{G \setminus Z}(W)$. Now consider an edge $e \in E(W)$, and let $T_5 = T_1 \setminus \{e\}$. As $G[W]$ is $(k+1)$-node-connected, $G[W] \setminus \{e\}$ is k-node-connected, and hence T_5 is a solution of kNCSP. Thus $x^{T_5} \in \mathscr{F} \subseteq F$, and $bx^{T_1} = bx^{T_5} = bx^{T_1} - b_e$. Which implies that $b_e = 0$. Therefore $b_e = 0$ for all $e \in E(W)$. Similarly, we have that $b_e = 0$ for all $e \in E(\overline{W})$. Now let $e \in [W, Z]$. Let $T_6 = T_1 \setminus \{e\}$. By ii), T_6 contains a matching of at least k edges in $\delta(W)$. Hence T_6 is a solution of kNCSP, which implies that $b_e = 0$. Consequently, we obtain that $b_e = \rho$ for all $e \in \delta_{G \setminus Z}(W)$ and $b_e = 0$ for all $e \in E \setminus \delta_{G \setminus Z}(W)$. Thus, $b = \rho a$. Which ends the proof of the theorem. □

Corollary 3 *If the graph G is complete, then the node-cut inequalities (2.4) are facet-defining for kNCSP(G) if $|W| \geq k+2$ and $|\overline{W}| \geq k+2$.*

Now, we discuss sufficient conditions for the F-node-partition and SP-node-partition inequalities to define facets of kNCSP(G).

Theorem 10 *Let $G = (V, E)$ be a graph and an integer $k \geq 2$. Let $Z \subset V$. Let $Z_i \subset Z$, with $|Z_i| \leq k-2$, for $i = 1, ..., p$, and $\pi = (V_0, V_1, ..., V_p)$ be a partition of $V \setminus Z$ where p is odd. Suppose that the following hold.*

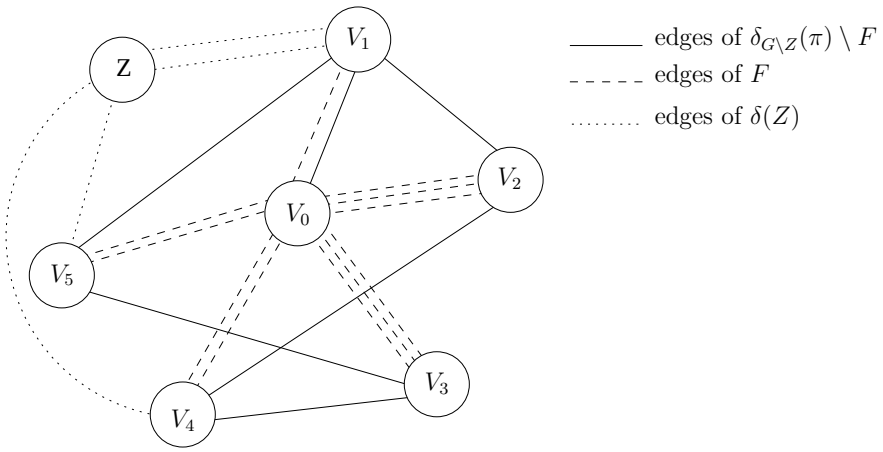

Figure 2.2: A F-node-partition configuration with $k = 4$

i) *$G[Z]$ is a complete graph.*

ii) *$G[V_i]$ is $(k+1)$-node-connected, for $i = 0, 1, ..., p$.*

iii) *For $i = 1, ..., p$, there exists a subset S_i of $k+1$ edges of $\delta(V_i)$ such that*

1) $|S_i \cap [Z_i, V_i]| = |Z_i|$ and covering all the nodes of Z_i,
2) $|S_i \cap [V_0, V_i]| = k - |Z_i| - 1$ and covering $k - |Z_i| - 1$ nodes of V_0,
3) $|S_i \cap [V_i, V_{i-1}]| = |S_i \cap [V_i, V_{i+1}]| = 1$,

where the indices are taken modulo p.

Moreover, if $|V_i| \geq 2$, S_i must cover at least $k+1$ nodes of V_i.

iv) *$[Z, V_0]$ contains a set of $k+1$ edges covering $k+1$ nodes of V_0 and $min(|Z|, k+1)$ nodes of Z.*

v) *For $i = 1, ..., p$, $[V_0, V_i \cup V_{i+1}]$ contains a set $R_i \subseteq S_i \cup S_{i+1}$ of $k - |Z_i| + 1$ edges covering $k - |Z_i| + 1$ nodes of V_0.*

Let $F_i = S_i \cap [V_0, V_i]$, for $i = 1, ..., p$, and let $F = \bigcup_{i=1}^{p} F_i$. Then the F-node-partition inequality (2.12), induced by π and F, defines a facet of kNCSP(G).

(see figure 2.2 for an illustration for $k = 4$)

Proof.

Remark that under these conditions we can easily see that G is $(k+1)$-node-connected, thus kNCSP(G) is full dimensional. Let us denote the F-node-partition inequality by $ax \geq \alpha$ and let $\mathscr{F} = \{x \in kNCSP(G) \mid ax = \alpha\}$. Clearly, $\mathscr{F}$ is a proper face of kNCSP(G). Now suppose that there exists a facet defining inequality $bx \geq \beta$ of kNCSP(G) such that $\mathscr{F} \subseteq \mathscr{F}' = \{x \in kNCSP(G) \mid bx = \beta\}$. We will show that $b = \rho a$ for some scalar $\rho \in \mathbb{R}$.
For this, first remark that the right hand side of the inequality (2.12) here is $\left\lceil \frac{p}{2} \right\rceil$. Let E_0 be the set of edges in $E \backslash F$ having both endnodes in the same element of π. Let $\Gamma = E_0 \cup F \cup E(Z) \cup \delta(Z)$.

Let e_l be an edge of $S_l \cap [V_l, V_{l+1}]$, $l = 1, ..., p$. For $l \in \{1, ..., p\}$ consider the edge set

$$T_l \quad = \quad \Gamma \cup \{e_{l+2r}, r = 0, ..., \frac{p-1}{2}\},$$

where the indices are taken modulo p. Observe that $x^{T_l}(\delta_{G \backslash Z}(\pi) \setminus F) = \frac{p+1}{2}$. Moreover, we have the following.

Claim 1 *T_l induces a k-node-connected subgraph of G.*

Proof. Let G_l be the subgraph of G induced by T_l. First, we give the following remarks.

a) $|\delta_{G_l}(V_j)| = k$ for $j \in \{1, ..., p\} \backslash \{l\}$ and $|\delta_{G_l}(V_l)| = k + 1$.

b) $F_i \neq \emptyset$ since $|Z_i| \leq k - 2$, for $i \in \{1, ..., p\}$, and the graph obtained from G_l by removing subsets from $\{Z, V_1, ..., V_p\}$ is connected,

c) The graph G_l^* obtained from G_l by contracting the sets V_0, V_1, ..., V_p, Z, and replacing the multiple edges by a single edge, and deleting the edges between V_i and V_j, $i \neq j$, $i, j = 1, ..., p$, is connected.

Let $Z' \subset V$ with $|Z'| = k - 1$. We will show that the graph $G_l \backslash Z'$ is connected.

Case 1. $Z' \subset Z$ or $Z' \subset V_i$, for some $i \in \{1, ..., p\}$. Suppose that $Z' \subset Z$. If $|Z| = |Z'| = k - 1$, then by the Remark b) above, $G_l \backslash Z'$ is connected. So suppose $|Z| \geq k$. As $|Z'| = k - 1$ and $G[Z]$ is complete, the subgraph induced by $Z \backslash Z'$ is connected. Moreover, by Condition iv), there exists at least one edge connecting $Z \backslash Z'$ to V_0. Since $G_l \backslash Z$ is connected, we obtain that $G_l \backslash Z'$ is also connected.

Now suppose $Z' \subset V_i$ for some $i \in \{1, ..., p\}$. As $G[V_i]$ is $(k+1)$-node-connected and $|Z'| = k-1$, $G[V_i \setminus Z']$ is connected. Therefore, using Condition iii), the proof can be done along the same line.

Case 2. $Z' \subset V_0$. As $|Z'| = k-1$, by Condition iv), it follows that $[Z, V_0 \backslash Z']_{G_l} \neq \emptyset$. We distinguish two cases. Suppose first that for every $s \in \{1, ..., p\}$ such that $[V_s, V_{s+1}]_{G_l} \neq \emptyset$, at least one of the sets V_s and V_{s+1} is adjacent to Z in G_l. Then the graph obtained from G_l by removing V_0 is connected. Moreover, since $G[V_0]$ is $(k+1)$-node-connected, we have that $G[V_0 \backslash Z']$ is connected. Therefore $G_l \backslash Z'$ is connected.

If this is not the case, then there is $s \in \{1, ..., p\}$ such that $[V_s, Z]_{G_l} = \emptyset = [V_{s+1}, Z]_{G_l}$ and $[V_s, V_{s+1}]_{G_l} \neq \emptyset$. We then have $Z_s = Z_{s+1} = \emptyset$. Moreover, by Condition v), it follows that there is a set of $k+1$ edges between V_0 and $V_s \cup V_{s+1}$ that cover $k+1$ nodes of V_0. As $|Z'| = k-1$, at least one edge remains linking $V_0 \backslash Z'$ to $V_s \cup V_{s+1}$. Thus for $j = 1, ..., p$, if $[V_j, V_{j+1}]_{G_l} \neq \emptyset$, then at least one of the sets $[V_0 \backslash Z', V_j \cup V_{j+1}]_{G_l}$ and $[Z, V_j \cup V_{j+1}]_{G_l}$ is not empty. As $G[V_0]$ is $(k+1)$-node-connected and $|Z'| = k-1$, $G[V_0 \backslash Z']$ is connected, and hence $G_l \backslash Z'$ is connected.

Case 3. $Z' \subseteq \bigcup\limits_{i \in I} V_i$, where $I \subset \{1, ..., p\}$. Note that $|I| \leq k-1$. Let $I' = \{i \in I \mid |V_i| = 1\}$. First note that, by Remark b) above, the graph $G_l \backslash \bigcup\limits_{i \in I} V_i$ is connected. Also note that as $|Z'| = k-1$, and Z' is not contained in a single set, we have $|Z' \cap V_i| \leq k-2$ for $i \in I \backslash I'$. Since $G[V_i]$ is $(k+1)$-node-connected, it follows that $G[V_i \backslash Z']$ is connected for $i \in I \backslash I'$. Also by construction, we have $|[V_i, V_0 \cup Z]_{G_l}| \geq k-1$ for $i \in I \backslash I'$. Moreover by Condition iii), $[V_i, V_0 \cup Z]_{G_l}$ covers at least $k-1$ different nodes in V_i, for $i \in I \backslash I'$. So, if no more than $k-2$ nodes are removed from V_i, at least one edge remains connecting $V_i \backslash Z'$ to $Z \cup V_0$ for $i \in I \backslash I'$. Therefore $G_l \backslash Z'$ is connected.

Case 4. $Z' \subset V_0 \cup Z \cup (\bigcup\limits_{i \in I} V_i)$ where $I \subseteq \{1, ..., p\}$. Let $I' = \{i \in I \mid |V_i| = 1\}$. Suppose first that $Z' \cap Z \neq \emptyset$, $Z' \cap V_0 \neq \emptyset$ and $Z' \cap \bigcup\limits_{i \in I} V_i = \emptyset$. We have that $|Z' \cap Z| \leq k-2$ and $|Z' \cap V_0| \leq k-2$. By Condition v), $|[V_i \cup V_{i+1}, V_0]_{G_l}| \geq k - |Z_i| + 1$ and covers $k - |Z_i| + 1$ nodes of V_0. Then there exists at least one edge linking $V_0 \backslash Z'$ and $V_i \cup V_{i+1}$. Thereby $G_l[(V_0 \backslash Z') \cup (\bigcup\limits_{i=1}^{p} V_i)]$ is connected, which is equal to $G_l \backslash Z$. Now suppose $Z \backslash Z' \neq \emptyset$. Since $G[V_0]$ is $(k+1)$-node-connected and $G[Z]$ is a complete graph, it follows that $G[V_0 \backslash Z']$ and $G[Z \backslash Z']$ are connected. By Condition iv), $[Z \backslash Z', V_0 \backslash Z']_{G_l} \neq \emptyset$. Moreover, by construction, at least one edge remains connecting V_i to $V_{i+1} \cup V_{i-1}$. And we can show along the same way as in Case 2, that V_i is connected to $V_0 \backslash Z'$. Thus

$G_l \backslash Z'$ is connected.

Now suppose that $Z' \cap Z \neq \emptyset$, $Z' \cap V_i \neq \emptyset$, for $i \in I$, $I \subseteq \{1, ..., p\}$, and $Z' \cap V_0 = \emptyset$. If $Z \subset Z'$, the proof is similar to the previous case. Suppose that $Z \backslash Z' \neq \emptyset$. We have that $|Z' \cap Z| \leq k - |I| - 1$. Let $I' = \{i \in I \mid V_i \cap Z' \neq \emptyset \text{ and } |V_i| = 1\}$. Since $G[V_i]$, for $i \in I \backslash I'$, is $(k+1)$-node-connected and $G[Z]$ is complete, it follows that $G[Z \backslash Z']$, and $G[V_i \backslash Z']$, for $i \in I \backslash I'$, are connected. By Condition iv) there exists at least one edge connecting $Z \backslash Z'$ to V_0. Also by Condition iii), $[V_i, V_0 \cup Z]_{G_l}$ contains $k-1$ edges covering different nodes in $V_0 \cup Z$ and covers $k-1$ nodes in V_i. So if no more than $k-2$ nodes are removed from $V_i \cup V_0 \cup Z$, at least one edge remains connecting $V_i \backslash Z'$ to $(Z \backslash Z') \cup V_0$. Thus $G_l \backslash Z'$ is connected.

Now, if $Z' \cap V_0 \neq \emptyset$ and $Z' \cap V_i \neq \emptyset$, for $i \in I$, and $Z' \cap Z = \emptyset$, then we have that $|Z' \cap V_0| \leq k - |I| - 1$ and $|Z' \cap V_i| \leq k - |I| - 1$ for $i \in I$. Since $G[V_i]$, for $i \in \{0\} \cup (I \backslash I')$, is $(k+1)$-node-connected, it follows that $G[V_i \backslash Z']$ is connected, for $i \in \{0\} \cup (I \backslash I')$. By Condition iv) we have that $[V_0 \backslash Z', Z] \neq \emptyset$. Moreover by Condition iii), $[V_i, V_0 \cup Z]$ contains $k-1$ edges that covers $k-1$ nodes in $V_0 \cup Z$ and $k-1$ nodes in V_i. So if no more than $k-2$ nodes are removed from $V_i \cup V_0$ at least an edge remains connecting $V_i \backslash Z'$ to $Z \cup (V_0 \backslash Z')$. Thus $G_l \backslash Z'$ is connected.

Suppose now that $Z' \cap Z \neq \emptyset$, $Z' \cap V_0 \neq \emptyset$ and $Z' \cap (\bigcup_{i \in I} V_i) \neq \emptyset$. We have that $|Z' \cap Z| \leq k - |I| - 2$ and $|Z' \cap V_i| \leq k - |I| - 2$ for $i \in I$. Since $G[V_i]$, for $i \in \{0\} \cup I \backslash I'$, is $(k+1)$-node-connected and $G[Z]$ is complete, it follows that $G[Z \backslash Z']$ and $G[V_i \backslash Z']$ are connected, for $i \in \{0\} \cup I \backslash I'$. By Condition iv), there exists at least one edge connecting $Z \backslash Z'$ to V_0. Also by Condition iii), there are $|[V_i, V_0 \cup Z]_{G_l}| \geq k-1$ edges covering different nodes in V_i, V_0 and Z. So if no more than $k-4$ nodes are removed from $V_i \cup V_0 \cup Z$, then at least an edge remains connecting $V_i \backslash Z'$ to $(Z \backslash Z') \cup (V_0 \backslash Z')$. Thus $G_l \backslash Z'$ is connected. Consequently $G_l = (V, T_l)$ is k-node-connected.

Thus $x^{T_l} \in \mathscr{F}$. ♦

Now we show that $b(e) = \rho a(e)$ for all $e \in E \backslash \Gamma$, for some $\rho \in \mathbb{R}$.

As $x^{T_1}, ..., x^{T_p}$ belong to $\mathscr{F}$, it follows that $bx^{T_1} = ... = bx^{T_p} = \beta$. Hence $b(e_1) = ... = b(e_p)$. As e_1 and e_p are arbitrary edges of $[V_1, V_2]$ and $[V_p, V_1]$, respectively, we obtain

$$b(e) = \rho \text{ for all } e \in [V_i, V_{i+1}],\ i = 1, ..., p, \text{ for some } \rho \in \mathbb{R}.$$

Let g_{l+1} be a fixed edge of $[V_{l+1}, V_0] \backslash F$, for $l \in \{0, ..., p-1\}$. Consider the edge set

$$\tilde{T}_l \;=\; (T_l \backslash \{e_l\}) \cup \{g_{l+1}\}.$$

Similary, we can show that $\tilde{T}_l$ induces a k-node-connected subgraph of G. As x^{T_l} and $x^{\tilde{T}_l}$ belong to $\mathscr{F}$, it follows in a similar way that $b(e_l) = b(g_{l+1})$. As $b(e_l) = b(e_{l+1}) = \rho$, this yields $b(g_{l+1}) = \rho$. By exchanging the roles of V_{l+1} and V_l, $l = 1, ..., p$, we obtain that $b(e) = \rho$ for all $e \in \delta_G(V_0) \backslash F$. In consequence, the $b(e)$, for all $e \in E \backslash \Gamma$ have the same value ρ.

Next, we will show that $b(e) = 0$ for all $e \in \Gamma$.

Note that there are $k+1$ edges incident to V_l in the graph induced by T_l. By using Condition iii) we can show in a similar way as in the claim above that for any edge $e \in F_l$, $l \in \{0, ..., p\}$, $T_l^* = T_l \backslash \{e\}$ also induces a k-node-connected subgraph of G. As x^{T_l} and $x^{T_l^*}$ belong to $\mathscr{F}$, it follows that $bx^{T_l} = bx^{T_l^*} = \beta$, implying that $b(e) = 0$ for all $e \in F_l$. As l is arbitrarily chosen, we obtain that $b(e) = 0$ for all $f \in F$. Moreover, as the subgraphs induced by $V_0, ..., V_p$ are all $(k+1)$-node-connected, the subgraph induced by $T_l \backslash \{e\}$, for all $e \in E_0$, is k-node-connected. This yields as before $b(e) = 0$ for all $e \in E_0$.

Now suppose that $e \in E(Z)$. By Conditions i) and iv) we can clearly see that $T_l \backslash \{e\}$ also induces a k-node-connected subgraph of G. Implying that $b(e) = 0$.

Let h be an edge of $\delta(Z)$. We can show in a similar way as in the claim above that $\overline{T}_l = T_l \backslash \{h\}$ also induces a k-node-connected subgraph of G. As $x^{\overline{T}_l}$ belongs to $\mathscr{F}$, it follows that $b(h) = 0$. Consequently $b(e) = 0$ for all $e \in \Gamma$.

Thus we obtain that $b = \rho a$, which ends the proof of the theorem. □

Corollary 4 *If the graph G is complete, then the F-node-partition inequalities* (5) *are facet-defining for kNCSP(G) if $|V_i| \geq k+2$, $i = 0, 1, ..., p$.*

Corollary 5 *Suppose that $|V_i| = 1$, for $i = 1, ..., p$, $|[V_0, V_i]| = k - |Z_i| - 1$, $|[V_i, Z_i]| = |Z_i|$, $V_1, ..., V_p$ induce an odd cycle C, and $G[V_0]$ is $(k+1)$-node-connected and $G[V_0 \cup Z]$ is complete. Then the inequality*

$$x(C) \geq \left\lceil \frac{p}{2} \right\rceil$$

is valid for kNCSP(G), and defines a facet.

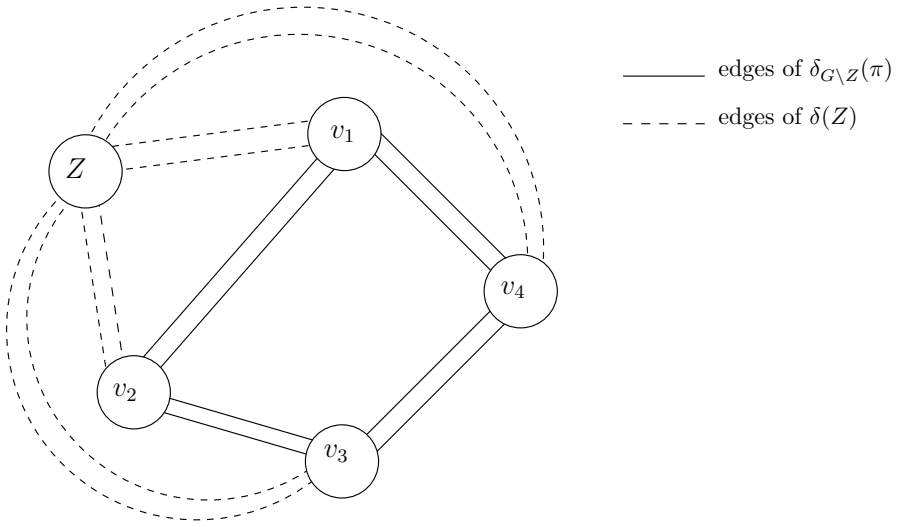

Figure 2.3: An SP-node-partition configuration for $k = 5$ and $|Z| = 2$

Theorem 11 *Let $G = (V, E)$ be a graph and an integer $k \geq 2$. Let $Z \subset V$, such that $|Z| \leq k - 1$ and $k - |Z|$ is odd. Let $\pi = (V_1, ..., V_p)$ be a partition of $V \setminus Z$ such that $(G \setminus Z)_\pi$ is series-parallel. The SP-node-partition inequality (2.10) associated with π defines a facet of kNCSP(G) if the following conditions hold.*

i) $p \geq k + 1$.

ii) $G[V_i]$ *is* $(k + 1)$*-node-connected, for* $i = 1, ..., p$.

iii) $G[Z]$ *is complete.*

iv) *For* $i = 1, ..., p$, *there exists a subset* $S_i \subset \delta(V_i)$, *with* $|S_i| = k+1$, S_i *is a matching and such that*

a) $|S_i \cap [Z, V_i]| = |Z|$,

b) $|S_i \cap [V_i, V_{i-1}]| = |S_i \cap [V_i, V_{i+1}]| = \left\lceil \dfrac{k - |Z|}{2} \right\rceil$,

where the indices are taken modulo p.

Proof.

Let us denote the SP-node-partition inequality by $ax \geq \alpha$ and let $\mathscr{F} = \{x \in kNCSP(G) \mid ax = \alpha\}$. Clearly, $\mathscr{F}$ is a proper face of kNCSP(G). Now suppose that there exists a facet defining inequality $bx \geq \beta$ of kNCSP(G) such that

$\mathscr{F} \subseteq \mathscr{F}' = \{x \in kNCSP(G) \mid bx = \beta\}$. We will show that $b = \rho a$ for some scalar $\rho \in \mathbb{R}$.

Let $E_0 = \bigcup_{i=1}^{p} E(V_i)$. And let F_i be the edge set $S_i \cap [V_i, V_{i+1}]$ for $i = 1, ..., p$. Note that $|F_i| = \frac{k-|Z|+1}{2}$. Consider $T_1 = (\bigcup_{i=1}^{p} F_i) \cup E_0 \cup E(Z) \cup \delta(Z)$.

Claim 2 *The graph H induced by T_1 is $(k+1)$-node-connected.*

Proof. Since $G[V_i]$ is $(k+1)$-node-connected, and hence $|V_i| \geq k+2$, and $|Z'| = k$, we have that $V_i \backslash Z' \neq \emptyset$ and $G[V_i \backslash Z']$ is connected for $i = 1, ..., p$. Suppose, on the contrary, that $H \setminus Z'$ is not connected. Then there is $W \subset V \setminus Z'$ such that $\delta_{H \backslash Z'}(W) = \emptyset$. As $G[V_i \backslash Z'] = H[V_i \backslash Z']$ is connected, it follows that either $W \cap (V_i \backslash Z') = \emptyset$ or $V_i \backslash Z' \subseteq W$. Also as $G[Z]$ is complete, we have that either $W \cap (Z \setminus Z') = \emptyset$ or $Z \setminus Z' \subseteq W$.

We will distinguish two cases.

Case 1. $Z \subset Z'$. Remark that any cut in $H \setminus Z$ not intersecting E_0 contains a matching of $k - |Z| + 1$ edges. Hence $H \setminus Z$ cannot be disconnected by removing $k - |Z|$ nodes. Which contradicts the fact that $\delta_{H \backslash Z'}(W) = \emptyset$.

Case 2. $Z \setminus Z' \neq \emptyset$. Thus $[Z \setminus Z', V_i] \neq \emptyset$, for $i = 1, ..., p$. This implies that $\delta_{H \backslash Z'}(W) \neq \emptyset$. Thus w.l.o.g. we may suppose $W \cap (Z \setminus Z') = \emptyset$. Here, we can show along the same line as in Case 1 that $H \setminus Z'$ cannot be disconnected. Which ends the proof. ♦

As H is a subgraph of G, G is $(k+1)$-node-connected, thus kNCSP(G) is full dimensional.

Clearly, the graph induced by $T_j = T_1 \setminus f_j$ for some $f_j \in F_j$, $j \in \{1, ..., p\}$, is k-node-connected. Moreover x^{T_j} belongs to $\mathscr{F}$.

Similary, we can see that $T_{j+1} = (F_j \setminus \{f_{j+1}\}) \cup \{f_j\}$, with $f_{j+1} \in F_{j+1}$, also induces a k-node-connected subgraph of G. As x^{T_j} and $x^{T_{j+1}}$ belong to $\mathscr{F}$, we have that $b(x^{T_j}) = b(x^{T_{j+1}})$ and hence $b(f_j) = b(f_{j+1})$. As f_j and f_{j+1} are arbitrary edges of F_j and F_{j+1}, respectively, it follows that $b(e) = b(e')$ for all $e \in F_j$ and $e' \in F_{j+1}$. Moreover, as F_j and F_{j+1} are arbitrary subsets of $[V_j, V_{j+1}]$ and $[V_{j+1}, V_{j+2}]$, respectively, we obtain that $b(e) = b(e')$ for all $e \in [V_j, V_{j+1}]$ and $e' \in [V_{j+1}, V_{j+2}]$, $j = 1, ..., p$. Consequently, by symmetry, we get $b(e) = b(e')$ for all $e, e' \in \bigcup_{i=1}^{p} [V_i, V_{i+1}]$.

Now let $h \in [V_{i_0}, V_{j_0}]$, $i_0, j_0 \in \{1, ..., p\}$ with $|i_0 - j_0| > 1$. Consider the edge sets $T_2 = (T_j \setminus \{f_{i_0-1}\}) \cup \{h\}$ and $T_3 = (T_2 \setminus \{h\}) \cup \{f_{i_0}\}$. Similary, we can show that T_2 and T_3 induce k-node-connected subgraphs of G. As x^{T_2} and x^{T_3} belong to $\mathscr{F}$, it follows that $b(x^{T_2}) = b(x^{T_3})$ and hence $b(h) = b(f_{i_0})$. This yields $b(e) = b(e')$ for all $e, e' \in \delta_{G\setminus Z}(\pi)$. Now, we will show that $b(e) = 0$ for all $e \in E_0 \cup E(Z) \cup \delta(Z)$. Consider the edge set $T_4 = T_j \setminus \{e\}$ for some $e \in E_0$. As $G[V_i]$ is $(k+1)$-node-connected, for $i = 1, ..., p$, T_4 induces a k-node-connected subgraph of G. As x^{T_j} and x^{T_4} belong to $\mathscr{F}$, we have that $b(x^{T_j}) = b(x^{T_4})$, and thus $b(e) = 0$ for all $e \in E_0$. Now suppose that $e \in E(Z)$. By Condition i) and iii) we can clearly see that $T_j \setminus \{e\}$ also induces a k-node-connected subgraph of G. Implying that $b(e) = 0$. Let g be an edge of $\delta(Z)$. We can show in a similar way that $T_5 = T_j \setminus \{g\}$ also induces a k-node-connected subgraph of G. As x^{T_5} belongs to $\mathscr{F}$, it follows that $b(e) = 0$ for all $g \in \delta(Z)$. Thus $b(e) = 0$ for all $e \in E_0 \cup E(Z) \cup \delta(Z)$. Therefore we obtain that $b = \rho a$ and the proof is complete. □

Corollary 6 *If the graph G is complete, then the SP-node-partition inequalities* (2.10) *are facet-defining for kNCSP(G) if $p \geq k+1$ and $|V_i| \geq k+2$, $i = 1, ..., p$.*

2.4 Structural properties

In this section, we discuss some structural properties of the extreme points of the linear relaxation $P(G, k)$ of the kNCSP polytope. Recall that $P(G, k)$ is the polytope given by inequalities (2.1)-(2.4).

For this, we first give some notations and definitions. Let $\overline{x} \in P(G, k)$ be a solution. We say that an inequality $ax \geq \alpha$ is *tight* for $\overline{x}$ if $a\overline{x} = \alpha$. We will denote by $E_0(\overline{x})$, $E_1(\overline{x})$ and $E_f(\overline{x})$, the following edge sets

- $E_0(\overline{x}) = \{e \in E \mid \overline{x}(e) = 0\}$,
- $E_1(\overline{x}) = \{e \in E \mid \overline{x}(e) = 1\}$,
- $E_f(\overline{x}) = \{e \in E \mid 0 < \overline{x}(e) < 1\}$.

Also we let $\mathscr{C}_{PE}(\overline{x})$ (resp. $\mathscr{C}_{PN}(\overline{x})$) be the set of cuts $\delta(W)$ (resp. node-cuts $\delta_{G\setminus Z}(W)$) that are tight for $\overline{x}$. If $\overline{x}$ is an extreme point of $P(G, k)$, then $\overline{x}$ is the unique solution of the linear system

$$S(\overline{x}) \begin{cases} x(e) = 0, & \text{for all } e \in E_0(\overline{x}), \\ x(e) = 1, & \text{for all } e \in E_1(\overline{x}), \\ x(\delta_G(W)) = k, & \text{for all cuts } \delta_G(W) \in \mathscr{C}^*_{PE}(\overline{x}), \\ x(\delta_{G\setminus Z}(W)) = k - |Z|, & \text{for all node-cuts } \delta_{G\setminus Z}(W) \in \mathscr{C}^*_{PN}(\overline{x}), \end{cases}$$

where $\mathscr{C}^*_{PE}(\overline{x})$ (resp. $\mathscr{C}^*_{PN}(\overline{x})$) is a subset of $\mathscr{C}_{PE}(\overline{x})$ (resp. $\mathscr{C}_{PN}(\overline{x})$).

Lemma 1 *Let $\overline{x} \in P(G,k)$ and $W \subseteq V$ such that the cut induced by W is tight for $\overline{x}$.*

1) If for some $R \subset V$, $\overline{x}(\delta(R)) = k$, then

$$\overline{x}(\delta(W \cap R)) = k \text{ and } \overline{x}(\delta(W \cup R)) = k.$$

2) If for some $Z \subset V$ such that $|Z| \leq k-1$, and $T \subset V \setminus Z$, $\overline{x}(\delta_{G\setminus Z}(T)) = k - |Z|$, then

$$\overline{x}(\delta_{G\setminus(Z\cap W)}(W \cap T)) = k - |Z \cap W| \text{ and } \overline{x}(\delta_{G\setminus(Z\cap \overline{W})}(W \cup T)) = k - |Z \cap \overline{W}|.$$

Proof.

1) The proof is similar to that of [20].

2) Suppose that $Z \cap W \neq \emptyset \neq Z \cap \overline{W}$. Also suppose that $T \cap W \neq \emptyset$ and $T \not\subset W$, $W \not\subset T$ and $T \cup W \neq V \setminus Z$. If this is not the case, then we are done. Let $T_1 = T \cap W$, $T_2 = T \cap \overline{W}$, $Z_1 = Z \cap W$, $Z_2 = Z \cap \overline{W}$, $T_3 = W \setminus (T \cup Z_1)$ and $T_4 = \overline{W} \setminus (T \cup Z_2)$. Thus $T_i \neq \emptyset$ for $i = 1, ..., 4$. As $\delta(W) \in \mathscr{C}_P E(\overline{x})$, we have that

$$\begin{aligned} k = \overline{x}(\delta(W)) &= \overline{x}(\delta(T_1,T_2)) + \overline{x}(\delta(T_1,T_4)) + \overline{x}(\delta(T_3,T_2)) \\ &+\overline{x}(\delta(T_3,T_4)) + \overline{x}(\delta(T_1,Z_2)) + \overline{x}(\delta(T_3,Z_2)) \\ &+\overline{x}(\delta(T_2,Z_1)) + \overline{x}(\delta(T_4,Z_1)) + \overline{x}(\delta(Z_1,Z_2)). \end{aligned} \tag{2.13}$$

And as $\delta_{G\setminus Z}(T) \in \mathscr{C}_n(\overline{x})$, we have that

$$\begin{aligned} k - |Z| = \overline{x}(\delta_{G\setminus Z}(T)) &= \overline{x}(\delta(T_1,T_3)) + \overline{x}(\delta(T_1,T_4)) \\ &+\overline{x}(\delta(T_2,T_3)) + \overline{x}(\delta(T_2,T_4)). \end{aligned} \tag{2.14}$$

By considering the node-cuts $\delta_{G\setminus Z_1}(T_1)$, $\delta_{G\setminus Z_2}(T_2)$, $\delta_{G\setminus Z_1}(T_3)$ and $\delta_{G\setminus Z_2}(T_4)$, we have that

$$\begin{array}{l} k-|Z_1| \leq \overline{x}(\delta_{G\setminus Z_1}(T_1)) = \overline{x}(\delta(T_1,T_2)) \\ +\overline{x}(\delta(T_1,T_3)) + \overline{x}(\delta(T_1,T_4)) + \overline{x}(\delta(T_1,Z_2)), \end{array} \tag{2.15}$$

$$\begin{array}{l} k-|Z_2| \leq \overline{x}(\delta_{G\setminus Z_2}(T_2)) = \overline{x}(\delta(T_2,T_1)) + \overline{x}(\delta(T_2,T_3)) \\ +\overline{x}(\delta(T_2,T_4)) + \overline{x}(\delta(T_2,Z_1)), \end{array} \tag{2.16}$$

$$\begin{array}{l} k-|Z_1| \leq \overline{x}(\delta_{G\setminus Z_1}(T_3)) = \overline{x}(\delta(T_3,T_1)) \\ +\overline{x}(\delta(T_3,T_2)) + \overline{x}(\delta(T_3,T_4)) + \overline{x}(\delta(T_3,Z_2)), \end{array} \tag{2.17}$$

$$\begin{array}{l} k-|Z_2| \leq \overline{x}(\delta_{G\setminus Z_2}(T_4)) = \overline{x}(\delta(T_4,T_1)) + \overline{x}(\delta(T_4,T_2)) \\ +\overline{x}(\delta(T_4,T_3)) + \overline{x}(\delta(T_4,Z_1)). \end{array} \tag{2.18}$$

As $\overline{x}(e) \geq 0$ for all $e \in E$, by adding (2.13), (2.14) (2.17) and (2.18), and combining the resulting equations with (2.15) and (2.16), we get $x(\delta_{G\setminus Z_1}(T_1)) = k-|Z_1|$ and $x(\delta_{G\setminus Z_2}(T_4)) = k-|Z_2|$, which ends the proof.

□

From Lemma 1, we can show the following result. Its proof is omitted since it follows the same lines as a similar result in [20].

Lemma 2 *Let $\overline{x}$ be an extreme point of $P(G,k)$, and $W \subset V$ such that $\overline{x}(\delta(W)) = k$. Then the system $S(\overline{x})$ can be chosen so that*

1) *a cut $\delta(R) \in \mathscr{C}^*_{PE}(\overline{x})$ is such that $R \subset W$ or $R \subset \overline{W}$;*
2) *a node-cut $\delta_{G\setminus Z}(T) \in \mathscr{C}^*_{PN}(\overline{x})$ is such that $(T \cup Z) \subset W$, $(T \cup Z) \subset \overline{W}$, $T \subset W$ and $Z \subset \overline{W}$, or $T \subset \overline{W}$ and $Z \subset W$.*

2.5 Reduction operations

In this section we introduce some reduction operations defined with respect to a solution $\overline{x}$ of $P(G,k)$. These operations will be considered in a preprocessing phase for separating violated inequalities in our Branch-and-Cut algorithm. Let θ_1, θ_2, θ_3 and θ_4 be the reduction operations defined as follows.

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, $\overline{x}(e) = 1$ for all $e \in E(W)$ and $\overline{x}(\delta(W)) = k$.

θ_3**:** Contract a node subset $W \subseteq V$ such that $|W| \geq k$, $|\overline{W}| \geq k$, $\overline{x}(e) = 1$ for all $e \in E(W)$, and $|\delta_G(W)| = k$.

θ_4**:** Replace a set of parallel edges by only one edge.

We have the following results.

Lemma 3 *Let $G' = (V, E')$ be the graph obtained from G by the application of Operation θ_1 and let $\overline{x}'$ be the restriction of $\overline{x}$ to G'. Then $\overline{x}'$ is an extreme point of $P(G', k)$ if and only if $\overline{x}$ is an extreme point of $P(G, k)$.*

Proof. Easy. □

Lemma 4 *Let $G' = (V', E')$ and $\overline{x}'$ be the graph and the solution obtained from G and $\overline{x}$ by the application of Operation θ_2. Suppose that*

1) $\overline{x}' \in P(G', k)$,

2) for all $Z \subset W$, $|Z| \leq k-1$, $\delta_{G \setminus Z}(T) \notin \mathscr{C}_{PN}(\overline{x})$ for all $T \subseteq \overline{W}$.

Then $\overline{x}'$ is an extreme point of $P(G', k)$ if $\overline{x}$ is an extreme point of $P(G, k)$.

Proof. Let W be a node set of G contracted by Operation θ_2. As $\delta(W) \in \mathscr{C}_{PE}(\overline{x})$, by Lemma 2, the system $S(\overline{x})$ can be chosen in such a way that for every $\delta(R) \in \mathscr{C}^*_{PE}(\overline{x})$ (resp. $\delta_{G\setminus Z_T}(T) \in \mathscr{C}^*_{PN}(\overline{x})$) either $R \subseteq W$ or $R \subseteq \overline{W}$ (resp. $(T \cup Z_T) \subset W$, $T \subset W$ and $Z_T \subset \overline{W}$, or $T \subset \overline{W}$ and $Z_T \subset W$). As $\overline{x}(e) = 1$ for all $e \in E(W)$ and $G[W]$ is k-edge connected, this implies that $\mathscr{C}^*_{PE}(\overline{x}) \subseteq \mathscr{C}_{PE}(\overline{x}')$. Moreover by 2) it follows that if $\delta_{G\setminus Z_T}(T)$ is tight for $\overline{x}$ and $Z_T \subseteq W$, then $W \cap T \neq \emptyset$ and $W \setminus (Z_T \cup T) \neq \emptyset$. Let $T_1 = W \cap T$ and $T_2 = W \setminus (Z_T \cup T)$. We have that $k - |Z_T| = \overline{x}(\delta_{G\setminus Z_T}(T)) \geq \overline{x}(\delta(T_1, T_2)) \geq k$, a contradiction. The last inequality comes from the fact that $G[W]$ is k-edge connected and $\overline{x}(e) = 1$ for all $e \in E(W)$. In consequence, all the node-cuts $\delta_{G\setminus Z_T}(T)$ of $\mathscr{C}^*_{PN}(\overline{x})$ are such that $Z_T \subset \overline{W}$. However these are at the same time tight for $\overline{x}'$. Thus $\mathscr{C}^*_{PN}(\overline{x}) \subset \mathscr{C}_{PN}(\overline{x}')$. Let $S'(\overline{x})$ be the system obtained from $S(\overline{x})$ by deleting the equations $x(e) = 1$ for all $e \in E(W)$. Then $\overline{x}'$ is the unique solution of $S'(\overline{x})$. As all the equations of $S'(\overline{x})$ come from $P(G', k)$ and by 1) $\overline{x}' \in P(G', k)$, it follows that $\overline{x}'$ is an extreme point of $P(G', k)$. □

Lemma 5 *Let $G' = (V', E')$ and $\overline{x}'$ be the graph and the solution obtained from G and $\overline{x}$, respectively, by the application of Operation θ_3. Then $\overline{x}'$ is an extreme point of $P(G', k)$.*

Proof. Let $W \subseteq V$ be a node set satisfying the conditions of Operation θ_3. First observe that as $|\delta(W)| = k$, we have that $\overline{x}(e) = 1$ for all $e \in \delta(W)$ and $\overline{x}(\delta(W)) = k$. Thus, by Lemma 2, $S(\overline{x})$ can be chosen so that for every node-cut $\delta_{G\setminus Z}(T) \in \mathscr{C}^*_{PN}$, we have $(T \cup Z) \subset W$, $(T \cup Z) \subset \overline{W}$, $T \subset W$ and $Z \in \overline{W}$, or $T \subset \overline{W}$ and $Z \in W$. We will show that any cut $\delta(R) \in \mathscr{C}^*_{PE}(\overline{x})$ is such that $R \subset \overline{W}$, and any node-cut $\delta_{G\setminus Z}(T) \in \mathscr{C}^*_{PN}(\overline{x})$ is such that $(T \cup Z) \subset \overline{W}$. Suppose the contrary and consider first that for some $\delta(R) \in \mathscr{C}^*_{PE}(\overline{x})$, $R \subsetneq W$. As $\overline{x}(e) = 1$, for all $e \in E(W) \cup \delta(W)$, one can see that $|\delta(R)| = k$, and hence $\overline{x}(\delta(R)) = k$ can be obtained from $\overline{x}(e) = 1$, for all $e \in \delta(R)$, contradicting the fact that $\delta(R) \in \mathscr{C}^*_{PE}(\overline{x})$. Now suppose that for some node-cut $\delta_{G\setminus Z}(T) \in \mathscr{C}^*_{PN}(\overline{x})$ either $(T \cup Z) \subset W$ or $T \subseteq W$ and $Z \subseteq \overline{W}$. We can show similarily to the previous case that $|\delta_{G\setminus Z}(T)| = k - |Z|$ and that $\overline{x}(\delta_{G\setminus Z}(T)) = k - |Z|$ can be obtained from $\overline{x}(e) = 1$, for all $e \in \delta_{G\setminus Z}(T)$, which yields a contradiction.

We consider now a node-cut $\delta_{G\setminus Z}(T) \in \mathscr{C}^*_{PN}(\overline{x})$ such that $T \subseteq \overline{W}$ and $Z \subseteq W$. Notice that, as $|W| \geq k$, we have that $W \setminus Z \neq \emptyset$. If $T = \overline{W}$, then $\overline{x}(\delta_{G\setminus Z}(T)) = \overline{x}(\delta(W \setminus Z, T)) = |\delta(W \setminus Z, T)| = k - |Z|$. Thus, $\overline{x}(\delta_{G\setminus Z}(T)) = k - |Z|$ can be obtained from the equations $\overline{x}(e) = 1$, for all $e \in \delta_{G\setminus Z}(T)$, contradicting the fact that $\delta_{G\setminus Z}(T) \in \mathscr{C}^*_{PN}(\overline{x})$. Thus, $\overline{W} \setminus T \neq \emptyset$. For convenience, we let $T_1 = W \setminus Z$ and $T_2 = \overline{W} \setminus T$. First, note that

$$\overline{x}(\delta_{G\setminus Z}(T)) = \overline{x}(\delta(T, T_1)) + \overline{x}(\delta(T, T_2)) = k - |Z|. \tag{2.19}$$

Equation 2.19 together with the cut inequality induced by T yields

$$\overline{x}(\delta(T,Z)) \geq |Z|. \tag{2.20}$$

Also, as by the assumption $|\delta(W)| = k$, we have that

$$\overline{x}(\delta(T,T_1)) + \overline{x}(\delta(T,Z)) + \overline{x}(\delta(T_2,T_1)) + \overline{x}(\delta(T_2,Z)) = k. \tag{2.21}$$

This equation, together with the node-cut inequality induced by $\delta_{G\setminus Z}(T_2)$ implies that

$$\overline{x}(\delta(T,Z)) + \overline{x}(\delta(T_2,Z)) \leq |Z|. \tag{2.22}$$

Thus, by inequalities (2.20) and (2.22), we have that $\overline{x}(\delta(T,Z)) = |Z|$ and $\overline{x}(\delta(T_2,Z)) = 0$, and hence

$$\overline{x}(\delta(T)) = \overline{x}(\delta(T,T_1)) + \overline{x}(\delta(T,Z)) + \overline{x}(\delta(T,T_2)) = k. \tag{2.23}$$

Moreover, as $\overline{x}(e) = 1$, for all $e \in \delta(W)$, we have that $\overline{x}(\delta(T,Z)) = |Z| = |\delta(T,Z)|$. Therefore, $\overline{x}(\delta_{G\setminus Z}(T)) = k - |Z|$ can be obtained from (2.23) and the $\overline{x}(e) = 1$, for all $e \in \delta(T,Z)$, and hence, can be replaced in $S(\overline{x})$ by equation (2.23).

Consequently, the system $S(\overline{x})$ can be chosen so that $R \subseteq \overline{W}$ for every cut $\delta(R) \in \mathscr{C}^*_{PE}(\overline{x})$ and $T \cup Z \subseteq \overline{W}$ for every node-cut $\delta_{G\setminus Z}(T) \in \mathscr{C}^*_{PN}(\overline{x})$. This also implies that $\mathscr{C}^*_{PE}(\overline{x}) \cup \mathscr{C}^*_{PN}(\overline{x}) \subseteq \mathscr{C}^*_{PE}(\overline{x}') \cup \mathscr{C}^*_{PN}(\overline{x}')$. Thus, $\overline{x'}$ is the unique solution of a subsystem of $S(\overline{x})$. As all the equations of that subsystem correspond to constraints of $P(G \setminus W, k)$, this implies that $\overline{x}'$ is an extreme point of $P(G \setminus W, k)$. □

Lemma 6 *Let $G' = (V', E')$ be the graph obtained from G by the application of Operation θ_4. Let E_0 be the set of parallel edges of G and e_0 the edge replacing E_0 in G'. Let $\overline{x}'$ be the solution given by $\overline{x}'(e) = \overline{x}(e)$ if $e \in E \setminus E_0$ and $\overline{x}'(e) = 1$ if $e = e_0$. Then $\overline{x}'$ is an extreme point of $P(G', k)$.*

Proof. Observe that for every cut $\delta(W)$ (node-cut $\delta_{G\setminus Z}(W)$) either $E_0 \subseteq \delta(W)$ ($E_0 \subset \delta_{G\setminus Z}(W)$) or $E_0 \cap \delta(W) = \emptyset$ ($E_0 \cap \delta_{G\setminus Z}(W) = \emptyset$). Moreover, E_0 cannot contain more than two edges with fractional value. Indeed, if $e_1, e_2 \in E_0$ and $0 < x(e_1) < 1$ and $0 < x(e_2) < 1$, let $\overline{x}^*$ be the solution given by $\overline{x}^*(e) = \overline{x}(e)$ if $e \in E \setminus \{e_1, e_2\}$, $\overline{x}^*(e) = \overline{x}(e) + \epsilon$ if $e = e_1$ and $\overline{x}^*(e) = \overline{x}(e) - \epsilon$ if $e = e_2$, where ϵ is a positive scalar sufficiently small. We then have that $\overline{x}^*$ is also a solution of $S(\overline{x})$, which is a contradiction.

We claim that E_0 does not contain any edge with fractional value. Suppose, on the contrary that h is such an edge. Then $\overline{x}(E_0) > 1$. Therefore there exists a cut or a node-cut of system $S(\overline{x})$ containing h. Let v be an extremity of h. Let $\delta(S)$ be a cut of $\mathscr{C}_e^*(\overline{x})$ that contains h. Thus $E_0 \subset \delta(S)$. Suppose w.l.o.g., that $v \in \overline{S}$. Consider the node-cut $\delta_{G\setminus v}(S)$. We have that $\overline{x}(\delta_{G\setminus v}(S)) \leq \overline{x}(\delta(S)\setminus E_0) < k-1$, a contradiction. Now consider a node-cut $\delta_{G\setminus Z}(T)$ of $\mathscr{C}_n^*(\overline{x})$ that contains h and hence E_0. As $\overline{x}(E_0) > 1$, one must have $|Z| < k-1$. So suppose that $|Z| < k-1$. Suppose w.l.o.g., that $v \in V \setminus (T \cup Z)$. Let $Z' = Z \cup \{v\}$. We have $\overline{x}(\delta_{G\setminus Z'}(T)) \leq \overline{x}(\delta_{G\setminus Z}(T)) - 1 - \overline{x}(h) = k - (|Z|+1) - \overline{x}(h) < k - |Z'|$, a contradiction. Consequently, $\overline{x}(e) = 1$ for all $e \in E_0$. From the development above we also deduce that neither a cut of $\mathscr{C}_e^*(\overline{x})$ nor a node-cut of $\mathscr{C}_n^*(\overline{x})$ intersects E_0. Hence $\mathscr{C}_e^*(\overline{x}) \cup \mathscr{C}_n^*(\overline{x}) \subset \mathscr{C}(\overline{x}')$. Moreover, we have that $\overline{x}' \in P(G', k)$. Obviously, $\overline{x}'$ satisfies the trivial inequalities as well as the cut and node-cut inequalities that do not contain h. Let $\delta(W)$ be a cut that contains h. Suppose $v \in \overline{W}$. We have that $\overline{x}'(\delta(W)) = \overline{x}'(h) + \overline{x}'(\delta(W) \setminus \{h\}) = 1 + \overline{x}(\delta(W) \setminus E_0) = 1 + \overline{x}(\delta_{G\setminus v}(W)) \geq k$. Consider now a node-cut $\delta_{G\setminus Z}(T)$ containing h. If $|Z| = k-1$, as $\overline{x}'(h) = 1$ and $h \in \delta_{G\setminus Z}(T)$, we have that $\overline{x}'(\delta_{G\setminus Z}(T)) \geq 1$. If $|Z| < k-1$, then let $Z' = Z \cup \{v\}$. We have that $\overline{x}'(\delta_{G\setminus Z}(T)) \geq 1 + \overline{x}'(\delta_{G\setminus Z'}(T)) \geq 1 + k - |Z'| = 1 + k - |Z| - 1 = k - |Z|$. □

As we will see later, the reduction operations $\theta_1, ..., \theta_4$ can be used as a preprocessing for the separation procedures in our Branch-and-Cut algorithm.

2.6 Conclusion

We have studied in this chapter the k-node-connected subgraph problem with high connectivity requirement, that is, when $k \geq 3$. We have investigated the dimension of the associated polytope, we have presented some classes of valid inequalities and described some conditions for these inequalities to be facet defining for the associated polytope. We have also investigated the structural properties of the extreme points of the linear relaxation of the problem and presented some reduction operations, which will be used as a preprocessing phase for the separation of the valid inequalities.

Chapter 3

Branch-and-Cut Algorithm for the kNCSP

Contents

3.1 Branch-and-Cut algorithm

In this section, we present a Branch-and-Cut algorithm for the kNCSP. The algorithm is based on the theoretical results presented in the previous sections. We will first present the general framework of the algorithm, then we will address the main issues of our algorithm, that are the separation procedures for the various inequalities we will use, and a primal heuristic.

We will consider a graph $G = (V, E)$ and a weight vector $c \in \mathbb{R}^E$ associated with the edges of G. We let $k \geq 1$ be the connectivity requirement.

3.1.1 General framework

To start the optimization we consider the following linear program consisting in the cut constraints induced by node sets $\{u\}$, for every $u \in V$ together with the trivial inequalities, that is

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

If the optimal solution $\overline{y} \in \mathbb{R}^E$ of the above LP is feasible for the kNCSP, that is, it is integer and it satisfies all the cut and node-cut inequalities, then it is optimal for the problem. Usually, $\overline{y}$ is not feasible for the kNCSP. Thus, we need to generate further valid inequalities for the problem which are violated by $\overline{y}$. This is done by addressing the separation problem associated with the cut and node-cut inequalities, respectively, and the other families of inequalities we consider in our algorithm. Recall that the *separation problem* associated with $\overline{y}$ and a family of inequalities $\mathcal{F}$ is to say if $\overline{y}$ satisfies or not all the inequalities of $\mathcal{F}$, and if not, exhibit at least one inequality of $\mathcal{F}$ which is violated by $\overline{y}$. An algorithm solving a separation problem is called a *separation algorithm*. In our algorithm, we use the inequalities that we described in the previous sections and perform their separation in the following order

1) cut inequalities,

2) node-cut inequalities,

3) SP-node-partition inequalities,

4) F-node-partition inequalities,

5) node-partition inequalities.

We move to a class of inequalities when the separation algorithm for the previous class of inequalities has not found any violated inequality. We may add several inequalities at the same time in the Branch-and-Cut algorithm. Moreover, all the inequalities are global, that is they are valid for all the nodes of the Branch-and-Cut tree.

Remark that the separations are done on the graph obtained after repeated applications of the reduction operations $\theta_1, ..., \theta_4$ to the graph G and solution $\overline{y}$. If G' is the reduced graph and $\overline{y}'$ is the restriction of $\overline{y}$ to G', then by Lemmas 3-6, $\overline{y}'$ is an extreme point of $P(G', k)$ if $\overline{y}$ is an extreme point of $P(G, k)$. Moreover, we have the following results which are easily seen to be true.

Lemma 7 *Let $a'x \geq \alpha$ be an F-node-partition inequality (respectively node-partition inequality) valid for kNCSP(G'), induced by a partition $\pi' = (V'_0, V'_1, ..., V'_p)$, $p \geq 2$ and an edge set F (respectively $\pi' = (V'_1, ..., V'_p)$, $p \geq 3$) of $V' \setminus Z$, with $Z \subset V$. Let $\pi = (V_0, V_1, ..., V_p)$, $p \geq 2$ (respectively $\pi = (V_1, ..., V_p)$, $p \geq 3$) be the partition of V obtained by expanding the elements of π'. Let $ax \geq \alpha$ be the inequality such that*

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

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

Lemma 8 *Let $a'x \geq \alpha$ be an SP-node-partition inequality valid for kNCSP(G'), induced by a partition $\pi' = (V'_1, ..., V'_p)$, $p \geq 3$ of $V' \setminus Z$, with $Z \subset V$ such that $|Z| \leq k-1$. Let $\pi = (V_1, ..., V_p)$, $p \geq 3$ be the partition of $V \setminus Z$ obtained by expanding the subsets V'_i of π'. Let $ax \geq \alpha$ be the lifted SP-node-partition inequality obtained from $a'x \geq \alpha$ by application of the lifting procedure described in Section 2.2 for the edges of $E \setminus E'$. Then $ax \geq \alpha$ is violated by $\overline{y}$, if $a'x \geq \alpha$ is violated by $\overline{y}'$.*

Lemmas 7 and 8 show that the separation of F-node-partition, SP-node-partition and node-partition inequalities can be done in the reduced graph associated with any fractional solution of $P(G, k)$.

3.1.2 Separation algorithms

Now we describe the separation algorithms we have devised for the cut, node-cut, SP-node-partition, F-node-partition and node-partition inequalities.

We start by the separation of the cut inequalities (2.3). It is well known that the separation of the cut inequalities (2.3) reduces to computing a minimum weight cut in G with respect to weight vector $\overline{y}$. Indeed, there is a violated cut inequality (2.3) if and only if the minimum weight of a cut, w.r.t. weight vector $\overline{y}$, is $< k$. One can compute a minimum weight cut in polynomial time by using any minimum cut algorithm, and especially by using the Gomory-Hu algorithm [40] which computes the so-called Gomory-Hu cut tree. This algorithm consists in $|V| - 1$ maximum flow computations.

Now we discuss the separation of the node-cut inequalities (2.4). In what follows, we show that these inequalities can be separated in polynomial time. In fact, Grötschel et al. [47] present a separation algorithm for inequalities (2.4) based on a transformation of the graph G into a directed graph $\widetilde{G} = (\widetilde{V}, \widetilde{A})$. This transformation is presented as follows. For each node $u \in V$, we add in $\widetilde{V}$ two copies u^- and u^+ of u. The arc set is built in the following way. First, for each edge $uv \in E$, we add two arcs (v^+, u^-) and (u^+, v^-). Finally, for every node $u \in V$, we add an arc of the form (u^-, u^+). We also let $\widetilde{y} \in \mathbb{R}^{\widetilde{A}}$ be a weight vector given by

$$\widetilde{y}(a) = \begin{cases} \overline{y}(uv) & \text{for } a = (u^+, v^-) \text{ and } a = (v^+, u^-), \\ 1 & \text{if } a = (u^-, u^+) \text{ for all nodes } u \in V. \end{cases}$$

One can see that a cut $\delta(W)$ in G corresponds to a dicut which does not contain an arc of the form (u^-, u^+). Conversely, a dicut $\delta^+_{\widetilde{G}}(\widetilde{W})$ of $\widetilde{G}$ which does not contain any arc of the form (u^-, u^+) corresponds to a cut of G. Also, a node-cut $\delta_{G\setminus Z}(W)$ of G corresponds to a dicut of $\widetilde{G}$ which contains $|Z|$ arcs of the form (u^-, u^+). Conversely, a dicut of $\widetilde{G}$ which contains at least one arc of the form (u^-, u^+) corresponds to a node-cut of G. The corresponding node set Z is given by the nodes $u \in V$ such that $(u^-, u^+) \in \delta^+_{\widetilde{G}}(\widetilde{W})$, and the edges of $\delta_{G\setminus Z}(W)$ are given by the arcs of $\delta^+_{\widetilde{G}}(\widetilde{W})$ of the form (u^+, v^-) with $u^+ \in \tilde{W}$, $v^- \in \tilde{V} \setminus \tilde{W}$.
Thus, cuts and node-cuts of G corresponds to dicuts of $\widetilde{G}$ which does not contain arcs of the form (u^-, u^+), and vice-versa. Moreover, we have that

- if $\delta(W)$ in G and $\delta^+_{\widetilde{G}}(\widetilde{W})$ in $\widetilde{G}$ are corresponding cuts, then $\overline{y}(\delta(W)) = \widetilde{y}(\delta^+_{\widetilde{G}}(\widetilde{W}))$;
- if $\delta_{G\setminus Z}(W)$ in G and $\delta^+_{\widetilde{G}}(\widetilde{W})$ in $\widetilde{G}$ are corresponding cuts, then $\overline{y}(\delta_{G\setminus Z}(W)) + |Z| = \widetilde{y}(\delta^+_{\widetilde{G}}(\widetilde{W}))$.

Thus, there is a cut or node-cut inequality violated by $\overline{y}$ if and only if there exists a dicut $\delta^+_{\widetilde{G}}(\widetilde{W})$ in $\widetilde{G}$ whose weight with respect to $\widetilde{y}$ is $< k$. Notice that, if we assume

the cut inequalities to be all satisfied by $\overline{y}$, finding violated node-cut inequalities then reduces to compute a minimum weight cut in $\widetilde{G}$ w.r.t. weight vector $\widetilde{y}$.

Consequently, our separation algorithm for node-cut inequalities is as follows. First, we assume that the cut inequalities are all satisfied by $\overline{y}$. We build the graph $\widetilde{G}$ and compute a minimum weight cut, say $\delta^+_{\widetilde{G}}(\widetilde{W}^*)$, w.r.t. $\widetilde{y}$. If $\widetilde{y}(\delta^+_{\widetilde{G}}(\widetilde{W}^*)) \geq k$, then every node-cut inequality is satisfied by $\overline{y}$, and the algorithm stops. If $\widetilde{y}(\delta^+_{\widetilde{G}}(\widetilde{W}^*)) < k$, then there is a violated node-cut inequality induced by a node-cut $\delta_{G\setminus Z}(W)$ with $Z \subseteq V$, $|Z| \leq k-1$, and $W \subseteq V \setminus Z$. The node sets Z and W are given by

$$Z = \{\ u \in V \text{ such that } (u^-, u^+) \in \delta^+_{\widetilde{G}}(\widetilde{W}^*)\},$$

$$W = \{u \in V \text{ such that } u^-, u^+ \in \widetilde{W} \text{ or } u^+ \in \widetilde{W} \text{ and } u^- \in \widetilde{V} \setminus \widetilde{W}\}.$$

Finally, computing a minimum weight cut in $\widetilde{G}$ can be done in polynomial time by computing, for every pair of nodes $(s,t) \in V \times V$, with $s \neq t$, a maximum flow in $\widetilde{G}$ from source node s^+ to destination t^-. This, hence, reduces our algorithm to $|V|(|V|-1)/2$ maximum flow computations in $\widetilde{W}$, which is polynomial.

Finally, we consider the separation problems for node-partition, SP-node-partition and F-node-partition inequalities. First notice that the separation problem of node-partition inequalities is NP-Hard even when $Z = \emptyset$. For our purpose, we consider these inequalities in the case where $Z = \emptyset$. Thus, the corresponding node-partition, SP-node-partition and F-node-partition inequalities also correspond to partition, SP-partition and F-partition inequalities, which are valid for the kECSP on G. Therefore, to separate these inequalities, we use the separation heuristics developed in Bendali et al. [9] for these latter inequalities. These algorithms are applied on the graph G' and solution $\overline{y}'$ obtained by the application of the reduction operations to G and $\overline{y}$. As mentioned before, by Lemmas 7 and 8, any violated node-partition, SP-node-partition and F-node-partition inequality found in G' by the separation procedure is valid for kNCSP(G) and is also violated by $\overline{y}$.

3.1.3 Primal heuristic

Next, we discuss a primal heuristic for the problem. The aim of this heuristic is to produce, for a given instance, good upper bounds of the optimal solution of the problem. Such upper bounds are used by the Branch-and-Cut algorithm to prune unrelevant branches of the Branch-and-Cut tree. This also ensures that Branch-and-Cut algorithm produces a feasible solution, even if it reaches the maximum CPU time.

The primal heuristic we have developed for our purpose consider a fractional solution $\overline{y}$ obtained at the end of the cutting plane phase, at the root node of the Branch-and-Cut tree. The aim of the heuristic is to transform $\overline{y}$ into a feasible solution for the problem. To do this, we first build the graph $\overline{G} = (V, \overline{E})$ obtained by removing from G every edge $e \in E$ with $\overline{y}(e) = 0$. Then, we iteratively remove from $\overline{G}$ all the edges uv such that u and v are both incident in $\overline{G}$ to at least $k + 1$ edges. We denote by $\overline{G}' = (V, \overline{E}')$ the resulting graph, and by z the incidence vector of $\overline{G}'$. Next, we check if $\overline{G}'$ is k-node-connected. We do this by calling the separation algorithms for the cut and node-cut inequalities described in the previous section on z and $\overline{G}'$. If there is a cut (resp. node-cut) inequality induced by a cut $\delta_{\overline{G}'}(W)$ (resp. node-cut $\delta_{\overline{G}'\setminus Z}(W)$), which is violated by z, then we add in $\overline{G}'$ an edge $e \in \delta_G(W) \setminus \overline{E}'$ (resp. $e \in \delta_{G\setminus Z}(W) \setminus \overline{E}'$) whose weight $c(e)$ is minimum. If there is no cut and node-cut inequality violated by z, then $\overline{G}'$ is feasible for the kNCSP. We repeat this procedure until the graph $\overline{G}'$ is k-node-connected.

Finally, the algorithm computes and returns the weight of the graph $\overline{G}'$ obtained at the end of the previous step. The whole procedure is summarized by Algorithm 3 below.

Algorithm 3: Primal Heuristic Algorithm for the kNCSP

Data: An undirected graph $G = (N, E)$, an integer k, a fractional solution $\overline{y}$
Result: An Upper Bound UB for the kNCSP
begin
  Build the graph $\overline{G} = (V, \overline{E})$ by removing from G every edge e such that $\overline{y}(e) = 0$;

  /*Remove the edges uv such that $|\delta_{\overline{G}}(u)| \geq k+1$ and $|\delta_{\overline{G}}(v)| \geq k+1$*/
  foreach *edge* $uv \in \overline{E}$ **do**
    if $|\delta_{\overline{G}}(u)| \geq k+1$ *and* $|\delta_{\overline{G}}(v)| \geq k+1$ **then**
      $\overline{E} \leftarrow \overline{E} \setminus \{uv\}$;
    end
  end

  /*Check if the resulting graph is k-node-connected*/
  $FeasibleSolutionFound \leftarrow False$;
  repeat
    Let z be the incidence vector of $\overline{E}$;
    Call the separation procedure for cut inequalities;
    if *there is a cut inequality violated by* z **then**
      Let $\delta_{\overline{G}}(W)$ be the cut inducing the violated cut inequality;
      Choose an edge $e \in \delta_G(W) \setminus \overline{E}$ with minimum weight;
      $\overline{E} \leftarrow \overline{E} \cup \{e\}$;
    end
    else
      Call the separation procedure for node-cut inequalities with solution z and graph $\overline{G}$;
      if *there is a node-cut inequality violated by* z **then**
        Let $\delta_{\overline{G}\setminus Z}(W)$ be the node-cut inducing the violated node-cut inequality;
        Choose an edge $e \in \delta_{G\setminus Z}(W) \setminus \overline{E}$ with minimum weight;
        $\overline{E} \leftarrow \overline{E} \cup \{e\}$;
      end
      else
        $FeasibleSolutionFound \leftarrow True$;
      end
    end
  until *FeasibleSolutionFound = True*;
  $UB \leftarrow 0$;
  foreach *edge* $e \in \overline{E}$ **do**
    $UB \leftarrow UB + c(e)$;
  end
  return UB;
end

3.2 Computational Results

Now we present the computational results we have obtained with our Branch-and-Cut algorithm for the kNCSP. The algorithm has been implemented in C++ using CPLEX 12.5 [3] and Concert Technology library. All the experiments have been done on a computer equiped with a 2.10 GHz x4 Intel Core(TM) i7-4600U processor and running under linux with 16 GB of RAM. We have set the maximum CPU time to five hours. We have tested our algorithm on several instances composed of graphs taken from SNDLIB [1] and TSPLIB [2]. These are complete graphs where each node is given coordinates in the plane. The weight of each edge uv is the rounded euclidian distance between the vertices u and v. The graphs we have considered have up to 65 nodes for SNDLIB graphs and up to 150 nodes for TSPLIB graphs.
The tests have been performed for $k = 3, 4, 5$, and in all the experiments, we have used the reduction operations described in the previous sections, unless specified.

For each instance, we have run the algorithm three times. The first run (Run 1) is performed with all the inequalities presented before and the reduction operations included in the algorithm. The second run (Run 2) is performed without the reduction operations. The third run (Run 3) is performed with the reduction operations and with only the cut and node-cut inequalities. The results are given in Tables 3.2-3.8. Each instance is given by its name followed by the number of nodes of the graph. The other entries of

#EC	the number of generated cut inequalities
#NC	the number of generated node-cut inequalities
#SPC	the number of generated SP-node-partition inequalities
#NFPC	the number of generated F-node-partition inequalities
#NPC	the number of generated node-partition inequalities
COpt	the value of the optimal solution
Gap1	the relative error between the best upper bound and the lower bound obtained at the root node of the Branch-and-Cut tree during Run 1
Gap2	the relative error between the best upper bound and the lower bound obtained at the root node of the Branch-and-Cut tree during Run 2
Gap3	the relative error between the best upper bound and the lower bound obtained at the root node of the Branch-and-Cut tree during Run 3
NSub1	the number of nodes in the Branch-and-Cut tree obtained at Run 1
NSub2	the number of nodes in the Branch-and-Cut tree obtained at Run 2
NSub3	the number of nodes in the Branch-and-Cut tree obtained at Run 3
CPU1	the total CPU time in hh:mn:sec achieved at Run 1
CPU2	the total CPU time in hh:mn:sec achived at Run 2
CPU3	the total CPU time in hh:mn:sec achived at Run 3

the tables are:

The gaps are all given in percentage. The instances indicated with "*" are those for which the maximum CPU time has been reached by the Branch-and-Cut algorithm.

We start our experiments by running the algorithm with $k = 3$, for both SNDLIB and TSPLIB graphs, and using all the inequalities and the reduction operations, that is Run 1. The results are given in Tables 3.1 and 3.2.

We first observe that all the SNDLIB instances of Table 3.1 have been solved to optimality within the CPU time limit. For TSPLIB graphs, all the instances have been solved to optimality, except one, u_159. The CPU time for the instances solved to optimality is less than 45min for SNDLIB instances and less than 2h50min for TSPLIB instances. We also observe that the gap between the optimal solution and the lower bound achieved at the root node of the Branch-and-Cut tree is less than 1% for all the SNDLIB instances except one, geant_22, for which the gap is 1.07%. For TSPLIB graphs, the gap is less than 1%, except for 6 instances for which the gap is less than 7.1%. We can also notice that the number of nodes in the Branch-and-Cut

Instance	#EC	#NC	#SPC	#FNPC	#NPC	COpt	Gap1	NSub1	CPU1
atlanta_15	15	606	1	17	1	3265	0.01	3	0:00:01
geant_22	72	1990	19	28	6	375	1.07	60	0:00:26
france_25	80	7500	15	36	7	3254	0.08	37	0:00:32
norway_27	68	4448	10	55	5	5730	0.76	15	0:00:43
sun_27	42	2582	8	28	0	4771	0.04	7	0:00:31
india_35	62	2231	5	26	6	452	0.33	8	0:00:53
cost266_37	135	10726	30	775	7	275	0.9	13	0:18:01
giul_39	62	2760	7	32	1	5878	0.03	5	0:02:19
pioro_40	11	2866	0	2	0	5637	0.00	1	0:00:09
germany_50	42	13094	5	14	2	112	0.01	4	0:02:37
ta2_65	124	7597	10	106	4	5334	0.07	9	0:43:55

Table 3.1: Results for SNDLIB instances with $k = 3$.

Instance	#EC	#NC	#SPC	#FNPC	#NPC	COpt	Gap1	NSub1	CPU1
bays_29	74	3709	11	39	8	14815	1.01	19	0:01:10
dantzig_42	137	9156	12	32	16	1232	0.03	42	0:14:14
att_48	138	13995	14	47	10	17527	0.02	48	0:42:11
eil_51	55	4680	7	30	1	745	0.02	4	0:06:41
berlin_52	133	9518	26	95	10	12644	0.22	30	0:27:05
eil_76	80	15321	8	84	4	947	0.11	8	0:48:05
gr_96	174	330	19	6	0	915	0.6	2	2:03:11
rat_99	112	294	9	19	0	2105	0.3	32	2:02:26
kroA_100	169	305	24	13	1	36492	0.21	2	2:04:35
rd_100	186	303	21	3	0	13391	0.13	21	2:01:03
kroB_100	145	300	12	52	1	37341	1.6	12	2:03:58
lin_105	214	317	15	6	6	24870	2.4	35	2:01:44
gr_120	90	332	10	0	0	11562	0.6	2	2:26:57
bier_127	136	364	16	2	0	199863	3.2	23	2:42:41
pr_124	179	403	12	0	0	99696	0.29	3	2:28:01
ch_130	122	371	10	0	0	10571	7.1	12	2:48:25
kroA_150	130	415	1	2	0	44952	2.6	23	2:49:56
*u_159	112	429	3	7	0	71772	*8.9*	59	5:00:00

Table 3.2: Results for TSPLIB instances with $k = 3$.

tree is quite small, less than 60 nodes, for all the instances. Our separation procedures have also detected several inequalities of each type (cut, node-cut, F-node-partition, SP-node-partition and node-partition inequalities), especially the cut and node-cut inequalities. Moreover, a large number of F-node-partition and SP-node-partition inequalities are generated while few node-partition inequalities have been generated. From these observations, we conclude that our Branch-and-Cut algorithm is efficient for solving the kNCSP with $k = 3$.

We have also run the algorithm, during Run 1, with $k = 4$ and $k = 5$. The results

for $k = 4$ are given by Table 3.3, for SNDLIB instances, and by Table 3.4 for TSPLIB instances. Note that when k is even, the SP-node-partition and partition inequalities we have considered in our algorithm are redundant with respect to the cut inequalities. Thus, they are not used in the Branch-and-Cut algorithm for $k = 4$ and do not appear in Tables 3.3 and 3.4.

Instance	#EC	#NC	#FNPC	COpt	Gap1	NSub1	CPU1
atlanta_15	0	246	2	4615	0.00	1	0:00:01
geant_22	0	912	0	521	0.00	1	0:00:01
france_25	0	594	0	4692	0.00	1	0:00:01
norway_27	0	793	4	8257	0.00	1	0:00:03
sun_27	0	696	0	6867	0.00	1	0:00:01
india_35	4	1324	2	640	0.00	1	0:00:08
cost266_37	0	1326	0	392	0.00	1	0:00:03
giul_39	0	1602	2	8314	0.00	1	0:00:07
pioro_40	0	1711	3	8137	0.00	1	0:00:15
germany_50	0	2610	0	156	0.00	1	0:00:12
ta2_65	0	4417	4	7631	0.00	1	0:02:01

Table 3.3: Results for SNDLIB instances with $k = 4$.

Instance	#EC	#NC	#FNPC	COpt	Gap1	NSub1	CPU1
bays_29	4	897	0	20945	0.00	1	0:00:01
dantzig_42	10	1858	9	1776	0.00	1	0:00:11
att_48	20	2458	5	17380	0.00	1	0:01:15
eil_51	0	2544	0	1051	0.00	1	0:00:12
berlin_52	6	2860	2	18351	0.00	1	0:00:54
eil_76	0	5981	2	1350	0.00	1	0:03:24
gr_96	62	438	76	1314	1.6	6	2:00:03
rat_99	29	10135	14	3045	0.00	1	0:49:33
kroA_100	22	15223	6	53111	0.00	1	0:32:25
rd_100	81	451	32	20341	1.9	4	2:02:38
kroB_100	95	5012	31	55182	2.5	8	0:45:09
lin_105	33	11339	4	36430	0.00	1	0:36:58
gr_120	6	14765	6	18714	0.00	1	0:48:40
pr_124	69	7553	16	144715	3.5	2	2:12:59
bier_127	30	16447	0	283154	0.00	1	0:34:22
ch_130	26	532	10	15123	2.3	23	2:06:45
kroA_150	19	595	4	68281	5.3	22	2:24:45
u_159	13	630	6	104664	7.2	15	4:38:14

Table 3.4: Results for TSPLIB instances with $k = 4$.

We can first observe that for $k = 4$ all the SNDLIB instances are solved to optimality, in less than 2min, and this, at the root node of the Branch-and-Cut tree. For TSPLIB instances, the problem is solved in less than 2h30 for all the instances with few nodes

(less than 23) in the Branch-and-Cut tree. For these latter instances, 11 of them over 17 instances are solved at the root node of the Branch-and-Cut tree. As for $k = 3$, several cut and node-cut inequalities have been generated, and few F-node-partition inequalities are generated. The comparison with $k = 3$ shows that the problem seems easier when $k = 4$, since the optimal solutions are obtained faster when $k = 4$ for all the instances. For example, ta_65 is solved in 43min55sec with 9 nodes in the Branch-and-Cut tree when $k = 3$, while it is solved in 2min at the root node of the Branch-and-Cut tree when $k = 4$. Moreover, instance u_159 is solved to optimality in 4h38min14sec when $k = 4$, whereas it is not solved to optimality within 5h when $k = 3$.

We have also run our algorithm for $k = 5$. The results are given in Tables 3.5 and 3.6.

Instance	#EC	#NC	#SPC	#FNPC	#NPC	COpt	Gap1	NSub1	CPU1
atlanta_15	14	326	0	12	0	6239	0.00	1	0:00:01
geant_22	2	1193	0	0	2	717	0.35	12	0:00:04
france_25	25	832	0	40	1	6478	0.00	1	0:00:20
norway_27	27	944	0	47	0	11217	0.00	1	0:00:51
india_35	30	2023	0	21	0	864	0.17	1	0:00:51
sun_27	27	1296	0	46	0	9383	0.34	4	0:00:32
cost266_37	37	1677	0	149	1	527	0.19	4	0:04:38
giul_39	39	1838	0	75	0	11264	0.00	1	0:03:41
pioro_40	0	1728	0	4	0	10952	0.00	1	0:00:12
germany_50	9	2651	0	4	0	206	0.00	1	0:00:31
ta2_65	67	4723	0	103	0	10276	0.00	1	0:45:20

Table 3.5: Results for SNDLIB instances with $k = 5$

Here also, we can see that several instances are solved to optimality at the root node of the Branch-and-Cut tree for both SNDLIB and TSPLIB instances. The comparison with $k = 3$ also shows that the problem seems easier when $k = 5$. Indeed, for SNDLIB graphs, 8 instances over 11 have been solved at the root node of the Branch-and-Cut tree when $k = 5$ whereas only one instance has been solved at the root node for $k = 3$. The observation is the same for TSPLIB instances. Here, 5 instances have been solved at the root node when $k = 5$ whereas no instance has been solved at the root node when $k = 3$. Also the CPU time is, in general, better when $k = 5$. For example, instance gr_120 is solved in 2h26min57sec when $k = 3$ and in 1h09min03sec when $k = 5$. All these observations suggest that the kNCSP becomes easier when k increases.

A comparison between the case $k = 4$ and $k = 5$ shows that the problem seems easier when $k = 4$. Indeed, the CPU time is in general better when $k = 4$ and fewer nodes are generated in the Branch-and-Cut tree when $k = 4$. We can say from this

Instance	#EC	#NC	#SPC	#FNPC	#NPC	COpt	Gap1	NSub1	CPU1
bays_29	30	1066	0	64	0	28504	0.00	1	0:01:09
dantzig_42	25	1989	1	4	2	1931	0.00	1	0:00:36
att_48	59	2597	0	25	4	17945	0.21	19	0:05:10
eil_51	51	3012	0	187	0	1435	0.21	6	0:27:27
berlin_52	28	5401	0	17	4	24913	0.05	1	0:04:41
eil_76	0	6030	0	3	0	1792	0.00	1	0:04:27
gr_96	48	14203	3	32	4	1792	0.16	6	1:38:50
rat_99	62	561	0	37	0	4113	1.2	4	2:01:15
kroA_100	61	10496	0	82	4	72119	0.14	9	2:03:22
rd_100	73	10485	0	60	8	27273	0.07	9	2:02:07
kroB_100	44	581	0	55	6	75143	0.31	8	2:01:37
lin_105	35	588	0	16	0	50669	0.6	12	2:00:50
gr_120	0	14814	0	9	2	23135	0.00	1	1:09:03
pr_124	45	629	0	15	2	199713	3.4	2	2:03:39
bier_127	65	763	0	14	0	391092	4.1	2	2:01:48
ch_130	8	595	0	12	0	21618	3.8	2	2:09:41
kroA_150	13	681	0	8	0	88237	2.6	25	2:22:03
*u_159	5	630	0	2	0	75915	9.3	91	5:00:00

Table 3.6: Results for TSPLIB instances with $k = 5$

that the problem is harder when k is odd than when k is even. The remarks made here are similar to those made by Bendali et al. [9] for the kECSP. They also concluded from their experiments that the kECSP is harder when k is odd, and that the kECSP becomes easier when k increases with the same parity.

The next series of experiments concerns the efficiency of the reduction operations $\theta_1, ..., \theta_4$. For this, we have run the Branch-and-Cut algorithm with $k = 3$ and without the reduction operations (Run 2).The results are given by Table 3.7.

Instance	Gap1	Gap2	NSub1	NSub2	CPU1	CPU2
atlanta_15	0.01	0.02	3	15	00:00:01	00:00:32
geant_22	1.07	1.94	60	77	00:00:26	00:01:21
france_25	0.08	0.09	37	57	00:00:32	00:01:48
norway_27	0.76	1.78	15	34	00:00:43	00:02:05
india_35	0.33	2.05	8	24	00:00:53	00:02:08
giul_39	0.03	1.4	5	11	00:02:19	00:15:34
ta2_65	0.07	1.7	9	35	00:43:55	02:42:37
dantzig_42	0.03	0.79	42	74	00:14:14	01:37:37
att_48	0.02	2.4	48	68	00:42:11	02:39:38
eil_76	0.11	3.4	8	53	00:48:05	05:00:00
gr_96	0.6	7.8	2	41	02:03:11	03:24:57

Table 3.7: Comparison of results for $k = 3$ with and without the reduction operations.

We can observe from Table 3.7 that, for the considered instances, the performances of the Branch-and-Cut algorithm are decreased when the reduction operations are not used. One can see that both the CPU time and the number of nodes in the Branch-and-Cut tree are increased when the reduction operations are not used in the algorithm. Also, the gap increases for all the instances, which indicates that a fewer number of inequalities or less efficient inequalities are generated during the separation phases. Moreover, one instance, eil_76 which is solved to optimality at Run 1 is not solved to optimality within 5h without the reduction operations. This clearly proves the efficiency of the reduction operations on the resolution process.

Our last series of experiments aims to check the efficiency of the F-node-partition, SP-node-partition and node-partition inequalities in solving the kNCSP. For this, we have run the Branch-and-Cut algorithm in Run 3 with only the cut and node-cut inequalities. The results are presented in Table 3.8.

Instance	Gap1	Gap3	NSub1	NSub3	CPU1	CPU3
atlanta_15	0.01	0.03	3	9	00:00:01	00:00:45
geant_22	1.07	2.1	60	84	00:00:26	00:02:09
france_25	0.08	0.17	37	43	00:00:32	00:02:33
norway_27	0.76	1.97	15	60	00:00:43	00:02:51
india_35	0.33	1.71	8	17	00:00:53	00:04:14
giul_39	0.03	1.8	5	14	00:02:19	00:17:57
ta2_65	0.07	1.2	9	21	00:43:55	01:18:37
dantzig_42	0.03	1.64	42	57	00:14:14	00:45:52
att_48	0.02	1.9	48	71	00:42:11	01:34:59
eil_76	0.11	6.5	8	64	00:48:05	02:54:35
gr_96	0.6	13.2	2	34	02:03:11	05:00:00

Table 3.8: Comparison of results for $k = 3$ with and without the F-node-partition, SP-node-partition and node-partition inequalities.

Here also, the comparison between Run 1 and Run 3 shows that the performances are decreased when F-node-partition, SP-node-partition and node-partition inequalities are not used in the algorithm. The CPU time, the number of nodes in the Branch-and-Cut tree and the gap are increased for all the instances of Table 3.8. Also, instance gr_96 is not solved to optimality within 5h when F-node-partition, SP-node-partition and node-partition inequalities are not used, while it is in Run 1. This also shows the efficiency of the above inequalities in solving the kNCSP.

We conclude this computational study by comparing the optimal solutions obtained here for the kNCSP with those of the kECSP obtained by Bendali et al. [9]. The aim is to know how often optimal solutions of the kECSP and kNCSP are equal. The next

table, Table 3.9, presents, for some TSPLIB instances, the optimal solutions of the kECSP, those of the kNCSP for $k = 3$ and the gap between the two solutions, given by

$$Gap = \frac{COpt\_3\text{NCSP} - COpt\_3\text{ECSP}}{COpt\_3\text{ECSP}}.$$

Instance	COpt_3ECSP	COpt_3NCSP	Gap
dantzig_42	1210	1232	1.82
att_48	17499	17527	0.16
berlin_52	12601	12644	0.34
eil_76	876	947	8.11
rat_99	2029	2105	3.75
kroA_100	36337	36492	0.43
kroB_100	37179	37341	0.44
rd_100	13284	13391	0.81
gr_120	11442	11562	1.05
bier_127	198184	199863	0.85
ch_130	10400	10571	1.64
kroA_150	44718	44952	0.52

Table 3.9: Comparison between of the best solutions of kECSP and the kNCSP for $k = 3$.

From Table 3.9, we can see that the optimal solutions of the two problems are different for all the considered instances. However, we can see that the gap between the two solutions is relatively small for most of them. This let us conclude that the best solutions obtained by Bendali et al. [9] for the 3ECSP are good upper bounds of the optimal solutions of the 3NCSP. Clearly, this remark cannot be generalized since we may find graphs for which the gap between the optimal solutions of the kNCSP and the kECSP is more important, but solving the kECSP could produce a good approximation of the kNCSP.

3.3 Conclusion

We have studied the k-node-connected subgraph problem with high connectivity requirement, that is, when $k \geq 3$. We have presented some classes of valid inequalities and described some conditions for these inequalities to be facet defining for the associated polytope. We have also investigated the structural properties of the extreme points of the linear relaxation of the problem and presented some reduction operations. Using these results, we have devised a Branch-and-Cut algorithm for the problem. The

computational results we have obtained have shown that the F-node-partition, SP-node-partition and partition inequalities are effective for solving the problem. Also, the reduction operations we have used are shown to be efficient in the separation phase of the Branch-and-Cut algorithm. The experiments also show that, as for the kECSP, the kNCSP becomes easier when k increases, and is harder when k is odd than when k is even.

The study presented in this chapter shows the efficiency of some valid inequalities, namely F-node-partition, SP-node-partition and partition inequalities, in solving the kNCSP. It would be interesting to investigate the polytope of the problem in a deeper way and identify cases in which these inequalities completely define the polytope of the problem.

Chapter 4

Branch-and-Cut-and-Price Algorithm for the 2NCSP

Contents

In this chapter we present a Branch-and-Cut-and-Price algorithm we have devised to solve the 2NCSP. In Section 4.1 we will present an extended formulation for the problem. In sections 4.2.1 and 4.2 we will describe the framework of the column generation algorithm. In Section 4.3, we will present some computational results and Section 4.4 will be devoted to some concluding remarks.

4.1 Extended formulation

We give a cycle-based integer programming formulation for the 2-node-connected subgraph problem. We consider an undirected graph $G = (V, E)$, and let $w(e)$ be the weight of the edge e, for all $e \in E$. We suppose that $w(e) > 0$, for all $e \in E$.

First, we observe that in a solution of the problem, each two nodes of the graph belong to a cycle. Thus any solution of the problem is composed of a collection of cycles.

Let θ_{uv} be the set of simple cycles of G going through nodes u and v, for all $u, v \in V$, and let $\theta = \bigcup_{u,v \in V, u \neq v} \theta_{u,v}$. Also for a cycle C and two nodes $u, v \in V$, let

$$a_C(u,v) = \begin{cases} 1 & \text{if the cycle } C \text{ goes through the nodes } u \text{ and } v, \\ 0 & \text{otherwise.} \end{cases}$$

for all $u, v \in V$, and

$$b_C(e) = \begin{cases} 1 & \text{if the cycle } C \text{ uses edge } e \\ 0 & \text{otherwise.} \end{cases}$$

for all $e \in E$.

Now let $y(C)$ be a $0-1$ variable whose value is 1 if the cycle $C \in \theta$ is taken in the solution and 0 otherwise, and $x \in \{0,1\}^E$ such that $x(e) = 1$ if edge e is taken in the solution and 0 otherwise. Consider the following inequalities which are satisfied by every solution of the 2NCSP.

$$\sum_{C \in \theta} a_C(u,v) y(C) \geq 1 \qquad \text{for all } u, v \in V, \tag{4.1}$$

$$\sum_{C \in \theta} b_C(e) y(C) \leq M x(e) \qquad \text{for all } e \in E, \tag{4.2}$$

$$y(C), x(e) \in \{0,1\} \qquad \text{for all } e \in E \text{ and } c \in \theta. \tag{4.3}$$

Here M is an upper bound on the number of cycles going through an edge in a solution and can be chosen to be $M = (|V|-2)!$.

Inequalities (4.1) ensure that between each two nodes u and v in V there exists a cycle going through u and v, and inequalities (4.2) ensure that if an edge is not taken in the solution, all the cycles containing the edge are not considered.

Moreover, the following is easily seen.

Theorem 12 *The 2NCSP is equivalent to the following integer program*

$$\min\{wx \mid x \text{ satisfies } (4.1) - (4.3) \text{ and } x \in \mathbb{Z}_+^E\}. \tag{4.4}$$

Formulation (4.4) uses an exponential number of variables but a polynomial number of contraints. We will use a column generation algorithm to solve its linear relaxation.

4.2 Branch-and-Cut-and-Price algorithm

4.2.1 Column generation algorithm

A column generation algorithm for the linear relaxation of (4.4) starts by solving a linear program obtained from (4.4) by considering a subset of variables (columns) which induce a feasible basis for the initial problem. For our purpose, we consider first sets of hamiltonian cycles $C_H \subset \theta$. Note that the subgraph induced by the cycles of C_H contains 2 node-disjoint paths between every pair of nodes of V. Thus, the edge set corresponding to the cycles induces a solution of the 2NCSP, and, together with the sets C_H, induces a feasible solution for the linear relaxation of Formulation (4.4). Hence, we consider as initial set of variables those induced by the edge set corresponding to the cycles C_H and the sets C_H. The first linear program solved in the column generation algorithm is, therefore, the one obtained from the linear relaxation of Formulation (4.4) and these variables. This linear program is

$$Min \sum_{e \in E} w(e)x(e)$$

$$\sum_{C \in C_H} a_C(u,v)y(C) \geq 1 \qquad \text{for all } u, v \in V, \tag{4.5}$$

$$\sum_{C \in C_H} b_C(e)y(C) \leq Mx(e) \qquad \text{for all } e \in E, \tag{4.6}$$

$$y(C) \geq 0 \qquad \text{for every } C \in C_H, \tag{4.7}$$

$$x(e) \leq 1 \qquad \text{for all } e \in E. \tag{4.8}$$

At each iteration, if the current solution is not optimal, the algorithm tries to generate new columns, that is to add to C_H, cycles $C \in \theta \setminus C_H$ such that the variable $y(C)$ has a negative reduced cost. This is done by solving the so-called *pricing problem* which consists here in finding, a cycle C^* such that $C_r(C^*) = \min\{c_r(C) \mid C \in \theta\}$ and $c_r(C^*) < 0$, where $c_r(C)$ is the reduced cost of the variable $y(C)$.

The reduced cost $c_r(C)$ is computed using the dual optimal solution. Let α_{uv}, $u, v \in V$, be the dual variables associated with inequalities (4.5), β_e the dual variable associated with each inequality (4.6), for all $e \in E$, and γ_e the dual variables associated with inequalities (4.8).

The dual problem is

$$Max \sum_{u,v \in V} \alpha_{uv}$$

$$\sum_{u,v \in V} a_C(u,v)\alpha_{uv} - \sum_{e \in E} b_C(e)\beta_e \leq 0 \qquad \text{for all } C \in \theta, \tag{4.9}$$

$$M\beta_e - \gamma_e \leq w(e) \qquad \text{for all } e \in E, \tag{4.10}$$

$$\alpha_{uv} \geq 0, \beta_e \geq 0 \qquad \text{for all } u, v \in V \text{ and } e \in E. \tag{4.11}$$

Then, given a cycle $C \in \theta$, the reduced cost of the variable $y(C)$ is given by

$$c_r(C) = \sum_{u,v \in V} a_C(u,v)\alpha_{uv} - \sum_{e \in E} b_C(e)\beta_e. \tag{4.12}$$

If $c_r(C) < 0$ then

$$\sum_{u,v \in V(C)} \alpha_{uv} - \sum_{e \in E(C)} \beta_e > 0. \tag{4.13}$$

where $V(C)$ and $E(C)$ denotes respectively the set of nodes and set of edges of the cycle C.

Thus, the pricing problem reduces to find a longest cycle in the graph G, with respect to lengths α_{uv}, on the pairs $(u,v) \in V \times V$ and $-\beta_e$ on the edges $e \in E$. If a longest cycle of G, say C^*, is such that $\sum_{u,v \in V(C)} \alpha_{uv} < \sum_{e \in E(C)} \beta_e$, then $c_r(C^*) < 0$. If not, then $c_r(C) \geq 0$ for every cycle $C \in \theta$.

The longest cycle between each two nodes u and v in V is computed using the compact formulation of Dixon and Goodman [32]. The formulation is an integer linear program in bounded variables. The ILP model will find the longest cycle through a given node, say node 1, and application to each of the $|V|$ nodes in the graph will get the longest among all the cycles. The variables are

$$X_{ij} = \begin{cases} 1 & \text{if edge } (i,j) \text{ is used in the longest cycle,} \\ 0 & \text{otherwise.} \end{cases}$$

$$F_i = \begin{cases} 1 & \text{if node } i \text{ is used in the longest cycle,} \\ 0 & \text{otherwise.} \end{cases}$$

The integer programming model is then

$$max\ z = \sum_{i,j \in V} \alpha_{ij} F_i F_j - \sum_{(i,j) \in E} \beta_{(i,j)} X_{ij}$$

subject to the constraints

$$\sum_{i=1}^{|V|+1} X_{ik} = F_k \qquad k = 2, 3, ..., |V| \tag{4.14}$$

$$\sum_{j=1}^{|V|+1} X_{kj} = F_k \qquad k = 2, 3, ..., |V| \tag{4.15}$$

$$\sum_{j=2}^{|V|} X_{1j} = \sum_{i=1}^{|V|} X_{i|V|+1} \tag{4.16}$$

$$Y_i - Y_j + (|V| + 1)X_{ij} \leq |V| \qquad i, j = 1, 2, ..., |V| + 1 \tag{4.17}$$

$$0 \leq Y_i \leq |V| + 1 \qquad Y_i \text{ integer} \tag{4.18}$$

The node $|V|+1$ is introduced by splitting node 1 into nodes 1 and $N+1$, each with the same adjacencies. The constraints (4.14)-(4.18) specify a simple path from node 1 to node $|V|+1$. Constraints (4.17) and (4.18) introduce dummy integer variables, one Y_i for each node $i \in V$, which do not permit the development of disjoint side cycles. This assures us that all feasible solutions will contain exactly one path from 1 to $|V|+1$.

Now if $c_r(C) \geq 0$ for all $C \in \theta$, then the optimal solution of the current linear program is optimal for the linear relaxation of Formulation (4.4).

The optimal solution of the linear relaxation of Formulation (4.4) is feasible for Formulation (4.4) if it is integral. If this is not the case, then we add further valid inequalities for 2NCSP(G) that are violated by this solution. The inequalities that are considered are the cut inequalities (2.3), and the F-partition inequalities (2.12).

To obtain initial interesting feasible solutions we have implemented several heuristics that generate hamiltonian cycles. The first heuristic is quite simple. We initialize the cycle by randomly choosing a node. Then we add a minimum weight edge, not selected before, among its incidence edge list. We continue adding the edges in the same way until we complete the cycle.

The second is a Greedy heuristic. It gradually constructs a cycle by repeatedly selecting an edge having the smallest weight, and adding it to the cycle as long as it doesn't create a cycle with less than $|V|$ edges, or increases the degree of any node to more than 2. We must not add the same edge twice of course. The steps are:
1. Sort all edges.
2. Select an edge with the smallest weight and add it to the cycle if it doesn't violate

any of the above constraints.
3. If we do not have $|V|$ edges in the cycle, repeat step 2.

4.2.2 Pricing heuristics

The problem of finding the longest simple cycle or path in an edge-weighted network is NP-complete. And the exact pricing in our problem requires $|V|$ calls to the compact formulation in [32]. Therefore we also needed pricing heuristics we used before calling the exact pricing.

In the first pricing heuristic we first add a maximum weight edge. We then construct the cycle by adding the edge that maximizes the cycle weight, among the remaining edges. Once we obtain a cycle, we remove all the isolated branches.

The second heuristic is based on a Tabu Search. The fundamental concept in any neighborhood search is the notion of a neighborhood. The neighborhood N of a solution s (noted $N(s)$) is a set of solutions where each $s^* \in N(p)$ is reachable by a simple *move* from s. The definition of the move is arbitrary, in our heuristic a move scheme is either a swap of two adjacent nodes in the path, deleting a node from the path, or adding a node to the path. The move is chosen randomly with equal probabilities. In a naive neighborhood search, the algorithm starts with a solution s and then evaluates all solutions in $N(s)$. Then it chooses the best $s_b \in N(s)$, sets $s = s_b$, and iterates. The problem with a naive neighborhood search is that it can ascend to local maxima and returns a suboptimal solution. The main idea is that certain moves within the neighborhood are classified as tabu under certain circumstances. If a move is tabu, the solution generated by that move cannot usually be considered the best in the neighborhood. The exception to the rule is that if a tabu move would result in a solution which is better than any of the solutions that have been seen to that point. If such a solution exists, the algorithm will then choose that solution even though it is tabu. Finally, we define stopping conditions for the main loop. We set the maximum number of iterations at max_It=1000. The algorithm will terminate if this number of iterations is reached. The algorithm also stops if there is no improvement in max_It/5 iterations. The whole pricing procedure is summarized in Algorithm 4 below.

Algorithm 4: General framework for the Branch and Cut and Price algorithm

```
Data: An undirected graph G = (N, E), a fractional solution ȳ
Result: An integer solution for the 2NCSP
begin
    /*Compute the heuristics to initial feasible solutions*/
    compute_Initial_Cycles_Greedy();
    compute_Initial_Hamiltonian_Cycles();

    /*Check if the resulting graph is 2-node-connected*/
    FeasibleSolutionFound ← False;
    repeat
        Call the separation procedure for cut and F-partition inequalities;
        if there is a cut or an F-partition inequality violated by z then
            add the cut to the PL;
        end
        else
            Call the heuristic pricing procedure;
            compute_Longest_Cycle_Heuristic_Tabu();
            compute_maximum_weight_edge_Heuristic();
            if there is a violated cycle then
                add the cycle variable;
            end
            else
                Call the exact pricing procedure;
                compute_Longest_Cycle_Exact();
                if there is a violated cycle then
                    add the cycle variable;
                    BranchingIsDone ← True;
                end
                else
                    FeasibleSolutionFound ← True;
                end
            end
        end
    until FeasibleSolutionFound = True;
    return FeasibleSolution;
end
```

4.3 Computational results

The Branch-and-Cut-and-Price algorithm described in the previous section has been implemented in C++, using SCIP 3.2.1 [4] to manage the Branch-and-Price, and CPLEX 12.5 as LP-solver for the exact pricing. All experiments were run on a com-

puter equiped with a 2.10 GHz x4 Intel Core(TM) i7-4600U processor and running under linux with 16 GB of RAM. The maximum CPU time has been fixed to 5 hours. The test problems we have considered are complete euclidian graphs from SNDLIB library [1].

The entries of the tables presented below are:

$\|V\|$	:	the number of nodes of the graph,
Gen_Cuts	:	the number of generated cut and F-partition inequalities,
Gen_Cols	:	the number of generated columns,
COpt	:	value of the best upper bound obtained,
Gap	:	the relative error between the best upper bound and the lower bound obtained at the root node of the tree,
NSub	:	the number of nodes in the tree,
CPU	:	total CPU time in hours:min:sec.

Our first series of experiments concerns the SNDLIB graphs. The results are summurized in Table 4.1. For the SNDLIB graphs, with 11 up to 65 nodes, all the instances have been solved to optimality within the time limit. The CPU time for these instances, except the last one, is less than one hour. Some of the instances of the table didn't require branching, for example diyuan (11 nodes), newyork (16 nodes) and sun (27 nodes). For all the instances the relative error between the lower bound at the root node and the best upper bound (Gap) is less than 1 %. We can also see that the column generation algorithm added a large number of variables and our separation procedures have detected an important number of cut inequalities and F-partition inequalities.

Our second series of experiments concerns the TSPLIB graphs given in Table 4.2. The instances we have considered have graphs with 14 up to 137 nodes. We observe that 12 instances over 13 have been solved to optimality within the time limit. For the last instance the algorithm has not been able to finish the resolution within 5 hours. A large number of variables were added by the column generation algorithm, and an important number of cut inequalities and F-partition inequalities have been generated. The gap between the lower bound at the root node of the Branch-and-Cut tree and the best upper bound is less than 0.01 for almost all the instances, except for pr76.

In the rest of this section, we focus on the comparison between the results obtained by the column generation algorithm and the Branch-and-Cut algorithm presented in Chapter 3 for $k = 2$, in terms of CPU time, Gap, number of nodes in the Branch-and-Cut tree and in terms of best solution.

Table 4.1: Results for the SNDLIB instances.

Instance	$\|V\|$	Gen_Cuts	Gen_Cols	COpt	Gap	NSub	CPU
diyuan	11	2	0	989	0.00	1	0:00:02
pdh_12	11	19	0	18	0.00	1	0:00:03
abilene	12	19	24	121	0.04	15	0:00:18
atlanta	12	17	28	115	0.00	1	0:00:03
polska	12	26	4	27	0.05	11	0:00:01
nobel-us	14	32	0	117	0.00	5	0:00:02
newyork	16	24	56	2335	0.00	1	0:00:04
geant	22	58	80	226	0.02	15	0:00:20
ta1	24	45	27	1859	0.02	27	0:00:31
france	25	52	43	1985	0.01	7	0:00:48
janos-us	26	64	626	178	0.06	25	0:00:21
sun	27	50	22	2849	0.00	1	0:00:08
norway	27	91	1068	3847	0.08	139	0:01:19
janos-us-ca	37	85	14	201	0.06	27	0:00:24
cost266	37	118	7876	173	0.07	103	0:12:59
giul39	39	149	6576	3949	0.05	147	0:09:46
pioro	40	81	2168	3625	0.03	11	0:01:44
germany	50	67	4948	56	0.1	33	0:52:34
ta2	65	148	2439	3455	0.04	3	1:06:10

Table 4.2: Results for the TSPLIB instances.

Instance	$\|V\|$	Gen_Cuts	Gen_Cols	COpt	Gap	NSub	CPU
burma14	14	38	124	3841	0.06	29	0:00:08
ulysses22	22	56	868	8180	0.07	47	0:00:31
fri26	26	67	65	965	0.01	23	0:01:33
bays	29	84	1498	9884	0.05	55	0:03:41
dantzig42	42	82	1	699	0.00	1	0:00:20
eil51	51	114	6847	482	0.06	19	4:59:01
berlin52	52	141	127	8181	0.04	7	0:05:48
st70	70	191	2970	750	0.05	3	4:51:05
pr76	76	240	2595	130921	0.1	5	4:53:32
kroC100	100	376	2135	23295	0.05	46	2:45:27
lin105	105	420	8771	16766	0.07	36	4:41:23
ch130	130	331	3722	7129	0.07	48	4:55:44
gr137	137	363	5579	79460	0.06	25	5:00:00

The entries of the following tables are:

$\|V\|$	:	the number of nodes of the graph,
COpt_BC	:	value of the best upper bound obtained by the Branch-and-Cut,
COpt_BCP	:	value of the best upper bound obtained by the Branch-and-Cut-and-Price,
Gap_BC	:	the gap for the Branch-and-Cut,
Gap_BCP	:	the gap for the Branch-and-Cut-and-Price,
NSub_BC	:	the number of nodes in the tree for the Branch-and-Cut,
NSub_BCP	:	the number of nodes in the tree for the Branch-and-Cut-and-Price,
CPU_BC	:	total CPU time for the Branch-and-Cut,
CPU_BCP	:	total CPU time for the Branch-and-Cut-and-Price.

First, in Table 4.3 we compare the two formulations for the SNDLIB instances. We observe that for these instances the CPU time for the column generation algorithm is greater than for the Branch-and-Cut algorithm. We also observe that the number of instances solved to optimality is quite the same for the two formulations, and the gaps are generally close. We can also see that most of the instances required more branching in the column generation algorithm than in the Branch-and-Cut algorithm.

Table 4.3: Branch-and-Cut and Branch-and-Cut-and-Price comparison for SNDLIB instances.

Instance	$\|V\|$	COpt_BC	COpt_BCP	Gap_BC	Gap_BCP	NSub_BC	NSub_BCP	CPU_BC	CPU_BCP
atlanta	12	120	121	0.00	0.4	12	15	0:00:01	0:00:18
polska	12	27	27	4.17	0.05	12	11	0:00:01	0:00:01
nobel-us	14	117	117	0.00	0.00	14	5	0:00:01	0:00:02
newyork	16	2450	2543	0.00	0.04	16	17	0:00:01	0:00:12
geant	22	222	226	0.00	0.02	22	15	0:00:02	0:00:20
ta1	24	1850	1859	0.00	0.02	24	27	0:00:01	0:00:31
france	25	1960	1985	0.00	0.01	25	7	0:00:13	0:00:48
janos-us	26	178	178	0.00	0.06	26	25	0:00:02	0:00:21
norway	27	3720	3847	0.00	0.04	27	25	0:00:02	0:01:02
sun	27	2849	2849	0.88	0.00	27	1	0:00:01	0:00:08
india	35	270	184	0.00	0.06	35	117	0:00:21	0:09:35
cost266	37	165	173	0.00	0.07	37	103	0:00:04	0:12:59
janos-us-ca	37	201	201	0.00	0.06	37	27	0:00:06	0:00:24
giul39	39	3590	3949	0.6	0.05	39	147	0:00:34	0:09:46
pioro	40	3555	3625	0.00	0.03	40	11	0:01:19	0:01:44
ta2	65	3320	3455	0.09	0.04	65	3	0:12:35	1:06:10

Then, in Table 4.4 we compare the two formulations for the TSPLIB graphs. We observe that the efficiency of the different algorithms for solving the problem is not

the same. For the instances solved to optimality, the total CPU time for the Branch-and-Cut algorithm is better than for the column generation algorithm, except for the instance kroC100, it was solved in 3h 49mn 24s for the column generation algorithm and in 2h 45mn 27s for the Branch-and-Cut algorithm. The important CPU time for the column generation algorithm can be explained by the fact that the pricing problem used to solve the linear relaxation of the extended formulation, computes a longest cycle formulation, which contains a quadratic objective function. Thus, the difference of CPU time mainly is the time spent by the algorithm for the pricing. However, the gap obtained by the column generation algorithm is in general better than the gap obtained by the Branch-and-Cut algorithm.

Table 4.4: Branch-and-Cut and Branch-and-Cut-and-Price comparison for TSPLIB instances.

Instance	$\|V\|$	COpt_BC	COpt_BCP	Gap_BC	Gap_BCP	NSub_BC	NSub_BCP	CPU_BC	CPU_BCP
burma14	14	3323	3841	0.00	0.06	1	29	0:00:01	0:00:08
ulysses22	22	7013	8180	0.00	0.07	1	47	0:00:01	0:00:31
fri26	26	937	965	0.05	0.01	12	23	0:00:54	0:01:33
bays29	29	2020	9884	0.02	0.05	6	55	0:00:15	0:03:41
dantzig42	42	699	699	0.4	0.00	11	1	0:00:20	0:01:42
eil51	51	426	482	0.00	0.06	1	19	0:02:01	4:59:01
berlin52	52	7542	8181	0.00	0.04	1	7	0:00:16	0:05:48
s70	70	675	750	0.15	0.05	5	3	0:12:12	4:51:05
pr76	76	108159	130921	0.01	0.1	8	5	0:41:30	4:53:32
kroC100	100	20749	23295	4.57	0.05	78	46	3:49:24	2:45:27
lin105	105	14379	16766	0.39	0.07	63	36	2:15:24	4:41:23
ch130	130	6110	7129	0.12	0.07	57	48	2:45:25	4:55:44
gr137	137	69853	79460	7.38	0.06	42	25	5:00:00	5:00:00

4.4 Conclusion

In this chapter, we presented an extended formulation for the k-node-connected subgraph problem. We studied a column generation algorithm and efficient heuristics for the resolution of the problem. And using these results we devised a Branch-and-Cut-and-Price algorithm to solve the problem. The computational results have shown that the column generation algorithm is effective in solving the problem and producing good upper bound for the problem. Also, it has been shown that the Branch-and-Cut algorithm is more efficient in solving the problem to optimality. We also observed that for the column generation algorithm when approaching a near optimal LP value, the

algorithm struggles to find the optimal LP value even though it is close. This phenomenon is known as the tailing off effect. As described in [25], a main explanation for the tailing off effect is that the dual variables converge slowly towards their respective optimal value, and, from one iteration to another, take unrelated and random values. As a future work, we can aim at decreasing this effect, which will improve the efficiency of the resolution by introducing an algorithm that stabilizes and accelerates the solution process while remaining within the linear programming framework. The stabilized algorithm can be used to improve the solution times for difficult instances and to solve larger ones.

One can also try to extend the approach developed in this chapter to study the problem for $k \geq 3$ with an extended formulation using path variables, and devise efficient Branch-and-Price and Branch-and-Cut-and-Price algorithms for the problem in this case.

Chapter 5

The k node-disjoint hop-constrained survivable network problem

Contents

Given a graph with weights on the edges, a set of origin and destination pairs of nodes, and two integers $L \geq 2$ and $k \geq 2$, the k node-disjoint hop-constrained network design problem is to find a minimum weight subgraph of G such that between every origin and destination there exist at least k node-disjoint paths of length at most L.

In this chapter we consider this problem from a polyhedral point of view. We propose an integer linear programming formulation for the problem for $L = 2, 3$ and investigate the associated polytope. We introduce new valid inequalities for the problem, and give

necessary and sufficient conditions for these inequalities to be facet defining. We also devise separation algorithms for these inequalities. Using these results, we propose a Branch-and-Cut algorithm for solving the problem for $k \geq 0$ and $L = 3$, and for $k = 2$ and $L = 4$, along with some computational results.

5.1 Integer Programming Formulation

Let $G = (V, E)$ be a graph, and $F \subseteq E$ an edge set which induces a solution of the kNDHP. As F is a solution of the problem, the subgraph induced by F, say G_F, contains k edge-disjoint st-paths for every $(s, t) \in D$. Thus, by Menger's theorem [60], every st-cut of G_F contains at least k edges. Consequently, the incidence vector of F satisfies the following inequalities

$$x(\delta_G(W)) \geq k, \qquad \text{for all } st\text{-cut } \delta(W) \text{ and } (s,t) \in D. \tag{5.1}$$

Inequalities (5.1) are called *st-cut inequalities*.

Dahl [21] introduces a class of valid inequalities as follows.

Let $(V_0, ..., V_{L+1})$ be a partition of V with $s \in V_0$, $t \in V_{L+1}$, and $V_i \neq \emptyset$ for all $i \in \{1, ...L\}$. Let T be the set of edges $uv \in E$ such that $u \in V_i$, $v \in V_j$ and $|i - j| > 1$, that is,

$$T = \delta(V_0, ..., V_{L+1}) \setminus \bigcup_{i=0}^{L} [V_i, V_{i+1}].$$

The set T is called an *L-st-path cut*. Then the inequality

$$x(T) \geq 1$$

is valid for the *L-st*-path polyhedron. Using similar type of partitions, we can generalize these inequalities to the kNDHP as

$$x(T) \geq k, \text{ for every } L\text{-}st\text{-path-cut } T \text{ of } G, \quad \text{for any } (s,t) \in D. \tag{5.2}$$

Inequalities of type (5.2) are called *L-st-path-cut inequalities*. Figure 5.1 shows an L-st-path-cut.

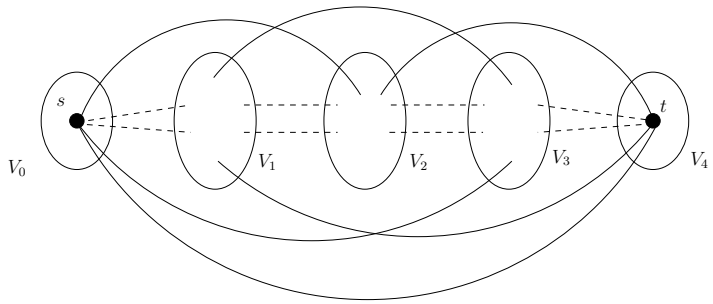

Figure 5.1: Support graph of an L-st-path-cut with $L = 3$ and T formed by the solid edges

Inequalities (5.1) and (5.2) can be easily adapted in order to ensure the existance of k node-disjoint paths of length at most L. Given node subsets $Z \subset V \setminus \{s,t\}$ for $(s,t) \in D$, and $W \subset V \setminus Z$, the *st-node cut* $\delta_{G\setminus Z}(W)$ of G is the st-cut induced by W in $G \setminus Z$. Any L-st-path cut in $G \setminus Z$ is called an *L-st-node path-cut* of G.

A solution $x \in \mathbb{R}^E$ of the kNDHP also satisfies the following inequalities

$$x(\delta_{G\setminus Z}(W)) \geq k - |Z|, \quad \text{for all } st\text{-node-cut } \delta_{G\setminus Z}(W), Z \subset V \setminus \{s,t\} \text{ such that } 1 \leq |Z| \leq k-1, \text{ and } (s,t) \in D, \tag{5.3}$$

$$x(T_{G\setminus Z}) \geq k - |Z|, \quad \text{for all } L\text{-}st\text{-node-path-cut } T_{G\setminus Z} \text{ of } G \setminus Z, Z \subset V \setminus \{s,t\} \text{such that } 1 \leq |Z| \leq k-1, \text{and } (s,t) \in D. \tag{5.4}$$

Inequalities (5.3) and (5.4) are called respectively *st-node-cut* and *L-st-node-path-cut inequalities*. Moreover, the incidence vector of an edge set F inducing a solution of the kNDHP satisfies

$$x(e) \geq 0, \text{ for all } e \in E, \tag{5.5}$$

$$x(e) \leq 1, \text{ for all } e \in E. \tag{5.6}$$

In the following we show that the st-cut, st-node-cut, L-st-path-cut, L-st-node-path-cut, and trivial inequalities, together with integrality constraints, suffice to formulate the kNDHP as a $0-1$ linear program when $L \in \{2,3\}$.

For this, we consider, for each demand (s,t) the directed $\widetilde{G}_{st} = (\widetilde{V}_{st}, \widetilde{A}_{st})$ obtained as follows (see also [10] and [26]). The node set $\widetilde{V}_{st}$ is formed by the nodes s, t, the

node set of $V \setminus \{s,t\}$ and a copy u' for each node $u \in V \setminus \{s,t\}$. The set of arcs $\widetilde{A}_{st}$ is obtained as follows. For each edge $su \in E$ (resp. $ut \in E$) we add in $\widetilde{A}_{st}$ an arc (s,u') (resp. (u',t)). For each edge $uv \in E$, with $u,v \neq s,t$, we add two arcs (u,v') and (v,u') in $\widetilde{A}_{st}$. Finally, for each node $u \in V \setminus \{s,t\}$, we add an arc (u,u') in $\widetilde{A}_{st}$. It is not hard to see that that every st-dipath of $\widetilde{G}_{st}$ corresponds to a 3-st-path of G, and vice-versa. Also, two node-disjoint 3-st-path of G correspond to two directed node-disjoint st-path of $\widetilde{G}_{st}$. However, the converse is not true, that is two node-disjoint st-dipaths of $\widetilde{G}_{st}$ may not correspond to node-disjoint 3-st-paths of G (see Figure 5.2 for illustration).

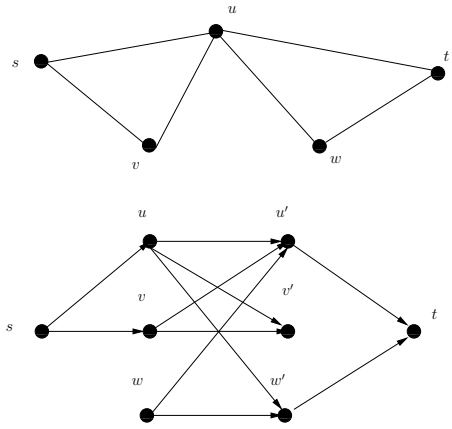

Figure 5.2: Construction of the graph H for $L = 3$.

Bendali et al. [10] show that every st-cut and 3-st-path-cut $C \subseteq E$ can be associated with a directed st-cut $\widetilde{C} \subseteq \widetilde{A}_{st}$ which does not contain an arc of the form (u,u'), with $u \in V \setminus \{s,t\}$, and vice-versa. Moreover, they show that a solution $\overline{x} \in \mathbb{R}^E$ can be associated with a solution $\overline{y} \in \mathbb{R}^{\widetilde{A}_{st}}$ such that $\overline{x}(C) = \overline{y}(\widetilde{C})$.

Now we give the following theorem.

Theorem 13 *Let $\overline{x} \in \{0,1\}^E$ be an integral solution, which satisfies all the cut and 3-st-path-cut inequalities (5.1) and (5.2). Then, $\overline{x}$ induces a solution of the kNHDP if and only if it satisfies all the st-node-cut and 3-st-node-path-cut inequalities.*

Proof. As the st-node-cut and the 3-st-node-cut inequalities are valid for the kNDHP, if $\overline{x}$ is a solution of the kNDHP, then it satisfies these inequalities.

Now suppose that $\overline{x}$ does not induce a feasible solution of the kNHDP, that is the subgraph of G induced by $\overline{x}$, denoted by $G(\overline{x}) = (V, E(\overline{x}))$ does not contain k node-disjoint 3-st-paths for some demand $(s,t) \in D$. We are going to show that there exists

an st-node-cut or a 3-st-node-path-cut inequality which is violated by $\overline{x}$.
Let $\widetilde{G}_{st}$ be the directed graph associated with (s,t) as described above, and let $\widetilde{y} \in \mathbb{R}^{\widetilde{A}_{st}}$ be a weight vector such that

$$\widetilde{y}(a) = \begin{cases} 1 & \text{if } a \text{ corresponds to edge } e \text{ and } e \in E(\overline{x}), \\ 0 & \text{if } a \text{ corresponds to edge } e \text{ and } e \notin E(\overline{x}), \\ +\infty & \text{if } a = (u,u') \text{ for all } u \in V \setminus \{s,t\}. \end{cases}$$

Remark that, as G is simple, that it is does not contain parallel edges, if two 3-st-paths P_1 and P_2 are not node-disjoint, then they are of the form $P_1 = (s,u,v,t)$ and $P_2 = (s,v,z,t)$ with $u,v,z \in V \setminus \{s,t\}$ and $u \neq v \neq z$. These two paths correspond in $\widetilde{G}_{st}$ to paths (s,u,v',t) and (s,v,z',t). Conversely, two paths (s,u,v',t) and (s,v,z',t) of $\widetilde{G}_{st}$ correspond to two paths (s,u,v,t) and (s,v,z,t) which are not node-disjoint. Consequently, when $\overline{x}$ is not feasible for the kNHDP, any maximum set of disjoint st-dipaths of the graph $\widetilde{G}_{st}$, will contain two paths of the form (s,u,v',t) and (s,v,z',t), with $u,v,z \in V \setminus \{s,t\}$ and $u \neq v \neq z \neq u$.

Now we introduce the following procedure, that we call *Procedure BuildZ*, which aims to build a node set $Z \subseteq V$, from which we will obtain the violated st-node-cut or 3-st-node-path-cut inequalities. Let $Z \subseteq V$ be a node set of G and denote by $\widetilde{Z}$ the nodes of $\widetilde{G}_{st}$ corresponding to those of Z, that is $\widetilde{Z} = \{u, u' \text{ such that } u \in Z\}$. At the begining of the procedure $Z = \emptyset$. Now compute a maximum flow from s to t in $\widetilde{G}_{st} \setminus \widetilde{Z}$, with each arc $a \in \widetilde{A}_{st}$ having the capacity $\widetilde{y}(a)$. This gives a maximum set $\widetilde{\mathcal{P}}$ of node-disjoint st-dipaths in $\widetilde{G}_{st} \setminus \widetilde{Z}$. This is because the flow going in or out of a node $v \in W \setminus \{s',t'\}$ is either 0 or 1, for each node $v \in W \setminus \{s',t'\}$ has at most one arc going in from s' and at most one arc goint out to t'. Remark that some of these paths may correspond to non node-disjoint 3-st-paths of G, that is they are of the form (s,u,v',t) and (s,v,z',t). Let $\widetilde{\mathcal{P}}'$ be the set of these paths. Also let $\mathcal{P}'$ be the set of paths of G corresponding to those of $\widetilde{\mathcal{P}}'$ and $U \subseteq V$ the set of nodes of G which are shared by two paths of $\mathcal{P}'$. Now, add to Z the nodes of U and repeat this procedure until $|Z| \geq k$ or $U = \emptyset$. It should be noticed that when $U = \emptyset$, the arc-disjoint st-dipaths obtained by the computation of the maximum flow in $\widetilde{G}_{st} \setminus \widetilde{Z}$ correspond to node-disjoint 3-st-paths of $G \setminus Z$.

The identification of the nodes of U can be easily done by simply considering, for each node $u \in V \setminus Z$, the arcs entering and leaving nodes u and u' with flow value 1. Namely, consider a node $v \in U$. This means that after the maximum flow computation, there are two paths (s,u,v',t) and (s,v,z',t). Since the arc capacities are either 0 or 1, this means that

- the flow value on arc (s, v) is 1,
- the flow value on arc (v, v') is 0,
- the flow value on arc (v', t) is 1.

Figure 6.2 illustrates the above remark. The solid lines represent arcs having flow value 1 and dashed lines represent arcs with flow value 0. The flow value of the arcs represented by dotted lines may be 0 or 1.

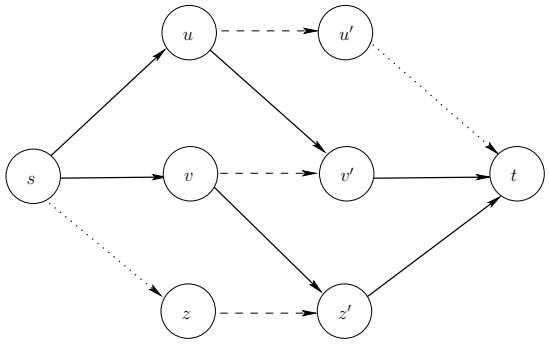

Figure 5.3: Two st-dipaths of $\widetilde{G}_{st} \setminus \widetilde{Z}$ inducing non node-disjoint 3-st-paths in $G \setminus Z$.

Thus, let $Z \subseteq V \setminus \{s, t\}$ be the node set obtained by the application of procedure BuildZ. It is not hard to see that by the construction of Z, the graph $G(\overline{x})$ contains $|Z|$ st-paths of the form (s, u, t), for all $u \in Z$. Clearly, these paths are node-disjoint. This also implies that $|Z| \leq k - 1$, for otherwise, $G(\overline{x})$ would contain at least k node-disjoint 3-st-paths, which is not possible. Now compute a maximum flow from s to t in $\widetilde{G}_{st} \setminus \widetilde{Z}$, and let f be the value of that flow. By the construction of Z, this later flow corresponds to a set of f disjoint 3-st-paths of G which are node-disjoint. Moreover, these paths are node-disjoint from those induced by Z. Thus, together with the paths induced by Z, we obtain $|Z| + f$ node-disjoint 3-st-paths in $G(\overline{x})$. As by assumption, $G(\overline{x})$ does not contain k node-disjoint 3-st-paths, we have that $|Z| + f < k$, that is $f < k - |Z|$.

Now, as f is the value of the maximum flow of $\widetilde{G}_{st} \setminus Z$, the weight of a minimum cut $\widetilde{C}$ of $\widetilde{G}_{st} \setminus Z$ is $\widetilde{y}(\widetilde{C}) = f < k - |Z|$. Finally, as shown by Bendali et al. [10], $\widetilde{C}$ corresponds to to an edge set C which is either an st-cut or a 3-st-path-cut of $G \setminus Z$, that is C corresponds to an st-node-cut or a 3-st-node-path-cut of G whose weight is $\overline{x}(C) = \widetilde{y}(\widetilde{C}) = f < k - |Z|$. Consequently, the st-node-cut or 3-st-node-cut induced by C is violated by $\overline{x}$. □

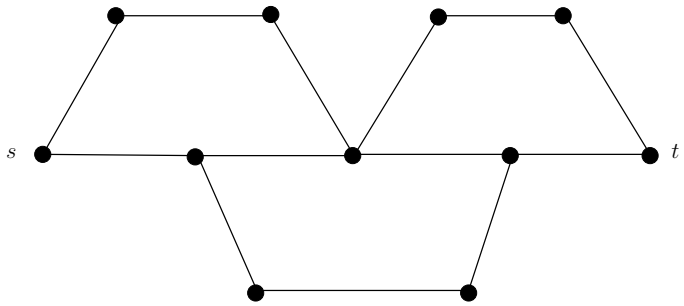

Figure 5.4: Infeasible solution of the 2NDHP with $L = 5$ and $D = \{(s, t)\}$.

From Theorem 13 the kNDHP is equivalent to

$$\min\{cx \mid x \text{ satisfies } (5.1) - (5.6) \text{ and } x \in \mathbb{Z}_+^E\}. \tag{5.7}$$

We will call inequalities (5.1)-(5.6) *basic inequalities*. Here *basic* means that they are necessary in the basic formulation of the problem. We will denote by kNDHP(G, L) the convex hull of all the integer solutions of (5.1)-(5.6), and call kNDHP(G, L) *the k-Node-Disjoint Hop-Constrained Problem polytope*.

Formulation (5.7) is no longer valid for $L \geq 5$. Consider for example the graph shown in Figure 5.4. For $k = 2$ its incidence vector satisfies inequalities (5.1)-(5.6) but the graph does not contain two node-disjoint st-paths of length at most $L = 5$. This example is borrowed from [50].

5.2 Polytope and valid inequalities

In this section, we present several classes of valid inequalities inspired from the kEDHP that have been introduced in the literature. Since any solution of the kNDHP is also solution of the kEDHP, any valid inequality for the kEDHP polytope on G is also valid for kNDHP(G, L). Also note that if $S \subseteq E$ is a solution of kNDHP in G and $Z \subset V$, such that $|Z| \leq k - 1$, then the restriction of S on $G \setminus Z$ is a solution of the $(k - |Z|)$NDHP on $G \setminus Z$ with respect to origin-destination pairs contained in $G \setminus Z$.

Lemma 9 *Let $Z \subset V$, and let $D' \subseteq D$ be a subset of origin-destination pairs in $G \setminus Z$. Suppose that $D' \neq \emptyset$. If an inequality $ax \geq \alpha(k)$ is valid for kNDHP(G, L) in G with*

respect to D then the inequality $a'x \geq \alpha(k - |Z|)$ is valid for kNDHP$(G \setminus Z, L)$, with respect to D', where a' is the restriction of a on $G \setminus Z$.

Note that in Lemma 9 we consider $\alpha(k)$ as a right hand side in the inequality $ax \geq \alpha(k)$ just to express the fact that the right hand side of a valid inequality of the kNDHP may depend of k.

5.2.1 Generalized L-st-path-cut inequalities

Dahl and Gouveia [23] introduce the so-called generalized L-*st*-path-cut inequalities for the problem of finding an L-*st*-path between two nodes s and t. They are defined as follows. Let $(s,t) \in D$ and $\pi = (V_0, ..., V_{L+r})$, $r \geq 1$, be a partition of V such that $s \in V_0$ and $t \in V_{L+r}$. Then the generalized *L-st-path-cut inequality* induced by (s,t) and π is

$$\sum_{e \in [V_i, V_j], i \neq j} \min(|i-j|-1, r)x(e) \geq r. \tag{5.8}$$

These inequalities can be easily extended to the kNDHP by replacing the right-hand-side of inequality (5.8) by $(k - |Z|)r$, with $Z \subset V$, $|Z| \leq k-1$, yielding

$$\sum_{e \in [V_i, V_j], i \neq j} \min(|i-j|-1, r)x(e) \geq (k - |Z|)r. \tag{5.9}$$

Inequality (5.9) is valid for kNDHP(G, L). A *jump* is an edge between two non-consecutive sets of π. Inequality (5.9) gives the minimum number of jumps in a partition $\pi = (V_0, ..., V_{L+r})$ needed in a solution of the problem. Inequalities of type (5.9) will also be called *generalized L-st-path-cut inequalities*.

5.2.2 Double cut inequalities

Huygens et al. [49] introduce the so-called *double cut inequalities* for the 2EDHP for $L = 3$. They are defined as follows. Consider the partition $\pi = (V_0^1, V_0^2, V_1, ..., V_4)$ of V such that $(V_0^1, V_0^2 \cup V_1, V_2, V_3, V_4)$ 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 double cut inequality can be written as

$$\begin{aligned} &x([V_0^1, V_1 \cup V_2 \cup V_3 \cup V_4]) + x([V_0^2, V_1 \cup V_3 \cup V_4]) + \\ &x([V_1, V_3 \cup V_4]) + x([V_0^2 \cup V_1 \cup V_4, V_2]) \geq \left\lceil 3 - \frac{|F|}{2} \right\rceil \end{aligned} \tag{5.10}$$

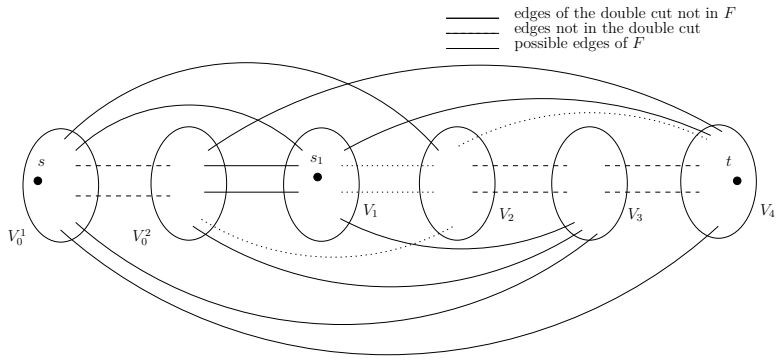

Figure 5.5: A double cut with $L = 3$ and $t_1 = t$.

We now generalize these inequalities for the kNDHP for $L \geq 2$. Let $Z \subset V \setminus \{s, t\}$, for $(s, t) \in D$, and $V_0, ..., V_{i_0-1}, V_{i_0}^1, V_{i_0}^2, V_{i_0+1}, ..., V_{L+1}$ be a family of node subsets of $V \setminus Z$ such that $\pi = (V_0, ..., V_{i_0-1}, V_{i_0}^1, V_{i_0}^2 \cup V_{i_0+1}, ..., V_{L+1})$ induces a partition of $G \setminus Z$. Suppose that

1. there exists an $(s, t) \in D$ such that $V_{i_0}^1 \cup V_{i_0}^2$ induces an st-node-cut in $G \setminus Z$ and $s \in V_{i_0}^1$ or $t \in V_{i_0}^1$,

2. there exists an $(s, t) \in D$ such that V_{i_0+1} induces an st-node-cut in $G \setminus Z$,

3. there exists an $(s, t) \in D$ such that π induces an L-st-node-path-cut in $G \setminus Z$ with $s \in V_0$ (resp. $t \in V_0$) and $t \in V_{L+1}$ (resp. $s \in V_{L+1}$).

Let $\overline{E} = [V_{i_0-1}, V_{i_0}^1] \cup [V_{i_0+2}, V_{i_0}^2 \cup V_{i_0+1}] \cup \left(\bigcup_{k,l \notin \{i_0, i_0+1\}, |k-l|>1} [V_k, V_l] \right)$ and $F \subseteq \overline{E}$ such that $|F|$ and $k - |Z|$ have different parities.

Let also $\hat{E} = (\bigcup_{i=0}^{i_0-2} [V_i, V_{i+1}]) \cup (\bigcup_{i=i_0+2}^{L} [V_i, V_{i+1}]) \cup F$. Then we have the following inequality.

$$x(\delta(\pi) \setminus \hat{E}) \geq \left\lceil \frac{3(k - |Z|) - |F|}{2} \right\rceil \tag{5.11}$$

Theorem 14 *Inequalities* (5.11) *are valid for kNDHP(G, L).*

Proof. Let $T_{G\setminus Z}$ be the L-st-node-path-cut of $G \setminus Z$ induced by the partition π and Z. Thus, the following inequalities are valid for kNDHP(G, L),

$$\begin{aligned} x_{G\setminus Z}(T) &\geq k - |Z|, \\ x(\delta_{G\setminus Z}(V_{i_0}^1 \cup V_{i_0}^2)) &\geq k - |Z|, \\ x(\delta_{G\setminus Z}(V_{i_0+1})) &\geq k - |Z|, \\ -x(e) &\geq -1 \text{ for all } e \in F, \\ x(e) &\geq 0 \quad \text{for all } e \in \overline{E} \setminus F. \end{aligned} \tag{5.12}$$

By summing these inequalities, divinding by 2 and rouding up the right hand side, we obtain inequality (5.11). □

These inequalities will also be called *double-cut inequalities*.

If $L = 3$ and $i_0 = 0$, inequality (5.11) can be written as

$$\begin{aligned} x([V_0^1, V_1 \cup V_2 \cup V_3 \cup V_4]) + x([V_0^2, V_1 \cup V_3 \cup V_4]) + x([V_1, V_3 \cup V_4]) + \\ x([V_0^2 \cup V_1 \cup V_4, V_2] \setminus F) \geq \left\lceil \frac{3(k - |Z|) - |F|}{2} \right\rceil. \end{aligned} \tag{5.13}$$

Here $\pi = (V_0^1, V_0^2 \cup V_1, V_2, V_3, V_4)$ and $F \subseteq [V_0^2 \cup V_1 \cup V_4, V_2]$ such that $|F|$ and $k - |Z|$ have different parities.

5.2.3 Triple path-cut inequalities

Huygens et al. [49] introduce the so-called *triple path-cut inequalities* for the 2EDHP for $L = 3$. They are defined for a partition $(V_0, V_1, ..., V_5)$ of V with $s_1, s_2 \in V_0$, $t_1 \in V_4$ and $t_2 \in V_5$. Then the triple path-cut inequality

$$\begin{aligned} 2x([V_0, V_2]) + 2x([V_0, V_3]) + 2x([V_1, V_3]) + \\ x([V_0 \cup V_1 \cup V_2 \cup V_3, V_4 \cup V_5] \setminus \{e\}) + x([V_4, V_5]) \geq 3 \end{aligned} \tag{5.14}$$

where $e \in [V_2 \cup V_3, V_4] \cup [V_3, V_5]$, is valid for 2EDHP$(G, 3)$.

We now generalize these inequalities for the kNDHP for $L = 3$.

Theorem 15 *Let $Z \subset V \setminus R_D$, where R_D is the set of terminal nodes of G. Let $(V_0, ..., V_3, V_4^1, V_4^2, V_5^1, V_5^2)$ be a family of node sets of $V \setminus Z$ such that $(V_0, ..., V_3, V_4^1 \cup V_4^2, V_5^1 \cup V_5^2)$ induces a partition of $V \setminus Z$ and there exist two demands $\{s_1, t_1\}$ and $\{s_2, t_2\}$ with $s_1, s_2 \in V_0$, $t_1 \in V_4^2$ and $t_2 \in V_5^2$. The sets V_4^1 and V_5^1 may be empty and s_1 and s_2 may be the same. Let also $V_4 = V_4^1 \cup V_4^2$, $V_5 = V_5^1 \cup V_5^2$ and $F \subseteq [V_2, V_4^2] \cup [V_3, V_4 \cup V_5]$ such that $|F|$ and $k - |Z|$ have different parities. Then, the inequality*

$$2x([V_0, V_2]) + 2x([V_0, V_3]) + 2x([V_1, V_3]) + x([V_0 \cup V_1, V_4 \cup V_5]) + x([V_4, V_5]) + x([V_2, V_5]) + x(([V_2, V_4] \cup [V_3, V_4 \cup V_5]) \setminus F) \geq \left\lceil \frac{3(k - |Z|) - |F|}{2} \right\rceil \quad (5.15)$$

is valid for kNDHP(G, 3).

Proof. Let T_1 be the 3-s_1t_1-node-path-cut induced by the partition $(V_0, V_1 \cup V_5, V_2, V_3 \cup V_4^1, V_4^2)$ and Z, and T_2 and T_3 be the 3-s_2t_2-node-path-cuts induced by the partitions $(V_0, V_1 \cup V_4, V_2, V_3 \cup V_5^1, V_5^2)$ and $(V_0, V_1, V_2, V_3 \cup V_4 \cup V_5^1, V_5^2)$, respectively, and Z. The following inequalities are valid for kNDHP$(G, 3)$.

$$\begin{aligned} x_{G\setminus Z}(T_1) &\geq k - |Z|, \\ x_{G\setminus Z}(T_2) &\geq k - |Z|, \\ x_{G\setminus Z}(T_3) &\geq k - |Z|, \\ -x(e) &\geq -1 \qquad \text{for all } e \in F, \\ x(e) &\geq 0 \qquad \text{for all } e \in ([V_2, V_4^2] \cup [V_3, V_4 \cup V_5]) \setminus F. \end{aligned} \quad (5.16)$$

By summing these inequalities, divinding by 2 and rouding up the right hand side, we obtain inequality (5.15). □

Inequalities (5.15) will also be called *triple-path-cut inequalities*. Figure 5.6 gives an illustration.

5.2.4 Steiner-partition inequalities

Let R_D be the set of terminal nodes of G. Let $Z \subset V \setminus R_D$, and $(V_0, V_1, ..., V_p)$, $p \geq 2$, be a partition of $V \setminus Z$ such that $V_0 \subseteq V \setminus R_D$, and for all $i \in \{1, ..., p\}$ there is a demand

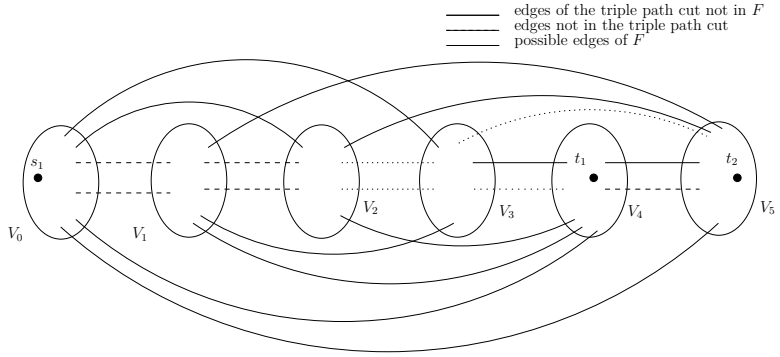

Figure 5.6: A triple-path cut with $k = 2$, $L = 3$ and $s_1 = s_2$.

$\{s, t\} \in D$ such that V_i induces an st-cut of G. We can see that V_0 may be empty. The partition $(V_0, V_1, ..., V_p)$ is called a *Steiner-partition*. And we have the following inequality

$$x(\delta_{G\setminus Z}(V_0, ..., V_p)) \geq \left\lceil \frac{(k - |Z|)p}{2} \right\rceil. \tag{5.17}$$

Inequalities of type (5.17) will be called *Steiner-partition inequalities*.

Theorem 16 *Inequalities* (5.17) *are valid for kNDHP(G, L).*

Proof. The following inequalities are valid for kNDHP(G, L).

$$\begin{aligned} x_{G\setminus Z}(V_i) &\geq k - |Z|, \text{ for } i = 1, ..., p, \\ x(e) &\geq 0, \qquad \text{for all } e \in \delta(V_0). \end{aligned} \tag{5.18}$$

By adding them we obtain

$$2x(\delta_{G\setminus Z}(V_0, ..., V_p)) \geq (k - |Z|)p.$$

By dividing by 2 and rounding up the right hand side, we get inequality (5.17). □

5.2.5 Steiner SP-partition inequalities

Diarrassouba et al. [27] introduced the so-called Steiner SP-partition inequalities for the kEDHP. In what follows we extend these inequalities to the kNDHP. They are defined as follows. Let $Z \subset V \backslash R_D$, where R_D is the set of terminal nodes of G. Consider a partition $\pi = (V_1, ..., V_p)$, $p \geq 3$, of $V \setminus Z$, such that the graph $G_\pi = (V_\pi, E_\pi)$ is series-parallel (G_π is the graph obtained by contracting the sets V_i, $i = 1, ...p$). Suppose that $V_\pi = \{v_1, ..., v_p\}$ where v_i is the node of G_π obtained after the contraction of the set V_i, $i = 1, ..., p$. The partition π is called a *Steiner-SP-partition* if and only if π is a Steiner-partition and either

1. $p = 3$ or

2. $p \geq 4$ and there exists a node $v_{i_0} \in V_\pi$ incident to exactly two nodes v_{i_0-1} and v_{i_0+1} such that after the contraction of the sets V_{i_0}, V_{i_0-1} and V_{i_0}, V_{i_0+1}, the partitions π_1 and π_2 obtained from π are also Steiner-SP-partitions.

Theorem 17 *[27] Let $\pi = (V_1, ..., V_p)$, $p \geq 3$, be a partition of V such that G_π is series-parallel. The partition π is a Steiner-SP-partition of G if and only if the subgraph of G_D induced by π is connected.*

From Theorem 17, note that if the demand graph is connected, then every Steiner-partition of $V \setminus Z$ inducing a series-parallel subgraph of $G \setminus Z$ is a Steiner-SP -partition of $V \setminus Z$. With a Steiner-SP-partition $(V_1, ..., V_p)$, $p \geq 3$, we associate the following inequality

$$x(\delta_{G\setminus Z}(V_1, ..., V_p)) \geq \left\lceil \frac{k - |Z|}{2} \right\rceil p - 1. \tag{5.19}$$

Inequalities of type (5.19) are called *Steiner-SP-partition inequalities.*

Theorem 18 *Inequalities* (5.19) *are valid for kNDHP(G, L).*

Proof. Let $\pi = (V_1, ..., V_p)$ be a Steiner-SP-partition. The proof is by induction on p. If $p = 3$, as π is a Steiner-partition then we associate with π the inequality

$$x(\delta_{G\setminus Z}(V_1, V_2, V_3)) \geq \left\lceil \frac{3(k-|Z|)}{2} \right\rceil = 3 \left\lceil \frac{(k-|Z|)}{2} \right\rceil - 1. \tag{5.20}$$

Now suppose that every inequality (5.19) induced by a Steiner-SP-partition of p elements, $p \geq 3$, is valid for kNDHP$(G, 3)$ and let $\pi = (V_1, ..., V_p,$
$V_{p+1})$ be a Steiner-SP-partition. Since G_π is series-parallel, then there exists a node set V_{i_0} of π such that it is incident to exactly two elements of π, V_{i_0-1} and V_{i_0+1}. Let $T_1 = [V_{i_0}, V_{i_0+1}]$ and $T_2 = [V_{i_0}, V_{i_0-1}]$. As π is a Steiner-SP-partition, it is also a Steiner-partition. As V_{i_0} and $Z \subset V \setminus \{s, t\}$ induce a valid st-node-cut inequality, for some $\{s, t\} \in D$. Thus $x(T_1) + x(T_2) \geq k - |Z|$. W.l.o.g., we suppose that

$$x(T_1) \geq \left\lceil \frac{(k-|Z|)}{2} \right\rceil. \tag{5.21}$$

Let $\pi' = (V_1, ..., V_{i_0-2}, V_{i_0-1} \cup V_{i_0}, V_{i_0+1}, ..., V_{p+1})$ be a partition. As π is a Steiner-SP-partition which contains more than three elements, π' is also a Steiner-SP-partition with p elements. Then, by the induction hypothesis, we have the following valid Steiner-SP-partition inequality induced by π'.

$$x(\delta_{G\setminus Z}(V_1, ..., V_{i_0-2}, V_{i_0-1} \cup V_{i_0}, V_{i_0+1}, ..., V_{p+1})) \geq \left\lceil \frac{k-|Z|}{2} \right\rceil p - 1. \tag{5.22}$$

By summing the inequalities (5.21) and (5.22), we get

$$x(\delta_{G\setminus Z}(V_1, ..., V_p, V_{p+1})) \geq \left\lceil \frac{k-|Z|}{2} \right\rceil (p+1) - 1. \tag{5.23}$$

Hence, we have the result. □

5.2.6 The rooted partition inequalities

A further class of valid inequalities is the rooted partition inequalities. We consider p demands, $|D| \geq p \geq 2$, of the form (s, t_i), $i = 1, ...p$, for $s \in V$ and $t_i \in V \setminus \{s\}$. Let $(V_0, V_1, ..., V_p)$ be a partition of V such that $s \in V_0$ and $t_i \in V_i$, for all $i \in \{1, ..., p\}$. This partition is called a *rooted partition*. Huygens et al. [49] showed that, for any $L \geq 2$, the following inequality is valid for the 2EDHP polytope.

$$x(\delta(V_0, V_1, ..., V_p)) \geq \left\lceil \frac{(L+1)p}{L} \right\rceil. \tag{5.24}$$

For a subset $Z \subset V$ with $|Z| = k-2$, the following inequality is valid for kNDHP(G, L).

$$x(\delta_{G\setminus Z}(V_0, V_1, ..., V_p)) \geq \left\lceil \frac{(L+1)p}{L} \right\rceil. \tag{5.25}$$

5.2.7 *st*-jump inequalities

Theorem 19 *Suppose that $|V| \geq 5$ and $L = 3$. Let $(s,t) \in D$, $Z \subset V$, and consider the partition $\pi = (V_0, V_1, ..., V_4)$ of $V \setminus Z$ such that $s \in V_0$ and $t \in V_4$. Let U_i be a set of nodes of V_i, $i = 1, 2, 3$, such that $|U_i| = k-1$. Then the st-jump inequality*

$$\sum_{i=0}^{2} x([V_i, V_{i+2}]) + \sum_{i=0}^{1} \sum_{j\geq i+3}^{4} 2x([V_i, V_j]) + \sum_{i=0}^{1} x([V_i, V_{i+1} \setminus U_{i+1}]) + \sum_{i=2}^{3} x([V_i \setminus U_i, V_{i+1}]) \geq \left\lceil \frac{4k+3}{5} \right\rceil \tag{5.26}$$

is valid for the kNDHP$(G, 3)$.

Proof. Let $U_1 \subset V_1$, $U_2 \subset V_2$ and $U_3 \subset V_3$ and let T_1, T_2, T_3 and T_4, be the *L-st-path-cuts* induced by $(V_0, U_1, V_2 \cup V_1 \setminus U_1, V_3, V_4)$, $(V_0, V_1, U_2, V_3 \cup V_2 \setminus U_2, V_4)$, $(V_0, V_1 \cup V_2 \setminus U_2, U_2, V_3, V_4)$ and $(V_0, V_1, V_2 \cup V_3 \setminus U_3, U_3, V_4)$, respectively. Then by summing the *L-st*-path-cut inequalities induced by T_i, $i = 1, ..., 4$, and the following *st*-node-cut inequalities induced by V_0 and U_1, $V_0 \cup V_1$ and U_2, and V_4 and U_3,

$$\begin{aligned} x(\delta_{G\setminus\{U_1\}}(V_0)) &\geq k-1, \\ x(\delta_{G\setminus\{U_2\}}(V_0 \cup V_1)) &\geq k-1, \\ x(\delta_{G\setminus\{U_3\}}(V_4)) &\geq k-1, \end{aligned}$$

we obtain the inequality

$$\sum_{i=1}^{4} x(T_i) + x(\delta_{G\setminus\{U_1\}}(V_0)) + x(\delta_{G\setminus\{U_2\}}(V_0 \cup V_1)) + x(\delta_{G\setminus\{U_3\}}(V_4)) \geq 4k+3.$$

This together with

$$\begin{aligned} x(e) &\geq 0, \text{ for all } e \in \delta(V_1 \setminus U_1, V_3) \cup \delta(V_2 \setminus U_2, V_4) \cup \delta(V_1, V_3 \setminus U_3), \\ 3x(e) &\geq 0, \text{ for all } e \in \delta(V_0, V_1 \setminus U_1 \cup V_4) \cup \delta(V_1, V_2 \setminus U_2) \cup \delta(V_3 \setminus U_3, V_4), \\ 4x(e) &\geq 0, \text{ for all } e \in \delta(V_0 \cup V_2 \setminus U_2 \cup V_3) \cup \delta(V_1, V_4), \end{aligned}$$

gives the inequality

$$\sum_{i=0}^{2} 5x([V_i, V_{i+2}]) + \sum_{i=0}^{1} \sum_{j \geq i+3}^{4} 10x([V_i, V_j]) + \sum_{i=0}^{1} 5x([V_i, V_{i+1} \setminus U_{i+1}]) + \sum_{i=2}^{3} 5x([V_i \setminus \{u_i\}, V_{i+1}]) \geq 4k + 3.$$

Dividing the resulting inequality by 5, and rounding up the right-hand side, we obtain inequality (5.26).

□

5.3 Facets of the kNDHP polytope

In this section, we investigate the conditions under which the inequalities presented in the previous section define facets of kNDHP(G, L). First, we discuss the dimension of kNDHP(G, L).

An edge $e \in E$ is said to be *essential* if there is no solution of the kNDHP on the graph obtained by deleting the edge e from G. Therefore e is essential if and only if it belongs to either an st-cut or an L-st-path-cut of cardinality k, or, to an st-node-cut or an L-st-path-node-cut of cardinality $k - |Z|$. Then we have the following theorem.

Theorem 20 *dim(kNDHP(G, L)) $= |E| - |E^*|$, where $|E^*|$ is the set of essential edges.*

Proof. We have that the edges of E^* belong to every solution of the problem, meaning that, $x^F(e) = 1$, for all $e \in E^*$, and every solution $F \subseteq E$ of the problem. Then we have $\dim(k\text{NDHP}(G, L)) \leq |E| - |E^*|$. By considering the edge sets E and $E_f = E \setminus \{f\}$, for every $f \in E \setminus E^*$, we can clearly see that they form $|E| - |E^*| + 1$ solutions, and their incidence vectors are affinely independent. Therefore $\dim(k\text{NDHP}(G, L)) \geq |E| - |E^*|$, and the result follows.

□

Corollary 7 *kNDHP(G, L) is full dimensional if $G = (V, E)$ is complete and $|V| \geq k + 2$.*

In the following we assume that G is complete and has at least $k+2$ nodes. By Corollary 7, kNDHP(G, L) is then full dimensional.

Now we investigate the conditions under which the trivial and basic inequalities define facets.

Theorem 21 *Inequality $x(e) \leq 1$ defines a facet of kNDHP(G, L) for every $e \in E$.*

Proof. For all $f \in E \setminus \{e\}$, consider the edge sets $E_f = E \setminus \{f\}$. Hence, E and the edge sets E_f constitute a set of $|E|$ solutions of the kNDHP. Furthermore, their incidence vectors satisfy $x(e) = 1$ and are affinely independent.

□

Theorem 22 *Inequality $x(e) \geq 0$, with $e = uv \in E$, defines a facet of kNDHP(G, L) if one of the following conditions hold.*
1) $|V| \geq k+3$,
2) $|V| = k+2$, $|D| \leq k-1$ *and* $(u, v) \notin D$.

Proof. Suppose that $|V| = k+2$, $|D| \leq k-1$, and $(u, v) \notin D$. Then, the edge sets $E \setminus \{e\}$ and $E_f = E \setminus \{e, f\}$, for all $f \in E \setminus \{e\}$, are solutions of kNDHP whose incidence vectors satisfy $x(e) = 0$ and are affinely independent.

Now, suppose that $|V| \geq k+3$. Then for all the demands $(s, t) \in D$, the graph G contains $k+2$ node-disjoint st-paths (edge st and the $k+1$ paths of the form $(s, u, t), u \in V \setminus \{s, t\}$). Thus the sets $E \setminus \{e\}$ and $E_f = E \setminus \{e, f\}$, for all $f \in E \setminus \{e\}$, form a set of $|E|$ solutions of the kNDHP. Moreover, their incidence vectors satisfy $x(e) = 0$ and are affinely independent. Hence $x(e) \geq 0$ defines a facet of kNDHP(G, L).

□

In what follows, we investigate the conditions under which the st-cut and the st-node-cut inequalities define facets of kNDHP(G, L).

Theorem 23 *The st-cut inequalities $x(\delta(W)) \geq k$ define facets of kNDHP(G, L) when $|D| = 1$.*

Proof. We denote by $ax \geq \alpha$ the st-cut inequality induced by W, and let $\mathscr{F} = \{x \in k\text{NDHP}(G, L) \mid ax = \alpha\}$. Suppose there exists a defining facet inequality $bx \geq \beta$ such that $\mathscr{F} \subseteq \mathscr{F}' = \{x \in k\text{NDHP}(G, L) \mid bx = \beta\}$. We will prove that there is a scalar ρ such that $b = \rho a$. As $|V| \geq k + 2$, there exists $W_1 \subseteq W \setminus \{s\}$ and $W_2 \subseteq \overline{W} \setminus \{t\}$ such that $|W_1| + |W_2| = k$. Let $E_1 = \{sv, v \in W_2\} \cup \{ut, u \in W_1\}$ and $T_1 = E_1 \cup E_0$ where $E_0 = E(W) \cup E(\overline{W})$. Clearly, T_1 is a solution of the kNDHP, and its incidence vector satisfies $ax \geq \alpha$ with equality. Consider an edge $e \in E_1$. It is not hard to see that $T_2 = (T_1 \setminus \{e\}) \cup \{st\}$ is a solution of the kNDHP and its incidence vector also satisfies $ax \geq \alpha$ with equality. Thus $bx^{T_1} = bx^{T_2}$. Since $bx^{T_2} = bx^{T_1} - b(e) + b(st)$, we obtain that $b(e) = b(st)$. As e is an arbitrary edge in E_1, this implies that

$$b(e) = b(st) = \rho \text{ for some } \rho \in \mathbb{R} \quad \text{for all } e \in E_1. \tag{5.27}$$

Now consider an edge $f = uv \in \delta(W) \setminus E_1$, with $u \in W \setminus \{s\}$ and $v \in \overline{W} \setminus \{t\}$. We distinguish two cases.

Case 1. $u \in W_1$, $v \in W_2$.

Consider $T_3 = (T_1 \setminus \{sv, ut\}) \cup \{f, st\}$. Clearly, T_3 is a solution of the kNDHP and its incidence vector satisfies $ax = \alpha$. Hence, we have that $bx^{T_3} = bx^{T_1}$. This implies that $b(sv) + b(ut) = b(f) + b(st)$. From (5.27), it follows that $b(f) = \rho$.

Case 2.$u \in W_1$ (resp. $u \in W \setminus (W_1 \cup \{s\})$), $v \in \overline{W} \setminus (W_2 \cup \{t\})$ (resp. $v \in W_2$).

Consider the edge set $T_4 = (T_1 \setminus \{tu\}) \cup \{f\}$. It is easy to see that T_4 is a solution of kNDHP such that $ax^{T_4} = \alpha$. Hence $bx^{T_4} = \beta$. As $bx^{T_1} = \beta$, it follows that $b(f) = b(tu) = \rho$.

If $u \in W \setminus (W_1 \cup \{s\})$ and $v \in W_2$ we also obtain by symmetry that $b(f) = \rho$.

Thus, together with (5.27) we obtain that $b(e) = \rho$ for all $e \in \delta(W)$.

Now consider an edge $e \in E_0$, and suppose, w.l.o.g., that $e \in E(W)$. If e does not belong to an L-st-path of T_1, then the edge set $T_5 = T_1 \setminus \{e\}$ also induces a solution of the kNDHP and satisfies $ax^{T_5} = \alpha$. Hence we have that $bx^{T_5} = bx^{T_1}$ implying $b(e) = 0$. If e belongs to an L-st-path of T_1, say (su, ut) where $e = su$, then the edge set $T_6 = (T_1 \setminus \{su, ut\}) \cup \{st\}$ induces a solution of the kNDHP, and its incidence vector satisfies $ax^{T_6} = \alpha$. Consequently $bx^{T_7} = bx^{T_1}$ and therefore, $b(st) = b(su) + b(ut)$. As (5.27), $b(ut) = b(st)$, it follows that $b(su) = 0$.

Hence $b(e) = 0$ for all $e \in E_0$.

Finally, we have that

$$b(e) = \begin{cases} \rho & \text{for all } e \in \delta(W), \\ 0 & \text{if not.} \end{cases}$$

Consequently, $b = \rho a$ with $\rho \in \mathbb{R}$, which finishes the proof. □

Theorem 24 *If $|V| \geq 2k+1$ and $|D| = 1$ with $D = \{(s,t)\}$, then every st-node-cut inequality $x(\delta_{G\setminus Z}(W)) \geq k - |Z|$ where $Z \subset V \setminus \{s,t\}$, and such that $s \in W$, $t \notin W$ and $W \setminus \{s\} \neq \emptyset \neq V \setminus ((W \cup Z) \setminus \{t\})$, defines a facet of kNDHP(G,L).*

Proof. Let us denote by $ax \geq \alpha$ the inequality (5.3) induced by W and Z, and let $bx \geq \beta$ be a facet defining inequality of kNDHP(G,L) such that $\{x \in k\text{NDHP}(G,L) : ax = \alpha\} \subseteq \{x \in k\text{NDHP}(G,L) : bx = \beta\}$. As before we will show that there exists a scalar $\rho \in \mathbb{R}$ such that $b = \rho a$.

The idea of the proof is to use the fact that $x(\delta_{G\setminus Z}(W)) \geq k - |Z|$ is a valid cut inequality for $(k-|Z|)$NDHP$(G \setminus Z, L)$, and hence, by Theorem 23, defines a facet of $(k-|Z|)$NDHP$(G \setminus Z, L)$. Thus there exist $\dim((k-|Z|)\text{NDHP}(G \setminus Z, L))$ solutions of the $(k-|Z|)$NDHP on $G \setminus Z$ whose incidence vectors satisfy $x(\delta_{G\setminus Z}(W)) \geq k - |Z|$ with equality and are affinely independent. In what follows, we will use these solutions to build $|E|$ solutions of the kNDHP on G satisfying $x(\delta_{G\setminus Z}(W)) \geq k - |Z|$ with equality and which are affinely independent. Notice that as G is complete, $|Z| \leq k-1$ and $|V| \geq 2k+1$, $G \setminus Z$ is also complete with $|V \setminus Z| \geq k+2$. Thus, by Corollary 7, the polytope $(k-|Z|)$NDHP$(G \setminus Z, L)$ is full dimensional, and hence $\dim((k-|Z|)\text{NDHP}(G \setminus Z, L)) = |E| - |\delta(Z)| - |E(Z)|$.

As $x(\delta_{G\setminus Z}(W)) \geq k-|Z|$ defines a facet of $(k-|Z|)$NDHP$(G\setminus Z, L)$, there must exist $m' = |E| - |\delta(Z)| - |E(Z)|$ solutions of the $(k-|Z|)$NDHP on $G \setminus Z$, denoted by T'_i, $i = 1, ..., m'$, whose incidence vectors are affinely independent and satisfy $x(\delta_{G\setminus Z}(W)) = k - |Z|$.

The edge sets $T_i = T'_i \cup \delta(Z) \cup E(Z)$, for all $i \in \{1, ..., m'\}$, induce solutions of the kNDHP. Indeed, since G is complete, the paths (s,z,t), $z \in Z$, form a set of $|Z|$ st-paths of length 2 in G. As these st-paths are node-disjoint and do not intersect $V \setminus (Z \cup \{s,t\})$, they form with the s-paths of T'_i a set at least k node-disjoint st-paths in G, for $i = 1, ..., m'$. Which implies that T_i, $i = 1, ..., m'$ are solutions of kNDHP.

Furthermore, their incidence vectors satisfy $x(\delta_{G\setminus Z}(W)) = k - |Z|$ and are affinely independant.

Let a' and b' be the restriction on $E \setminus (\delta(Z) \cup E(Z))$ of a and b, respectively. Thus, we have $a'x^{T_i} = \alpha$, for $i = 1, ..., m'$. Therefore, $b'x^{T_i} = \beta$, for $i = 1, ..., m'$. As x^{T_i}, $i = 1, ..., m'$, are affinely independent and $\alpha \neq 0$, it follows that x^{T_i}, $i = 1, ..., m'$, are linearly independent. Consequently, a is the unique solution of the system $a'x^{T_i} = \alpha$, for $i = 1, ..., m'$. Let ρ be such that $\beta = \rho\alpha$. It then follows that $b' = \rho a'$. This implies that $b(e) = 0$ for all $e \in E(W) \cup E(\overline{W})$.

Now we will show that $b(e) = 0$ for all $e \in \delta(Z) \cup E(Z)$. Let us denote the edges of $E(Z) \cup \delta(Z) \setminus \bigcup_{z \in Z} \{sz, zt\}$ by e_j, $j = 1, ..., |\delta(Z)| + |E(Z)| - 2|Z|$. Consider the edge sets $\Gamma_{m'+j} = T_{m'} \setminus \{e_j\}$, for $j = 1, ..., |\delta(Z)| + |E(Z)| - 2|Z|$. We can see that these sets induce solutions of the kNDHP, and their incidence vectors satsify $x(\delta_{G\setminus Z}(W)) = k - |Z|$. As $ax^{T_{m'}} = ax^{\Gamma_{m'+j}} = \alpha$, it follows that $bx^{T_{m'}} = bx^{\Gamma_{m'+j}} = \beta$. Hence, $b(e_j) = 0$ for $j = 1, ..., |\delta(Z)| + |E(Z)| - 2|Z|$.

Let T_1 be the set among $T_1, ..., T_{m'}$ containing the edge st. Such a set exists since the inequality defines a facet of kNDHP$(G\setminus, L)$ on $G \setminus Z$ different from a trivial inequality. As $W\setminus\{s\} \neq \emptyset \neq V\setminus((W\cup Z)\setminus\{t\})$. Let $u_1 \in W\setminus\{s\}$, $u_2 \in (V\setminus(W\cup Z))\setminus\{t\}$ and $z \in Z$. Consider the edge sets $T_0 = (T_1 \setminus \{sz\}) \cup \{su_1, u_1z\}$ and $T_0' = (T_1 \setminus \{zt\}) \cup \{sz, zu_2\}$. T_0 and T_1 are solutions of the kNDHP (Recall that the path (sz, zt) belongs to T_i). Moreover we have $ax^{T_0} = ax^{T_1} = \alpha$. Thus $bx^{T_1} = bx^{T_0} = bx^{T_0'} = \beta$. As $b(su_1) = b(u_1z) = b(zu_2) = b(u_2t) = 0$, it follows that $b(sz) = b(zt) = 0$.

Therefore $b = \rho a$, which ends the proof of the theorem. □

In what follows we describe conditions under which the L-st-path and L-node-st-path cut inequalities define facets when $L = 3$.

Theorem 25 *If $|D| = 1$, a 3-st-path inequality* (5.2) *induced by a partition $\pi = (V_0, ...V_4)$ with $s \in V_0$ and $t \in V_4$, defines a facet of kNDHP$(G, 3)$ if and only if*

(1) $|V_0| = |V_4| = 1$,

(2) $|[s, V_1]| + |[V_3, t]| \geq k$.

Proof. Let T be the 3-path-cut induced by $\pi = (V_0, ..., V_4)$ such that $s \in V_0$ and $t \in V_4$. Let us denote by $ax \geq \alpha$ the L-st-path inequality induced by T, and let $\mathscr{F} = \{x \in k\text{NDHP}(G, 3) \mid ax = \alpha\}$.

Necessity. (1) We will show that if $|V_0| \geq 2$, inequality $x(T) \geq k$ does not define a facet. The case where $|V_4| \geq 2$ follows by symmetry. Suppose that $|V_0| \geq 2$ and consider the partition $\pi' = (V'_0, ..., V'_4)$ given by

$$\begin{aligned} V'_0 &= \{s\}, \\ V'_1 &= V_1 \cup (V_0 \setminus \{s\}), \\ V'_i &= V_i, \ i = 2, 3, 4. \end{aligned}$$

The partition π' produces a 3-path-cut inequality $x(T') \geq k$, where $T' = T \setminus [V_0 \setminus \{s\}, V_2]$. Since G is complete, $[V_0 \setminus \{s\}, V_2] \neq \emptyset$ and T' is strictly contained in T. Thus, $x(T) \geq k$ is redundant with respect to the inequalities

$$\begin{aligned} x(T') &\geq k, \\ x(e) &\geq 0 \quad \text{for all} \ \ e \in [V_0 \setminus \{s\}, V_2], \end{aligned}$$

and cannot define a facet.

(2) Suppose that Condition 1) holds, and that $\mathscr{F}$ is a facet of kNDHP$(G, 3)$ different from a trivial inequality. Thus there exists a solution F of the kNDHP such that $x^F \in \mathscr{F}$ and $F \cap [V_1, V_3] \neq \emptyset$. If this is not the case, then $\mathscr{F}$ would be equivalent to a facet defined by any of the inequalities $x(e) \geq 0$, $e \in [V_1, V_3]$. Note that, since each 3-st-path of F intersects T at least once and $|F \cap T| = k$, F necessarily contains exactly k node-disjoint 3-st-paths. Moreover, each of these paths intersects T only once. If u_i is a node of V_i, $i = 1, ..., 3$, this implies that every 3-st-path of F is of the form

(i) $(su_1, u_1u_2, u_2t), (su_2, u_2u_3, u_3t)$, (su_1, u_1t), (su_3, u_3t), (st) or

(ii) (su_1, u_1u_3, u_3t).

If P is one of these st-paths, then $|P \cap A| = 1$ (resp. $|P \cap A| = 2$) if P is of type (i) (resp. (ii)), where $A = [s, V_1] \cup [V_3, t] \cup \{st\}$. As $F \cap [V_1, V_3] \neq \emptyset$, it follows that F contains at least one path of type (ii) and therefore $|F \cap A| \geq k + 1$. Hence $|[s, V_1]| + |[V_3, t]| \geq k$.

Sufficiency. Suppose that Conditions (1) and (2) hold. Now suppose that there exist a facet defining inequality $bx \geq \beta$ such that $\mathscr{F} \subseteq \{x \in k\text{NDHP}(G, 3) \mid bx = \beta\}$. As before, we will show that there exists a scalar $\rho \neq 0$ such that $b = \rho a$. As $|[s, V_1]| + |[V_3, t]| \geq k$, there exist two node sets $U_1 \subseteq V_1$ and $U_3 \subseteq V_3$ such that $|U_1| + |U_3| = k$. Consider the edge subset S_1 formed by the st-paths (su, ut), $u \in U_1 \cup U_3$.

Clearly these st-paths form a set of k node-disjoint 3-st-paths. Moreover, each of these paths intersects T only once. Thus S_1 induces a solution of kNDHP and its incidence vector belongs to $\mathscr{F}$.

Let $e \in S_1 \cap T$. Let $S_2 = (S_1 \setminus \{e\}) \cup \{st\}$. Since S_2 is a solution of the kNDHP whose incidence vector belongs to $\mathscr{F}$, we have $bs^{S_2} = bx^{S_1} = \beta$, implying that $b(e) = b(st)$. As e is an arbitrary edge, we obtain that

$$b(e) = \rho \text{ for all } e \in (S_1 \cap T) \cup \{st\}, \text{ for some } \rho \in \mathbb{R}. \tag{5.28}$$

Consider now $e \in E \setminus T$. If $e \notin S_1$, clearly $S_3 = S_1 \cup \{e\}$ is a solution of kNDHP. Moreover, its incidence vector belongs to $\mathscr{F}$. Hence, $b(e) = bx^{S_3} - bx^{S_1} = 0$. If $e \in S_1 \setminus T$, then e is either of the form su, $u \in U_1$, or vt, $v \in U_3$. Suppose, w.l.o.g., that $e = su$, the case where $e = vt$ is similar. Note that, by the definition of S_1, ut also belongs to S_1. Let $S_4 = (S_1 \setminus \{su, ut\}) \cup \{st\}$. We have that S_4 induces a solution of the kNDHP and $x^{S_4} \in \mathscr{F}$. Hence, $bx^{S_4} = bx^{S_1} = \beta$ and, in consequence, $b(su) + b(ut) = b(st)$. As, by (5.28), $b(ut) = b(st)$, we have that $b(su) = 0$. Thus, we obtain that

$$b(e) = 0 \text{ for all } e \in E \setminus T. \tag{5.29}$$

Now let $e \in T \setminus S_1$. Suppose that $e = sv$ with $v \in V_2$. The case where $e \in [V_2, t]$ is similar. By construction S_1 contains an st-path of the form (su_3, u_3t) where u_3 is a node of V_3. Then the edge set $S_5 = (S_1 \setminus \{su_3\}) \cup \{e, vu_3\}$ is a solution of the kNDHP whose incidence vector belongs to $\mathscr{F}$. Thus, $b^{S_5} - b^{S_1} = b(e) + b(vu_3) - b(su_3) = 0$. From (5.28) and (5.29), we then get $b(e) = \rho$.

Let $e = sv$ with $v \in V_3$. The case where $e \in [V_1, t]$ is similar. Consider the edge set $S_6 = (S_1 \setminus \{su_3\}) \cup \{e, vt\}$, where u_3 is a node of U_3, which induces a solution of the kNDHP. Moreover, its incidence vector belongs to $\mathscr{F}$. Hence $bx^{S_6} - bx^{S_1} + b(vt) = b(e) - b(su_3) + b(vt) = 0$. By (5.28) and (5.29), we get $b(e) = \rho$.

Now suppose that $e = uv \in [V_1, V_3]$. If $u \in U_1$ and $v \in U_3$, then by considering the edge set $S_8 = (S_1 \setminus \{ut, sv\}) \cup \{e, st\}$, which is a solution of kNDHP with $x^{T_8} \in \mathscr{F}$, we get $b(e)+b(st) = b(sv)+b(ut)$. From (5.28) and (5.29), we have that $b(e) = \rho$. If $u \notin U_1$ and $v \in U_3$, then by considering the edge set $S_9 = (S_1 \setminus \{sv\}) \cup \{su, e\}$, we obtain along the same line that $b(e) = \rho$. If $u \in U_1$ and $v \notin U_3$, it follows by symmetry that $b(e) = \rho$. If $u \notin U_1$ and $v \notin U_3$, since the edge set $S_{10} = (S_1 \setminus \{su_1, u_1t\}) \cup \{su, e, vt\}$ is a solution of kNDHP with $x^{T_{10}} \in \mathscr{F}$, we get as before $b(e) = \rho$. Thus, we obtain

$$b(e) = \rho \text{ for all } e \in T \setminus (S_1 \cup \{st\}). \tag{5.30}$$

From (5.28)-(5.30), we have

$$b(e) = \begin{cases} \rho & \text{for all } e \in T, \\ 0 & \text{if not.} \end{cases}$$

Therefore, $b = \rho a$, and the proof is complete.

□

Theorem 26 *If $|D| = 1$, a 3-st-node-path-cut inequality* (5.4) *induced by a node subset $Z \subset V$, such that $|Z| \leq k-1$, and a partition $\pi = (V_0, ...V_4)$ of $V \setminus Z$, with $s \in V_0$ and $t \in V_4$, defines a facet of kNDHP$(G, 3)$ if and only if*

(1) $|V_0| = |V_4| = 1$,

(2) $|[s, V_1]| + |[V_3, t]| \geq k - |Z|$.

Proof. The idea of the proof is the same as that used in proving Theorem 24. We can also use the fact that a 3-*st*-node-path-cut inequality, $x(T_{G\setminus Z}) \geq k-|Z|$, for some 3-*st*-path-cut T and some node set $Z \subset V \setminus \{s, t\}$, is valid for $(k-|Z|)$NDHP$(G \setminus Z, 3)$ (recall that $(k - |Z|)$NDHP$(G \setminus Z, 3)$ is the polytope associated with the 3-hop-constrained st-path problem on the graph $G \setminus Z$).
Note as before that G is complete, $|Z| \leq k-1$ and $|V| \geq 2k+1$, then $G \setminus Z$ is complete with $|V \setminus Z| \geq k + 2$. By Corollary 7 the polytope $(k - Z)$NDHP$(G \setminus Z, 3)$ is full dimensional. Thus dim$((k - |Z|)$NDHP$(G \setminus Z, 3))= |E| - |\delta(Z)| - |E(Z)|$.

As $x(T_{G\setminus Z}) \geq k - |Z|$ defines a facet of $(k - |Z|)$NDHP$(G \setminus Z, 3)$, there exist $n' = |E| - |\delta(Z)| - |E(Z)|$ solutions of the $(k - |Z|)$NDHP on $G \setminus Z$. We will denote them by S'_i, $i = 1, ..., n'$, their incidence vectors are affinely independent and satisfy $x(T_{G\setminus Z}) = k - |Z|$. The st-paths of S'_i, $i = 1, ..., m$, are node-disjoint, hence they are solutions of the polytope $(k - |Z|)$NDHP$(G \setminus Z, 3)$.

The edge sets $S_i = S'_i \cup \delta(Z) \cup E(Z)$, for all $i \in \{1, ..., n'\}$, induce solutions of the kNDHP. Since S'_i, $i \in \{1, ..., n'\}$ is a solution of the $(k - |Z|)$NDHP on $G \setminus Z$, there exist $(k - |Z|)$ st-paths of length at most 3, in the subgraph of $G \setminus Z$ induced by S'_i.

We will denote them by H_l, $l = 1, ..., k-|Z|$. Moreover, as G is complete, the edges sz and zt, for all $z \in Z$, are in G, and the sets (s, z, t), $z \in Z$, form $|Z|$ st-paths of length 2 in G. Hence the paths H_l, $l = 1, ..., k - |Z|$ and (s, z, t), $z \in Z$, are node-disjoint.

Thus, the sets S_i, $i = 1, ..., n'$ induce n' solutions of the kNDHP on G. Furthermore, their incidence vectors satisfy $x(T_{G\setminus Z}) = k - |Z|$.

Let a' and b' be the restriction on $E \setminus (\delta(Z) \cup E(Z))$ of a and b, respectively. Thus, we have $a'x^{S_i} = \alpha$, for $i = 1, ..., n'$. Therefore, $b'x^{S_i} = \beta$, for $i = 1, ..., n'$. As x^{S_i}, $i = 1, ..., n'$, are affinely independent and $\alpha \neq 0$, it follows that $x^{S_i} \neq 0$, $i = 1, ...n'$, and hence, x^{S_i}, $i = 1, ..., n'$, are linearly independent. Consequently, a is the unique solution of the system $a'x^{S_i} = \alpha$, for $i = 1, ..., n'$. Let ρ be such that $\beta = \rho\alpha$. It then follows that $b' = \rho a'$. This implies that $b(e) = 0$ for all $e \in E \setminus T$.

Now we will show that $b(e) = 0$ for all $e \in \delta(Z) \cup E(Z)$. Let us denote the edges of $E(Z) \cup \delta(Z) \setminus \bigcup_{z\in Z} \{sz, zt\}$ by e_j, $j = 1, ..., |\delta(Z)| + |E(Z)| - 2|Z|$. Consider the edge set $\Omega_{n'+j} = S_{n'} \setminus \{e_j\}$, for $j = 1, ..., |\delta(Z)| + |E(Z)| - 2|Z|$. These sets clearly induce solutions of the kNDHP, and their incidence vectors satsify $x(T_{G\setminus Z}) = k - |Z|$. As $ax^{S_{n'}} = ax^{\Omega_{n'+j}} = \alpha$, it follows that $bx^{S_{n'}} = bx^{\Omega_{n'+j}} = \beta$. Hence, $b(e_j) = 0$ for $j = 1, ..., |\delta(Z)| + |E(Z)| - 2|Z|$.

Let S_1 be the set among $S_1, ..., S_{n'}$ containing the edge st. Such a set exists since, as noted before, $x^{S_i} \neq 0$ for $i = 1, ..., n'$. Let $u_1 \in V_1$, $u_2 \in V_L$ and $z \in Z$. Consider the edge set $S_0 = (S_1 \setminus \{sz\}) \cup \{su_1, u_1z\}$ and $S_0' = (S_1 \setminus \{zt\}) \cup \{sz, zu_2\}$. It clearly induces a solution of the kNDHP. Moreover we have $x(T_{G\setminus Z}) = k - |Z|$. Thus, $bx^{T_1} = bx^{T_0} = bx^{T_0'} = \beta$. As $b(su_1) = b(u_1z) = b(zu_2) = b(u_2t) = 0$, it follows that $b(sz) = b(zt) = 0$. Therefore $b = \rho a$, which ends the proof of the theorem. □

Note that Theorem 21, 22, 23 and 24 are valid for $L \geq 4$.

Lemma 10 *The double cut inequality induced by the node sets $V_0^1, V_0^2 \cup V_1, V_2, ..., V_{L+1}$ of $V \setminus Z$, $F \subseteq E$ and $\{s, t\} \in D$ with $s \in V_0^1$ and $t \in V_{L+1}$, can be written as*

$$\begin{aligned} &x(T_{G\setminus Z}) + x(\delta_{G\setminus Z}(V_0^1 \cup V_0^2)) + x(\delta_{G\setminus Z}(V_1)) \\ &+ x(\overline{E} \setminus F) - x(F) + |F| \geq 3(k - |Z|) + 1 \end{aligned} \tag{5.31}$$

where $T_{G\setminus Z}$ is the L-st-node-path-cut induced by the partition $(V_0^1, V_0^2 \cup V_1, V_2, ..., V_{L+1})$. Moreover, the double cut inequality (5.13) *is tight for a solution $\tilde{x} \in \mathbb{R}^E$ if and only if one of the following conditions holds.*

i) $\tilde{x}(\overline{E} \setminus F) - \tilde{x}(F) + |F| = 1$ and $\tilde{x}(T_{G\setminus Z}) = \tilde{x}(\delta_{G\setminus Z}(V_0^1 \cup V_0^2)) = \tilde{x}(\delta_{G\setminus Z}(V_1)) = k - |Z|$;

ii) $\tilde{x}(\overline{E} \setminus F) - \tilde{x}(F) + |F| = 0$ *and*

a) $\tilde{x}(T_{G\setminus Z}) = k - |Z| + 1$, $\tilde{x}(\delta_{G\setminus Z}(V_0^1 \cup V_0^2)) = k - |Z|$ *and* $\tilde{x}(\delta_{G\setminus Z}(V_1)) = k - |Z|$;

b) $\tilde{x}(T_{G\setminus Z}) = k - |Z|$, $\tilde{x}(\delta_{G\setminus Z}(V_0^1 \cup V_0^2)) = k - |Z| + 1$ *and* $\tilde{x}(\delta_{G\setminus Z}(V_1)) = k - |Z|$;

c) $\tilde{x}(T_{G\setminus Z}) = k - |Z|$, $\tilde{x}(\delta_{G\setminus Z}(V_0^1 \cup V_0^2)) = k - |Z|$ *and* $\tilde{x}(\delta_{G\setminus Z}(V_1)) = k - |Z| + 1$;

Proof. Let C be the double cut inducing inequality (5.13). Then inequality (5.13) can be written as

$$x(C \setminus \overline{E}) + x(\overline{E} \setminus F) \geq \frac{3(k - |Z|) - |F| + 1}{2}.$$

Thus we have

$$2x(C \setminus \overline{E}) + 2x(\overline{E}) - 2x(F) \geq 3(k - |Z|) - |F| + 1. \tag{5.32}$$

By summing the left hand side of the *L-st*-node-path-cut inequality induced by $T_{G\setminus Z}$ and the node-cut inequalities induced by $\delta_{G\setminus Z}(V_0^1 \cup V_0^2)$ and $\delta_{G\setminus Z}(V_1)$, we obtain

$$x(T_{G\setminus Z}) + x(\delta_{G\setminus Z}(V_0^1 \cup V_0^2)) + x(\delta_{G\setminus Z}(V_1)) = 2x(C \setminus \overline{E}) + x(\overline{E}). \tag{5.33}$$

By combining (5.32) and (5.33), we get

$$x(T_{G\setminus Z}) + x(\delta_{G\setminus Z}(V_0^1 \cup V_0^2)) + x(\delta_{G\setminus Z}(V_1)) + x(\overline{E}) - 2x(F) \geq 3(k - |Z|) - |F| + 1.$$

Therefore

$$x(T_{G\setminus Z}) + x(\delta_{G\setminus Z}(V_0^1 \cup V_0^2)) + x(\delta_{G\setminus Z}(V_1)) + x(\overline{E} \setminus F) - x(F) + |F| \geq 3(k - |Z|) + 1.$$

Hence, the double cut inequality (5.13) is equivalent to (5.31).

Suppose that the double cut inequality is tight for a solution $\tilde{x}$, that is

$$\tilde{x}(T_{G\setminus Z}) + \tilde{x}(\delta_{G\setminus Z}(V_0^1 \cup V_0^2)) + \tilde{x}(\delta_{G\setminus Z}(V_1)) + \tilde{x}(\overline{E} \setminus F) - \tilde{x}(F) + |F| = 3(k - |Z|) + 1$$

As $\tilde{x}(T_{G\setminus Z}) \geq k - |Z|$, $\tilde{x}(\delta_{G\setminus Z}(V_0^1 \cup V_0^2)) \geq k - |Z|$ and $\tilde{x}(\delta_{G\setminus Z}(V_1)) \geq k - |Z|$, we have that $\tilde{x}(\overline{E} \setminus F) - \tilde{x}(F) + |F| \leq 1$. Thus, if $\tilde{x}(\overline{E} \setminus F) - \tilde{x}(F) + |F| = 1$, we have that $\tilde{x}(T_{G\setminus Z}) = \tilde{x}(\delta_{G\setminus Z}(V_0^1 \cup V_0^2)) = \tilde{x}(\delta_{G\setminus Z}(V_1)) = k - |Z|$. If $\tilde{x}(\overline{E} \setminus F) - \tilde{x}(F) + |F| = 0$, then either $\tilde{x}(T_{G\setminus Z})$ or $\tilde{x}(\delta_{G\setminus Z}(V_0^1 \cup V_0^2))$ or $\tilde{x}(\delta_{G\setminus Z}(V_1))$ is equal to $k - |Z| + 1$ and the others are equal to $k - |Z|$, and the statement follows. □

Theorem 27 *The double cut inequality* (5.13) *defines a facet of kNDHP*$(G, 3)$ *only if*

i) $|V_0^1| = |V_4| = 1$,

ii) $|[V_0^1, V_0^2 \cup V_1] \cup [V_3, V_4] \cup [V_0^1, V_4]| \geq k - |Z|$.

Proof. i) Let C be the double cut inducing inequality (5.13). Using the following family of sets $\Pi = (V_0^1, V_0^2, V_1, V_2, ..., V_4)$. Suppose that $|V_0^1| > 1$, the case when $|V_{L+1}| > 1$ is similar. Consider the family of sets $\Pi' = \{\{s\}, V_0^1 \setminus \{s\}, V_0^2, V_1, ..., V_4\}$. Let C' be the double cut induced by Π' and F. Since $C = C' \cup [V_0^1 \setminus \{s\}, V_1]$, then the double cut inequality induced by Π is redundant with respect to the one induced by Π', and the trivial inequalities $x(e) \geq 0$ for all $e \in [V_0^1 \setminus \{s\}, V_1]$. Thus it does not define a facet.
ii) Let $\mathscr{F}$ be a facet defining double cut inequality and let $T_{G\setminus Z}$ be the 3-st-node-path-cut induced by the partition $(V_0^1, V_0^2 \cup V_1, V_2, ..., V_4)$. As $\mathscr{F}$ defines a facet different from the node-cut inequalities, there exists a solution $x_0 \in \mathscr{F}$ such that $x_0(\delta_{G\setminus Z}(V_0^1 \cup V_0^2)) \geq k - |Z| + 1$. Then by Lemma 10, $x_0(T) = k - |Z|$. Thus, x_0 induces a graph which contains exactly $k - |Z|$ node-disjoint 3-st-paths, P_1, ..., $P_{k-|Z|}$. Furthermore, each P_i, $i = 1, ..., k - |Z|$ intersects $T_{G\setminus Z}$ in only one edge. Thus either $P_i \cap [V_0, V_4] \neq \emptyset$ or P_i uses at least one edge between two non consecutive set of the partition $(V_0^1, V_0^2, V_1, V_2, ..., V_4)$. In the latter case, P_i must intersect either $[V_0^1, V_0^2 \cup V_1]$ or $[V_3, V_4]$ or both. Hence, we have that $|[V_0^1, V_0^2 \cup V_1] \cup [V_3, V_4] \cup [V_0^1, V_4]| \geq k - |Z|$. Which ends the proof. □

5.4 Conclusion

In this chapter, we have studied the k-Node-Disjoint Hop-Constrained Network Design Problem (kNDHP) when $L \in \{2, 3, 4\}$. We have introduced an integer programming formulation for the problem when $L \in \{2, 3\}$ and investigated the associated polytope. We have presented several classes of valid inequalities and presented conditions under which these inequalities define facets.

Chapter 6

Branch-and-Cut Algorithm for the kNDHP

Contents

6.1 Branch-and-Cut Algorithm for the kNDHP with $L = 3$ and $k \geq 3$

In this section, we present a Branch-and-Cut algorithm for the kNDHP when $L = 3$. First, we present the general framework of the algorithm and then present the separation procedures we have devised for the inequalities involved in the algorithm.

6.1.1 The general framework

Our algorithm starts by solving the linear relaxation of Formulation (5.7), that is,

$$\min\{cx \mid x \in \mathbb{R}_+^E \text{ satisfies } (5.1) - (5.6)\}. \tag{6.1}$$

Since inequalities (5.1), (5.2), (5.3) and (5.4) are exponential in number in (6.1), we solve this linear relaxation using the so-called *cutting plane method*. We recall that the cutting plane method finds an optimal solution of a linear program by solving a series of LPs, each of them containing a subset of the constraints of the original LP. For our purpose, the algorithm starts with an LP containing the cut constraints (5.1) induced by terminal nodes and the trivial inequalities (5.5) and (5.6)

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

Then, it iteratively adds the inequalities (5.1)-(5.4) that are violated by the solution x^* of the current LP. The cutting plane algorithm stops when all the inequalities (5.1)-(5.4) are satisfied by x^*. In this case, x^* is optimal for (6.1). For finding inequalities (5.1)-(5.4) that are violated by x^*, if there is any, we solve the so-called separation problem associated with these inequalities. Recall that *the separation problem* associated with a family of inequalities $\mathcal{F}$ and a solution $\overline{x}$ is to verify if $\overline{x}$ satisfies all the inequalities of $\mathcal{F}$, and if not, to exhibit at least one of them which is violated by $\overline{x}$. An algorithm solving a separation problem is called a *separation algorithm*.

At the end of the cutting plane algorithm, if x^* is integral, then it is optimal for the problem (5.7). If x^* is fractional, then we reinforce the linear relaxation of the problem by adding, if possible, further valid inequalities. For this, we also add the Steiner SP-partition inequalities (5.19), the double cut inequalities (5.11) and the Steiner partition inequalities (5.17) in the cutting plane algorithm. The separation of the inequalities used in the Branch-and-Cut algorithm are performed in the following order

1) *st*-cut and *L*-*st*-path-cut inequalities,

2) st-node-cut and L-st-node-path-cut inequalities (only for integral solutions),

3) Steiner SP-partition inequalities,

4) double cut inequalities,

5) Steiner partition inequalities.

Notice that the st-node-cut and L-st-node-path-cut inequalities are separated only for integral solutions. Indeed, as we will see in the next subsection, these two families of inequalities can be efficiently separated when the solution x^* is integral.

All the inequalities that are added during the Branch-and-Cut algorithm are considered as global (i.e., valid at every node of the Branch-and-Cut tree) and we may add several inequalities at each iteration. Furthermore, we proceed to the separation of a class of inequalities only when the separation of the previous class of inequalities has not found any violated inequalities.

In the following, we describe the separation algorithms we have devised for the inequalities (5.1)-(5.4), the Steiner SP-partition inequalities (5.19), the double cut inequalities (5.11) and the Steiner partition inequalities (5.17).

6.1.2 Separation procedures

6.1.2.1 Separation of st-cut and 3-st-path-cut inequalities

We discuss first the separation of the st-cut and 3-st-path-cut inequalities (5.1) and (5.2). We give the theorem below which shows that the separation problem of these inequalities reduces to computing a maximum flow in a special graph, and hence can be solved in polynomial time.

Theorem 28 *The separation problem of st-cut and 3-st-path-cut inequalities (5.1) and (5.2) reduces to computing maximum flows in a special graph and can be solved in* $O(|D||E|^2|V|)$ *time.*

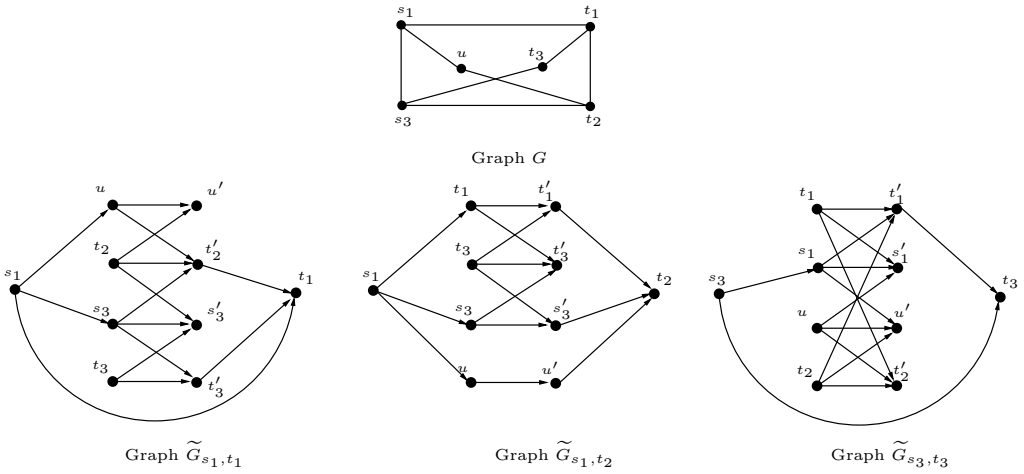

Figure 6.1: Construction of graphs $\widetilde{G}_{st}$ with $D = \{(s_1, t_1), (s_1, t_2), (s_3, t_3)\}$.

Proof. Let $\overline{x} \in \mathbb{R}^E$ be the solution for which we are separating the natural inequalities (5.1) and (5.2). To separate them, we consider the following graph transformation from [10] (see also [27]). Let $(s,t) \in D$ and let $V_{st} = V \setminus \{s,t\}$, V'_{st} be a copy of V_{st} and $\widetilde{V}_{st} = V_{st} \cup V'_{st} \cup \{s,t\}$. The copy in V'_{st} of a node $u \in V_{st}$ will be denoted by u'. From G and (s,t), we build the directed graph $\widetilde{G}_{st} = (\widetilde{V}_{st}, \widetilde{A}_{st})$. Its arc set $\widetilde{A}_{st}$ is obtained as follows. For an edge of the form $st \in E$, we add an arc (s,t) in $\widetilde{A}_{st}$. For each edge $su \in E$, $u \neq t$, (resp. $vt \in E$, $v \neq s$), we add in $\widetilde{A}_{st}$ an arc (s,u), $u \in V_{st}$ (resp. (v',t), $v' \in V'_{st}$). For each edge $uv \in E$, with $u, v \notin \{s,t\}$, we add two arcs (u,v') and (v,u') in $\widetilde{A}_{st}$, with $u, v \in V_{st}$ and $u', v' \in V'_{st}$. Finally, for each node $u \in V \setminus \{s,t\}$, we add an arc (u,u') in $\widetilde{A}_{st}$ (see Figure 6.1 for an illustration).

Notice that for each $(s,t) \in D$, $|\widetilde{V}_{st}| = 2|V| - 2$ and $|\widetilde{A}_{st}| = 2|E| - |\delta(s)| - |\delta(t)| + |[s,t]|$.

Bendali et al. [10] showed that there is a one-to-one correspondence between the st-cuts and the 3-st-path-cuts in G and the st-dicuts in $\widetilde{G}_{st}$ which do not contain arcs of the form (u,u'), for all $u \in V \setminus \{s,t\}$. Moreover, if each arc $a \in \widetilde{A}_{st}$, corresponding to an edge $e \in E$, is assigned the capacity $\widetilde{c}(a) = \overline{x}(e)$ and each arc of the form (u,u') is assigned an infinite capacity, then the weight of an st-cut or 3-st-path-cut in G with respect to $\overline{x}$ is the same as that of the corresponding st-dicut in $\widetilde{G}_{st}$ with respect to capacity vector $\widetilde{c}$. Thus, for a given $(s,t) \in D$, there is an st-cut or 3-st-path-cut inequality violated by $\overline{x}$ if and only if there is an st-dicut in $\widetilde{G}_{st}$ whose capacity is $< k$. Moreover, if there is a violated st-cut or 3-st-path-cut inequality induced by an edge set $C \subseteq E$, that is there is an st-dicut $\widetilde{C} \subseteq \widetilde{A}_{st}$ whose weight is $< k$, then the edges of C are those corresponding to the arcs of $\widetilde{C}$. Therefore, the separation problem of the

st-cut and the 3-st-path-cut inequalities reduces to computing a minimum st-dicut in $\widetilde{G}_{st}$ with respect to the capacity vector $\widetilde{c}$. By the Max Flow-Min Cut Theorem, this can be done by computing a maximum flow from s to t in $\widetilde{G}_{st}$.

Finally, the maximum flow computation in $\widetilde{G}_{st}$ can be handled by the Edmonds-Karp algorithm [35] which runs in $O(|\widetilde{A}_{st}|^2|\widetilde{V}_{st}|) = O(|E|^2|V|)$ time. Since this procedure is performed $|D|$ times (one for each demand), the whole separation algorithm can be implemented to run in $O(|D||E|^2|V|)$ time, and hence is polynomial. □

Our separation algorithm for st-cut and 3-st-path-cut inequalities is based on Theorem 28. It starts, for each demand $(s,t) \in D$, by building the graph $\widetilde{G}_{st}$ and then, computing a minimum weight st-dicut, say $\widetilde{C}$, w.r.t. weight vector $\widetilde{c}$. If the weight of such a st-dicut is $< k$, then the edge set C of G corresponding to the arcs of $\widetilde{C}$ corresponds to either a st-cut or a 3-st-path-cut which induces a violated inequality. The separation algorithm stops when it finds, for a given demand, a violated inequality or when all the demands have been considered without finding any violated inequality. From Theorem 28, this algorithm solves the separation problem of inequalities (5.1) and (5.2) in polynomial time.

6.1.2.2 Separation of st-node-cut and 3-st-node-path-cut inequalities

Now, we discuss the separation problem of st-node-cut and 3-st-node-path-cut inequalities (5.3) and (5.4). As mentionned before, we consider the separation problem of these inequalities only in the case where the considered solution $\overline{x} \in \mathbb{R}^E$ is integral. We restrict to this case because first, in order to guarantee that the B&C algorithm is correct, it is enough to separate the st-node-cut and 3-st-node-path-cut inequalities for integral solutions, only. Also, as we will see below, inequalities (5.3) and (5.4) can be separated in polynomial time when $\overline{x}$ is integral. Thus, suppose that $\overline{x}$ is integral, and w.l.o.g., satifies all the st-cut and 3-st-path-cut inequalities (5.1) and (5.2). Under these assumptions, solving the separation problem of inequalities (5.3) and (5.4) consists in saying if $\overline{x}$ is feasible for the kNDHP or not. Indeed, on one hand, if $\overline{x}$ is feasible, then it satisfies all the st-node-cut and 3-st-node-path-cut inequalities. On the other hand, if $\overline{x}$ is not feasible for the kNDHP, then for some demand $(s,t) \in D$, there exists a node set $Z \subseteq V$ such that $|Z| \leq k-1$, and an st-node-cut or a 3-st-node-path-cut C of G w.r.t. Z such that $\overline{x}(C) < k - |Z|$. Thus, the aim of our separation algorithm is to say if the subgraph of G induced by $\overline{x}$ contains, for each $(s,t) \in D$, k node-disjoint 3-st-paths, and if not, find node set Z and an st-node-cut or a 3-st-node-path-cut w.r.t.

Z which induces an inequality (5.3) or (5.4) violated by $\overline{x}$.

In the sequel, we present the main ingredient and main results that will support our algorithm. First, consider a demand $(s,t) \in D$ and let $\widetilde{G}_{st}$ be the directed graph described in the proof of Theorem 28. Also let $\widetilde{c}$ be the associated weight vector. Remark that, as G is simple, that is does not contain parallel edges, if two 3-st-paths P_1 and P_2 are not node-disjoint, then they are of the form $P_1 = (s,u,v,t)$ and $P_2 = (s,v,z,t)$ with $u,v,z \in V \setminus \{s,t\}$ and $u \neq v \neq z \neq u$. These two paths correspond in $\widetilde{G}_{st}$ to paths (s,u,v',t) and (s,v,z',t). Conversely, two paths (s,u,v',t) and (s,v,z',t) of $\widetilde{G}_{st}$ correspond to two paths (s,u,v,t) and (s,v,z,t) which are not node-disjoint. Consequently, when $\overline{x}$ is not feasible for the kNDHP, any maximum set of disjoint st-dipaths of the graph $\widetilde{G}_{st}$, for some demand $(s,t) \in D$, will contain two paths of the form (s,u,v',t) and (s,v,z',t), with $u,v,z \in V \setminus \{s,t\}$ and $u \neq v \neq z \neq u$.

Now we give the following procedure, that we call *Procedure BuildZ*, which aims to build a node $Z \subseteq V$. The procedure works as follows. Let $Z \subseteq V$ be a node set of G and denote by $\widetilde{Z}$ the nodes of $\widetilde{G}_{st}$ corresponding to those of Z, that is $\widetilde{Z} = \{u, u' \text{ such that } u \in Z\}$. At the begining of the procedure $Z = \emptyset$. Now, compute a maximum flow from s to t in the graph $\widetilde{G}_{st} \setminus \widetilde{Z}$, with each arc $a \in \widetilde{A}_{st}$ having the capacity $\widetilde{c}(a)$. This gives a maximum set $\widetilde{\mathcal{P}}$ of arc-disjoint st-dipaths in $\widetilde{G}_{st} \setminus \widetilde{Z}$. Remark that some of those paths may correspond to non node-disjoint 3-st-paths of G, that is they are of the form (s,u,v',t) and (s,v,z',t). Let $\widetilde{\mathcal{P}}'$ be the set of these paths. Also let $\mathcal{P}'$ be the set of paths of G corresponding to those of $\widetilde{\mathcal{P}}'$ and $U \subseteq V$ the set of nodes of G which are shared by two paths of $\mathcal{P}'$. Now, add to Z the nodes of U and repeat this procedure until $|Z| \geq k$ or $U = \emptyset$. It should be noticed that when $U = \emptyset$, the arc-disjoint st-dipaths obtained by the computation of the maximum flow in $\widetilde{G}_{st} \setminus \widetilde{Z}$ correspond to node-disjoint 3-st-paths of $G \setminus Z$.

The identification of the nodes of U can be easily done by simply watching, for each node $u \in V \setminus Z$, the arcs entering and leaving nodes u and u' with flow value 1 and 0. Namely, consider a node $v \in U$. This means that after the maximum flow computation, there are two paths (s,u,v',t) and (s,v,z',t). Since the arc capacities are either 0 or 1, this means that

- the flow value on arc (s,v) is 1,
- the flow value on arc (v,z') is 1, for some $z \in V \setminus \{s,t,v\}$,

- the flow value on arc (v, v') is 0,
- the flow value on arc (v', t) is 1.

Figure 6.2 illustrates the above remark. The solid lines represent arcs having flow value 1 and dashed lines represent arcs with flow value 0. The flow value of the arcs represented by dotted lines may be 0 or 1.

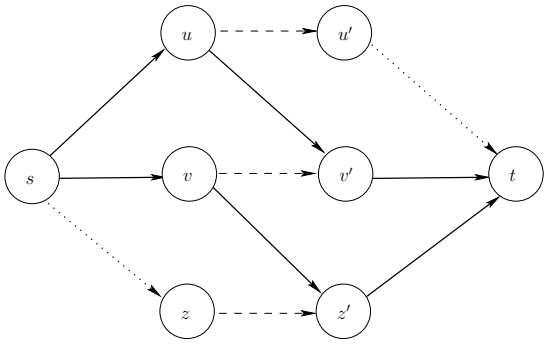

Figure 6.2: Two st-dipaths of $\widetilde{G}_{st} \setminus \widetilde{Z}$ inducing non node-disjoint 3-st-paths in $G \setminus Z$.

Observe that using Edmonds-Karp algorithm's [35] for each maximum flow computation, the identification of the node set U at each iteration can be implemented to run in $O(|E|^2|V| + |V|^2) = O(|E|^2|V|)$. Also, Procedure BuildZ runs in at most k iterations. Therefore, Procedure BuildZ builds a node set Z in $O(k|E|^2|V|)$ time.

Now, we give the following theorem.

Theorem 29 *Let $\overline{x} \in \{0,1\}^E$ be an integral solution and $(s,t) \in D$. Assume that $\overline{x}$ satisfies all the st-cut and 3-st-path-cut inequalities. Let $Z \subseteq V$ be the node set obtained by the application of Procedure BuildZ. Then, there is a st-node-cut or a 3-st-node-path-cut inequality violated by $\overline{x}$ if and only if $|Z| \leq k-1$ and the minimum weight of a cut of $\widetilde{G}_{st} \setminus \widetilde{Z}$ is $< k - |Z|$.*

Proof. Suppose that $|Z| \leq k-1$ and a minimum weight st-dicut $\delta^+_{\widetilde{G}_{st} \setminus Z}(\widetilde{W})$, w.r.t weight vector $\widetilde{c}$, is such that $\widetilde{c}(\delta^+_{\widetilde{G}_{st} \setminus Z}(\widetilde{W})) < k - |Z|$. As shown by Bendali et al. [10], this st-dicut corresponds to an st-cut or a 3-st-path-cut of $G \setminus Z$. Let C be this st-cut or 3-st-path-cut. Moreover, we have that $\overline{x}(C) = \widetilde{c}(\delta^+_{\widetilde{G}_{st} \setminus \widetilde{Z}}(\widetilde{W})) < k - |Z|$. Thus, C

induces a st-node-cut or 3-st-node-path-cut inequality violated by $\overline{x}$.

Now suppose that $|Z| \geq k$. By the construction procedure of Z, the subgraph of G induced by $\overline{x}$ contains edges of the form su and ut, for every node $u \in Z$, which form $|Z|$ paths (s, u, t). Clearly, these paths are node-disjoint, and hence, the subgraph of G induced by $\overline{x}$ contains $|Z| \geq k$ node-disjoint 3-st-paths. Consequently, $\overline{x}$ satisfies all the st-node-cut and 3-st-node-path-cut inequalities.

Finally, suppose that $|Z| \leq k-1$ and $f = \widetilde{c}(\delta^+_{\widetilde{G}_{st} \setminus \widetilde{Z}}(\widetilde{W})) \geq k - |Z|$, where $\widetilde{W}$ induces a minimum weight st-dicut of $\widetilde{G}_{st} \setminus \widetilde{Z}$, w.r.t. to weight vector $\widetilde{c}$. As before, by the construction procedure of Z, the subgraph of G induced by $\overline{x}$ contains $|Z|$ node-disjoint 3-st-paths. Also, the construction of Z implies the graph $\widetilde{G}_{st} \setminus \widetilde{Z}$ contains f arc-disjoint st-dipaths which correspond to f node-disjoint 3-st-paths of G. Moreover, these later node-disjoint 3-st-paths are node-disjoint from those induced by Z. Consequently, the subgraph of G induced by $\overline{x}$ contains $f + |Z| \geq k - |Z| + |Z| = k$ node-disjoint 3-st-paths, which implies that $\overline{x}$ satisfies all the st-node-cut and 3-st-node-path-cut inequalities. □

Now we describe our separation algorithm for st-node-cut and 3-st-node-path-cut inequalities when $\overline{x}$ is integral and satisfies all the st-cut and 3-st-path-cut inequalities. For each demand $(s, t) \in D$, we build the graph $\widetilde{G}_{st}$ and let $\widetilde{c}$ be the associated weight vector. Then we build, using Procedure BuildZ, the node set Z and let f be the weight of a minimum weight cut of $\widetilde{G}_{st} \setminus \widetilde{Z}$, w.r.t. weight vector $\widetilde{c}$. If $|Z| \leq k-1$ and $f < k - |Z|$, then, by Theorem 29, there is an st-node-cut or a 3-st-node-path-cut $C \subseteq E$ which induces an inequality (5.3) or (5.4) violated by $\overline{x}$. If $|Z| \geq k$ or $f \geq k - |Z|$, then we move to another demand. The algorithm stops when it has found a violated inequality (5.3) or (5.4) for some demand $(s, t) \in D$ or when all the demands have been explored without finding any violated inequality.

The separation algorithm can be implemented to run in $O(|D|k|E|^2|V|)$ time, which is polynomial.

6.1.2.3 Separation of double cut, Steiner SP-partition and partition inequalities

Now we consider the separation of Inequalities (5.11), (5.19) and (5.17). For our purpose, we look for those inequalities (5.11), (5.19) and (5.17) defined with a node set $Z = \emptyset$. To separate them, we use the separation heuristics developed in [26].

The heuristic developed for the double cut inequalities is implemented to run in $O\left(|V|^3 \log |V| \frac{(2|V|+|D_{source}|+|D_{dest}|)^2}{(|V|-1)(|V|+|D_{source}|+|D_{dest}|)}\right)$ time. Here D_{source} and D_{dest} denote the sets of nodes which are, respectively, the source and destination in a demand, which is polynomial.

For SP-partition inequalities (5.19), the heuristic proposed by [26] is implemented to run in $O(|V||E| + |D|)$, while the separation heuristic for partition inequalities (5.17) proposed by [26] is implemented to run in $O(|V||E| + |R|^2(|E| + |D|))$, where R is the set of terminal nodes.

Clearly, the three heuristics run in polynomial time.

6.1.3 Computational Results

We have implemented our Branch-and-Cut algorithm in C++, using CPLEX 12.5 and Concert Technology [3]. It was tested on a Xeon Quad-Core E5507 machine with a 2.27 GHz processor and 8GB RAM, running under Linux. The maximum CPU time has been fixed to 5 hours. Each instance is composed of a graph from TSPLIB [2] and a set of demands. TSPLIB graphs are complete Euclidean graphs, that is each node is assigned coordinates in the plane, and the weight of each edge is given by the Euclidean distance between its endnodes. The demands used in the instances are randomly generated. Each set of demands is either rooted, that is, of the form $\{(s, t_i) : i = 1, ..., d\}$ (s is the root node of the demands), or arbitrary.

The computational results are given in Tables 6.1-6.6. Each instance is described by the number of nodes of the graph and the number of demands. The number of nodes is preceded either by “r" if the demands are rooted or “a" if they are not rooted. The entries of the various tables presented below are:

$\|V\|$	:	the number of nodes of the graph,
$\|D\|$	:	the number of demands,
C-LPC	:	the number of generated *st*-cut and 3-*st*-path-cut inequalities,
NC-NLPC	:	the number of generated *st*-node-cut and 3-*st*-node-path-cut inequalities,
SP	:	the number of generated Steiner SP-partition inequalities,
DC	:	the number of generated double cut inequalities,
DC	:	the number of generated Steiner partition inequalities,
COpt	:	value of the best upper bound obtained,
Gap	:	the relative error between the best upper bound and the lower bound obtained at the root node of the Branch-and-Cut tree,
NSub	:	the number of nodes in the Branch-and-Cut tree,
CPU	:	total CPU time of the first run in hours:min.sec.

Note that for some instances, the algorithm spends all the CPU time (5 hours) without finding any feasible solution. In this case, the best upper bound (COpt) and the error with the lower bound achieved at the root node of the Branch-and-Cut tree (Gap) are indicated with "-".

Our first series of experiments concerns the kNDHP with $k = 3$ and $L = 3$. The results are given in Tables 6.1 and 6.2.

Table 6.1: Results for $k = 3$, $L = 3$ and rooted demands.

$\|V\|$	$\|D\|$	C-LPC	NC-NLPC	SP	DC	P	COpt	Gap	NSub	CPU
r 21	15	11775	74	10	0	0	5526	9.23	2265	00:04:46
r 21	17	22356	228	0	0	0	5939	9.4	4518	00:18:24
r 21	20	71354	116	0	0	0	6466	9.54	17673	03:10:39
r 30	15	15599	264	14	0	0	10109	6.87	1521	00:12:06
r 30	20	58659	1516	12	0	0	11376	8.41	15280	05:00:00
r 30	25	80999	615	18	0	0	12661	12.33	14281	05:00:00
r 48	20	51038	1632	26	0	0	18337	18.1	6133	05:00:00
r 48	30	66277	898	10	0	0	25437	28.68	5305	05:00:00
r 48	40	69242	257	2	0	0	31693	30.17	5628	05:00:00
r 52	20	49717	1674	22	0	0	11170	9.15	5707	05:00:00
r 52	30	62698	1692	18	0	0	14626	17.11	3845	05:00:00
r 52	40	68794	1024	16	0	0	17920	21.86	4953	05:00:00
r 52	50	77808	142	0	0	0	20873	24.49	4397	05:00:00

Table 6.2: Results for $k = 3$, $L = 3$ and arbitrary demands.

$\|V\|$	$\|D\|$	C-LPC	NC-NLPC	SP	DC	P	COpt	Gap	NSub	CPU
a 21	10	56593	2	0	483	0	6680	8.66	9191	05:00:00
a 21	11	29325	2	0	375	1	6770	6.8	2614	00:57:32
a 30	10	44057	25	18	38	0	10354	6.64	13274	05:00:00
a 30	15	53545	86	0	462	0	13936	11.69	6399	05:00:00
a 48	15	34047	0	0	20	0	-	-	1119	05:00:00
a 48	20	28329	0	2	10	0	-	-	229	05:00:00
a 48	24	23975	0	0	11	0	-	-	103	05:00:00
a 52	20	30157	108	6	41	0	-	-	1735	05:00:00
a 52	26	24217	0	0	96	0	-	-	307	05:00:00

We can see that for the rooted instances (Table 6.1), the algorithm has solved to optimality 4 instances out of 13, with graphs having up to 30 nodes and with 15 demands, and the CPU time varying from 4 minutes to 18 minutes. The gap achieved between the best upper bound (that is, the optimal solution) and the lower bound at the root node of the Branch-and-Cut tree (Gap) is relatively small (less than 10%) for these instances. For the instances that have not been solved to optimality, the value of the gap are also relatively small: less than 10% for 5 of them and less 31% for the 4 other instances. Table 6.1 also shows that a very large number of st-cut and 3-st-path-cut inequalities have been generated during the resolution. Also, a large number of st-node-cut and 3-st-node-path-cut inequalities have been generated for all the instances. We can also see that several Steiner SP-partition have been generated but no double cut and Steiner partition inequalities have been generated.

For the arbitrary demands, the algorithm has solved to optimality only one instance (d-21-11) over nine and has spent all the CPU for the other instances. Also, it has not found even a feasible solution for 5 instances. For the instances r-21-10, d-30-10 and d-30-15, that have not been solved to optimality, the gap between the lower bound at the root node of the Branch-and-Cut tree and the best upper bound is less than 12%. We can also see, as for the rooted instances, that a very large number of st-cut and 3-st-path-cut inequalities have been generated, but less st-node-cut and 3-st-node-path-cut inequalities have been generated. We can also notice that some Steiner SP-partition and a quite large number of double cut inequalities have been generated in the resolution.

Our next series of experiments concerns the kNDHP with $k = 4$ and $L = 3$. The results are given in Tables 6.3 and 6.4. Notice that in this case, the Steiner SP-partition and Steiner partition inequalities are not included in the Branch-and-Cut algorithm

as they are redundant w.r.t. *st*-cut inequalities. Thus, the corresponding columns in Tables 6.3 and 6.4 are omitted.

Table 6.3: Results for $k = 4$, $L = 3$ and rooted demands.

$\|V\|$	$\|D\|$	C-LPC	NC-NLPC	DC	COpt	Gap	NSub	CPU
r 21	15	4923	52	0	7322	4.57	1078	00:00:50
r 21	17	5732	24	0	7826	4.56	1186	00:01:06
r 21	20	35317	9	0	8556	5.32	17991	01:11:08
r 30	15	48473	2266	0	14315	6.66	10718	05:00:00
r 30	20	23784	0	0	15041	4.19	5664	00:35:22
r 30	25	54445	595	0	16379	5.93	11631	05:00:00
r 48	20	40090	1784	0	26131	20.86	6929	05:00:00
r 48	30	43988	621	0	29806	16.86	4140	05:00:00
r 48	40	51107	232	0	40037	24.77	4302	05:00:00
r 52	20	39125	1346	0	15480	8.72	5106	05:00:00
r 52	30	42750	2760	0	20976	20.28	5192	05:00:00
r 52	40	49499	831	0	24343	21.52	4865	05:00:00
r 52	50	56313	282	0	26541	17.92	4472	05:00:00

Table 6.4: Results for $k = 4$, $L = 3$ and arbitrary demands.

$\|V\|$	$\|D\|$	C-LPC	NC-NLPC	DC	COpt	Gap	NSub	CPU
a 21	10	50711	108	858	9339	10.37	9674	05:00:00
a 21	11	55432	127	703	9864	12.52	10221	05:00:00
a 30	10	36595	116	152	14582	6.3	9817	05:00:00
a 30	15	39442	37	319	18961	10.19	4593	05:00:00
a 48	15	24750	0	20	-	-	589	05:00:00
a 48	20	20007	0	4	-	-	137	05:00:00
a 48	24	16095	0	2	-	-	47	05:00:00
a 52	20	21556	0	54	-	-	867	05:00:00
a 52	26	14635	0	35	-	-	215	05:00:00

The results of Table 6.3 show that for the rooted instances, 4 instances over 13 have been solved to optimality. For the other instances, the gap is less than 9% for three instances and less than 22% for six instances. The results also show that a very large number of *st*-cut and 3-*st*-path-cut inequalities are generated while a large number of *st*-node-cut and 3-*st*-node-path-cut inequalities are generated for all the instances. Also, no double cut inequalities are generated for all the instances we have considered.

For arbitrary demands (Tables 6.4), all the instances have not been solved to optimality within the CPU time limit. Also, for five instances (from d-48-15 to d-52-26) over nine, the algorithm has not found a feasible solution. For the others, the gap is less than 13%. Contrarily to rooted demands, a quite large number of double cut inequalities have been generated.

Now we turn out attention to the resolution of the kNDHP with $k = 5$ and $L = 3$. The results are given in Tables 6.5 and 6.6 below.

Table 6.5: Results for $k = 5$, $L = 3$ and rooted demands.

$\|V\|$	$\|D\|$	C-LPC	NC-NLPC	SP	DC	P	COpt	Gap	NSub	CPU
r 21	15	8854	173	0	0	0	9560	3.24	3283	00:03:45
r 21	17	23490	804	0	0	0	10235	3.93	19537	00:54:09
r 21	20	31742	958	0	0	0	11095	4.1	59210	03:18:34
r 30	15	80055	6007	0	0	0	19624	10.01	25045	05:00:00
r 30	20	87973	838	0	0	0	20444	5.78	19283	05:00:00
r 30	25	77565	670	0	0	0	21604	5.31	23220	05:00:00
r 48	20	55964	4334	0	0	0	32753	18.53	11308	05:00:00
r 48	30	59897	1414	0	0	0	41200	22.3	9979	05:00:00
r 48	40	68991	319	0	0	0	48194	20.02	7758	05:00:00
r 52	20	55456	5330	0	0	0	28222	34.95	10387	05:00:00
r 52	30	56528	3283	0	0	0	31443	31.67	9388	05:00:00
r 52	40	61465	1330	0	0	0	30645	20.24	7997	05:00:00
r 52	50	77724	275	0	0	0	33994	17.27	8225	05:00:00

Table 6.6: Results for $k = 5$, $L = 3$ and arbitrary demands.

$\|V\|$	$\|D\|$	C-LPC	NC-NLPC	SP	DC	P	COpt	Gap	NSub	CPU
a 21	10	71158	279	0	645	0	11703	7.8	23196	05:00:00
a 21	11	74831	516	0	1097	1	12533	11.58	24980	05:00:00
a 30	10	49032	305	4	0	0	18613	2.94	20559	05:00:00
a 30	15	56285	242	0	589	0	24043	8.29	9227	05:00:00
a 48	15	37209	91	0	8	0	-	-	2479	05:00:00
a 48	20	28574	0	0	2	0	-	-	303	05:00:00
a 48	24	24246	0	0	0	0	-	-	113	05:00:00
a 52	20	33492	29	0	0	0	30754	17.31	2065	05:00:00
a 52	26	25465	0	0	20	0	-	-	347	05:00:00

We can see from Table 6.5, that for the rooted instances, the algorithm has solved to optimality three instances over 13. For the other ten instances, the gaps is less than 10%, for only three of them. For the remaining instances, the gaps are between 10% and 35%. Also, we notice that a large number of st-cut, 3-st-path-cut, st-node-cut and 3-st-node-path-cut inequalities are generated. However, no Steiner SP-partition, double cut and Steiner partition inequalities are generated. For the arbitrary demands (Table 6.6), all the instances have not been solved to optimality, and, for four instances, the algorithm has not found a feasible solution. We also notice that few Steiner SP-partition and Steiner partition inequalities and a quite large number of double cut inequalities have been generated.

In these experiments, we have also tried to check the impact of the different classes of inequalities we have considered in our algorithm. As we can see in the various tables,

Steiner SP-partition and double cut inequalities are generated in quite large number, and very few Steiner partition inequalities are found. We also observe that in the three cases $k = 3, 4, 5$, the double cut inequalities are not generated when the demands are rooted, and several of them are generated when the demands are arbitrary. In contrast with double cut inequalities, Steiner SP-partition inequalities are mainly generated when the demands are rooted, and few of them are generated for arbitrary demands. This observation can be compared with those of Diarrassouba et al. [28] who devised a Branch-and-Cut algorithm for the kNDHP with $k = 2$. In their experiments, they showed that the double cut inequalities were mainly generated when the demands are arbitrary. This suggests that the double cut inequalities (5.11) are mainly involved in the resolution of the problem when the demands are arbitrary, and when the demands are rooted, Steiner SP-partition inequalities may play an important role in solving the problem.

To conclude this experimental study, we have checked the impact of the connectivity on the resolution of the problem. Such a comparison has been made by Bendali et al. [9], for the k-edge-connected subgraph problem, and by Diarrassouba et al. [27], for the kEHDP, that is the hop-constrained survivable network design problem in which the L-st-paths are required to be edge-disjoint, for each demand $(s, t) \in D$. In both studies, the computational results suggest that the problem becomes easier to solve when the connectivity increases. However, for the kNDHP, our computational results do not allow to make the same conclusion. Indeed, by comparing Tables 6.1, 6.3 and 6.5, we can see that most of the instances that have not been solved to optimality for $k = 3$ have also not been solved to optimality for $k = 4$ and $k = 5$. Also, the number of nodes in the Branch-and-Cut tree is quite large in the three cases. Also, the different gaps achieved do not allow to see if the problem becomes easier when k increases. In fact, for some instances, like r-21-20, the gap decreases as k increases, while for some other instances, like r-30-15, the gap is better when $k = 4$ than when $k = 3$ and $k = 5$. Even, for some instances, like r-48-20, the gaps increase as k increases. The observations are the same for the arbitrary demands, that is, we cannot conclude from Tables 6.2, 6.4 and 6.6 that the resolution of the kNDHP becomes easier when the connectivity k increases.

6.2 Branch-and-Cut Algorithm for the kNDHP with $L = 4$ and $k = 2$

In this section, we present a Branch-and-Cut algorithm for the kNDHP when $L = 4$ and $k = 2$, based on the formulation presented in [50]. The formulation uses inequalities (5.1)-(5.6). First, we present the general framework of the algorithm and then present the separation procedures we have devised for the inequalities involved in the algorithm.

6.2.1 The general framework

The general framework of the algorithm is similar to the one presented before. To reinforce the linear relaxation of this problem we add the rooted partition inequalities (5.25) in the cutting plane algorithm. The separation of the inequalities used in the Branch-and-Cut algorithm are performed in the following order

1) st-cut and st-node-cut inequalities,
2) rooted partition inequalities,
3) L-st-path-cut inequalities and L-st-node-path-cut inequalities (only for integral solutions).

We apply the rooted partition inequalities (5.25) for the rooted 2NDHP (that is, when the set of demands is rooted in a single node) and do not apply them when arbitrary demands are considered.
Notice that the L-st-path-cut and L-st-node-path-cut inequalities are separated only for integral solutions. Indeed, as we will see in the next subsection, these two families of inequalities can be efficiently separated when the solution x^* is integral.
In the following, we describe the separation algorithms we have devised for the inequalities (5.1)-(5.4) and the rooted partition inequalities (5.25).

6.2.2 Separation procedures

6.2.2.1 Separation of st-cut inequalities and st-node-cut

It is well known that the separation of the st-cut inequalities (5.1) (resp. the st-node-cut inequalities (5.3)) reduces to computing a minimum weight cut in G (resp. in $G \setminus z$

for all $z \in V \setminus \{s,t\}$) with respect to weight vector $\overline{y}$. Indeed, there is a violated cut inequality (5.1) (resp. st-node-cut inequality (5.3)) if and only if the minimum weight of a cut, w.r.t. weight vector $\overline{y}$, is < 2 (resp. < 1). One can compute a minimum weight cut in polynomial time by using any minimum cut algorithm, and especially by using the Gomory-Hu algorithm [40] which computes the so-called Gomory-Hu cut tree. This algorithm consists in $|V| - 1$ maximum flow computations.

6.2.2.2 Separation of 4-st-path-cut and 4-st-node-path-cut inequalities

Now, we discuss the separation problem of 4-st-path-cut and 4-st-node-path-cut inequalities (5.2) and (5.4). As mentionned before, we consider the separation problem of these inequalities only in the case where the considered solution $\overline{x} \in \mathbb{R}^E$, is integral. The idea is similar to the one presented in the proof of the formulation in [50]. Consider an edge subset $F \subseteq E$, and let G_F be the graph induced by F. First we compute a Dijkstra algorithm to obtain a shortest st-path (in number of hops), say P_0, in G. If $|P_0| > 4$, then we detect a violated 4-path-cut inequality. We define V_i, $i = 0, ..., 4$, as the subset of nodes at distance i from s in G, and $V_5 = V \setminus (\bigcup_{i=0}^{4} V_i)$. We add the corresponding 4-path-cut inequality induced by the partition $(V_1, ..., V_5)$ to the LP. If $|P_0| \leq 4$, then we look for a second shortest path in $G \setminus \{st\}$, say P_1, such that P_0 and P_1 are node-disjoint. If $|P_1| \leq 4$, then F induces a solution for the 2NDHP. If $|P_1| > 4$, there are two cases. The first case is when $|P_0| = 1$, that is $P_0 = (st)$, we define a 4-path-cut inequality in the same way as in the previous case, and we add the violated inequality to the LP. The second case is when $|P_0| > 1$, in that case we remove the nodes of P_0, say $v_i^{P_0}$, $i = 1, ..., |P_0|$, one by one, then we define the corresponding 4-node-path-cuts in $G \setminus v_i^{P_0}$ in the same way, and add them to the LP.

6.2.2.3 Separation of rooted partition inequalities

To separate inequalities (5.25), we use the separation heuristic presented in [49]. This heuristic has been implemented to run in polynomial time.

6.2.3 Computational results

The same computational environment presented in the previous section is used for these experiments. Note that the rooted partition inequalities (5.25) are only used for

the instances with rooted demands.

The computational results are given in Tables 6.7 and 6.8. The entries of these two tables are the same as those of Section 6.1.3, except for Table 6.8, for which we add the entry

RP : the number of generated rooted partition inequalities.

Table 6.7: Results for $k = 2$, $L = 4$ and arbitrary demands.

$\|V\|$	$\|D\|$	C-NC	LPC-NLPC	COpt	Gap	NSub	CPU
a 5	2	1	0	2314	0	1	0:00:01
a 10	3	17	15	2358	1.64	23	0:00:01
a 10	4	30	175	2773	10.1	186	0:00:01
a 10	5	15	326	3219	11.14	615	0:00:01
a 14	5	46	102	3326	7.18	356	0:00:01
a 14	7	195	1604	3796	9.72	6439	0:00:08
a 17	8	3589	45507	3079	31.71	608368	5:00:00
a 21	10	2361	37423	4720	40.59	334898	5:00:00
a 21	11	2893	36324	4770	41.44	334084	5:00:00
a 48	10	26795	19456	57349	87.33	154736	5:00:00
a 48	15	22343	32540	-	-	145813	5:00:00
a 48	24	17236	30632	-	-	245307	5:00:00
a 52	10	33319	16521	19769	74.6	107555	5:00:00
a 52	15	29114	15508	32592	81.2	124780	5:00:00
a 52	20	14288	33394	-	-	162161	5:00:00
a 52	26	10778	33464	-	-	198811	5:00:00

We can see that for the rooted demands (Table 6.8), the algorithm has solved to optimality 7 instances out of 19 within the time limit. We can observe that the gaps obtained are quite large for most of the instances, but it is less than 30% for the relatively small instances. Table 6.8 also shows that a very large number of st-cut and 3-st-path-cut inequalities have been generated during the resolution, and a large number of st-node-cut and 3-st-node-path-cut inequalities have been generated for all the instances. We can also see that several rooted partition inequalities have been generated for some instances.

Table 6.8: Results for $k = 2$, $L = 4$ and rooted demands.

$\|V\|$	$\|D\|$	C-NC	LPC-NLPC	RP	COpt	Gap	NSub	CPU
r 10	5	41	24	36	2358	1.35	17	0:00:01
r 10	7	18	18	22	2848	4.09	23	0:00:01
r 10	9	15	565	19	3481	12.28	588	0:00:01
r 14	5	520	444	0	2601	7.9	383	0:00:01
r 14	7	862	4177	882	2998	17.61	5820	0:00:36
r 14	10	622	3481	668	3662	7.04	10540	0:00:31
r 17	16	3202	83928	0	2700	22.65	380463	5:00:00
r 21	7	717	1272	0	1789	6.59	425	0:00:02
r 21	10	15721	35009	0	2570	25.91	431618	5:00:00
r 30	29	7391	200914	0	13549	54.59	163377	5:00:00
r 48	10	60181	46849	0	27557	77.47	81057	5:00:00
r 48	15	69824	71232	0	37962	82.38	77855	5:00:00
r 48	20	89805	86653	0	27814	72.68	70272	5:00:00
r 48	30	87560	158432	0	38629	79.03	105848	5:00:00
r 52	10	85786	59630	0	23916	83.1	62374	5:00:00
r 52	20	116638	109693	0	19426	72.81	50496	5:00:00
r 52	30	111091	174649	0	22143	72.07	63565	5:00:00
r 52	40	43333	262234	0	22378	70.09	109805	5:00:00
r 52	50	18217	281265	0	20731	64.4	120560	5:00:00

For the arbitrary demands, the algorithm has solved to optimality 6 instances, with graphs having up to 14 nodes and with 7 demands, and has spent all the CPU for the other instances. We can also see, as for the rooted demands, that a very large number of st-cut and 3-st-path-cut inequalities have been generated, and as much st-node-cut and 3-st-node-path-cut inequalities have been generated. We also note that the number of nodes in the Branch-and-Cut tree is quite large for the two types of demands. Also, the different gaps achieved are important for the big instances. Finally, we notice that for the arbitrary demands, the algorithm ran out of memory for 4 instances, and did not find a feasible solution.

6.3 Conclusion

In this chapter, we have studied the k-Node-Disjoint Hop-Constrained Network Design Problem (kNDHP) when $L \in \{2, 3, 4\}$. We have introduced an integer programming formulation for the problem when $L \in \{2, 3\}$ and investigated the associated polytope.

We have presented several classes of valid inequalities and presented conditions under which these inequalities define facets. Then, we have devised a Branch-and-Cut algorithm for solving the problem based on the inequalities we have presented before. In particular, we have discussed the separation problem of the *st*-cut, 3-*st*-path-cut, *st*-node-cut, 3-*st*-node-path-cut inequalities, as well as that of the Steiner SP-partition, Steiner partition and double cut inequalities. Finally, we have presented Branch-and-Cut and computational results for the problem when $L = 3$ and $k = 3, 4, 5$ on one hand, and when $L = 4$ and $k = 2$ on the other hand.

The experiments we have done in this chapter have shown that the Branch-and-Cut algorithm is quite efficient for solving the kNDHP when $L = 3$ and $k = 3, 4, 5$, and this, for both rooted and arbitrary sets of demands. They also pointed out that the large size instances are still difficult to solve within 5 hours of CPU time, but the gaps achieved, are in most cases quite interesting. Moreover, the experiments have shown the importance of Steiner SP-partition and double cut inequalities (5.17) and (5.11) are important in solving the problem, and that Steiner partition inequalities (5.19) seems to be less effective.

It should also be noticed that, contrarily to the survivable network design problem without hop constraints (or the kNCSP with $L \geq |V| - 1$), our experiments cannot permit to conclude on the impact of an increasing of the connectivity k on the resolution of the problem. In fact, previous experiments done for the survivable network design problem (see [9] and [31], for example) have concluded that the problem without considering hop constraints seems to become easier when k increases. In our case (the kNDHP with $L < |V| - 1$), the impact of the connectivity on the resolution is less clear. It even seems, when comparing the results for $L = 3$ and $L = 4$, that the kNDHP becomes more difficult to solve when L increases.

The computational study pointed out that a very large number of *st*-cut and 3-*st*-path-cut inequalities are generated during the resolution of the problem. This can be an issue since it yields the Branch-and-Cut algorithm to manage a huge pool of constraints and can imply an execessive CPU time consumption for constraints management. This can even yield the Branch-and-Cut algorithm to solve linear programs with a large number, but still polynomial, number of constraints. Finally, all this may prevent the algorithm from a good exploration of the Branch-and-Cut tree.

The above observations suggests that an efficient algorithm for the kNDHP requires a tighter formulation for the problem, which may efficiently include simultaneously both the disjoint paths and the hop contraints. Also, it may require a deeper investigation of the polytope of the problem in order to provide more facet-defining inequalities and yield an efficient Branch-and-Cut algorithm.

For theoretical purposes, it should be interesting to study the polyope of the kNDHP in some special cases, like for example when the graph is series-parallel. Also, one could investigate the problem with respect to the distribution of the demands, since it may influence the polyhedral description of the solutions of the problem, and probably the efficiency of resolution algorithms.

Another question which would be of interest is to see whether one can use directed models for the kNDHP. This may provide stronger integer linear programming formulations. This is one of our research lines in the future.

Conclusion

In this thesis, we have studied, within a polyhedral context, two survivable network design problems, the k-node-connected subgraph (kNCSP) and the k node-disjoint hop-constrained survivable network (kNDHP) problems. In particular, we have considered these problems in the case where a high level of connectivity is required, that is when $k \geq 3$. These two problems are NP-hard when $k \geq 2$.

In the first part, we considered the k-node-connected subgraph problem (kNCSP). We focused our attention on the polyhedron associated with this problem. We derived new classes of valid inequalities. We then described necessary and sufficient conditions for theses inequalities to define facets. Moreover, we have studied some reduction operations inspired from the ones introduced by Didi Biha and Mahjoub [13] (see also [11]). These allow to perform the separation of the valid inequalities in a reduced graph. Using these results, we have devised a Branch-and-Cut algorithm for the problem and given computational results for $k = 3, 4, 5$. The later was implemented to solve instances from SNDlib and TSPlib with realistic topologies. The experiments show in particular the efficiency of the separation procedure used in the Branch-and-Cut algorithm and the significant improvement enabled by the introduced inequalities on strengthening the linear relaxation of the problem.

We studied afterwards a hop-constrained version of the problem that is kNDHP, taking into account the length of the paths between every origin and destination of the demands. We introduced a new integer linear programming formulation for the problem. We studied the polyhedron associated with the formulation, in an attempt to describe strong valid inequalities for the problem. We investigated the properties of this polyhedron as well as the facial structure of the basic inequalities. This led us to define several classes of valid inequalities, that are facets of polyhedron under certain necessary and sufficient conditions, that we described. These valid inequalities were incorporated within a Branch-and-Cut framework. The obtained algorithm allowed to solve the problem for arbitrary and realistic instances from the TSPlib.

There are many directions in which the research in this thesis can be continued for both considered problems.

The experiments we have performed for the kNDHP for $k = 3, 4, 5$ and $L = 3$ have relatively high CPU time. It would be interesting to pursue the approach used here for the kNDHP when $L = 3$. One may lead a deeper investigation of the polytope of the problem by using the appropriate directed graphs and exploiting the known results on arc-disjoint paths problems in directed graphs. This may help to provide new facet defining inequalities. It would also be interesting, from an algorithmic point of view, to improve the separation procedures provided for the various inequalities we have introduced in this work. We need to develop more efficient separation heuristics for the Branch-and-Cut algorithm. It will be also interesting to focus on more sophisticated preprocessing methods in order facilitate the problem resolution. It will also be interesting to perform the experimentations for the kNDHP for $L = 4$.

The same kind of study can be used for the kNDHP when $L \geq 5$. If possible, this may provide an integer programming formulation for the problem as well as a Branch-and-Cut algorithm for all $k \geq 2$ and $L = 5$.

Bibliography

[1] http://sndlib.zib.de/home.action.

[2] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

[3] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[4] http://scip.zib.de/.

[5] Two-edge Connected spanning subgraphs A. R. Mahjoub and polyhedral. Two-edge connected spanning subgraphs and polyhedra. *Mathematical Programmig*, 64:199–208, 1992.

[6] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Implementing the dantzig-fulkerson-johnson algorithm for large traveling salesman problems, 2003.

[7] Francisco Barahona and Ali Ridha Mahjoub. On two-connected subgraph polytopes. *Discrete Mathematics*, 147(1):19 – 34, 1995.

[8] Cynthia Barnhart. Planning and control of transportation systems: stochastic optimization for robust planning in transportation. Technical report, 2000.

[9] F. Bendali, I. Diarrassouba, M. Didi Biha, A.R. Mahjoub, and J. Mailfert. A branch-and-cut algorithm for the k-edge-connected subgraph problem. *Networks*, 55:13–32, 2010.

[10] F. Bendali, I. Diarrassouba, A.R. Mahjoub, and J. Mailfert. The k edge-disjoint 3-hop-constrained paths polytope. *Discrete Optimization*, 7:222–233, 2010.

[11] M. Didi Biha. *Graphes k-arêtes connexes et polyèdres*. PhD thesis, Université de Bretagne Occidentale, Brest, 1998.

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

[13] M. Didi Biha and A.R. Mahjoub. The k-edge connected subgraph problem i: Polytopes and critical extreme points. *Linear Algebra and its Applications*, 381:117–139, 2004.

[14] Q. Botton, B. Fortz, L. Gouveia, and M. Poss. Benders decomposition for the hop-constrained survivable network design problem. *INFORMS J Comput*, 25:13–26, 2013.

[15] Q. Botton, Bernard Fortz, and Luis Gouveia. On the hop-constrained survivable network design problem with reliable edges. *Computers & Operations Research*, 64:159–167, 2015.

[16] Gilmore P. C. and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9:849–859, 1961.

[17] Gilmore P. C. and R. E. Gomory. A linear programming approach to the cutting-stock problem - part ii. *Operations Research*, 11:863–888, 1963.

[18] Markus Chimani, Maria Kandyba, Ivana Ljubić, and Petra Mutzel. Orientation-based models for {0,1,2}-survivable network design: theory and practice. *Mathematical Programming*, 124(1):413–439, Jul 2010.

[19] S. Chopra. The k-edge connected spanning subgraph polyhedron. *SIAM J Discrete Mathematics*, 7:245–259, 1994.

[20] G. Cornuéjols, J. Fonlupt, and D. Naddef. The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programmig*, 33:1âĂŞ27, 1985.

[21] G. Dahl. Notes on polyhedra associated with hop-constrained paths. *Operations Research Letters*, 25:97–100, 1999.

[22] G. Dahl, N. Foldnes, and L. Gouveia. A note on hopconstrained walk polytopes. *Operations Research Letters*, 32:345–349, 2004.

[23] G. Dahl and L. Gouveia. On the directed hop-constrained shortest path problem. *Operations Research Letters*, 32:15–22, 2004.

[24] George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[25] Jacques Desrosiers and Marco E. Lübbecke. *A Primer in Column Generation*, pages 1–32. Springer US, Boston, MA, 2005.

[26] I. Diarrassouba. *Survivable network design problems with high connectivity requirements*. PhD thesis, Université Blaise Pascal, 2009.

[27] I. Diarrassouba, V. Gabrel, A. R. Mahjoub, L. Gouveia, and P. Pesneau. Integer programming formulations for the k-edge-connected 3-hop-constrained network design problem. *Networks*, 67:148–169, 2016.

[28] I. Diarrassouba, H. Kutucu, and A. R. Mahjoub. Two node-disjoint hop-constrained survivable network design and polyhedra. *Networks*, 67:316–337, 2016.

[29] I. Diarrassouba, M. Mahjoub, and A. R. Mahjoub. The k-node-connected subgraph problem: Formulation, polyhedra and branch-and-cut. *Mathematical Programming*, 124(1):413–439, Jul 2010.

[30] I. Diarrassouba, M. Mahjoub, and A. R. Mahjoub. The k-node-connected subgraph problem: Formulation, polyhedra and branch-and-cut. In *Proceedings - CIE 45: 2015 International Conference on Computers and Industrial Engineering*, 2015.

[31] I. Diarrassouba, M. Mahjoub, A. R. Mahjoub, and R. Taktak. The k-node connected subgraph problem: Polyhedral analysis and branch-and-cut. *Electronic Notes in Discrete Mathematics*, 52:117–124, 2016.

[32] E. T. Dixon and S. E. Goodman. An algorithm for the longest cycle problem. *Networks*, 6:139–149, 1976.

[33] J. Edmonds. Covers and packings in a family of sets. *Bulletin of the American Mathematical Society*, 68(5):494–499, 1962.

[34] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards (B) 69*, 69:9–14, 1965.

[35] J. Edmonds and R.M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19:248–264, 1972.

[36] Manuel Fuentes, Luis Cadarso, and Ángel Marín. A new approach to crew scheduling in rapid transit networks. *Transportation Research Procedia*, 10:554 – 563, 2015. 18th Euro Working Group on Transportation, EWGT 2015, 14-16 July 2015, Delft, The Netherlands.

[37] M. R. Garey and D. J. Johnson. *Computer and intractability: A guide to the theory of NP-completness*. W. H. Freeman, 1979.

[38] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, San Francisco, 1979.

[39] Michel X. Goemans and Dimitris J. Bertsimas. Survivable networks, linear programming relaxations and the parsimonious property. *Math. Program.*, 60(2):145–166, June 1993.

[40] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Society for Industrial and Applied Mathematics*, 9:551–570, 1961.

[41] L. Gouveia, P. Patricio, and A. de Sousa. Compact models for hop-constrained node survivable network design, an application to mpls, telecommunications planning: Innovations in pricing, network design and management. *Springer*, 33:167–180, 2005.

[42] Luis Gouveia and Markus Leitner. Design of survivable networks with vulnerability constraints. *European Journal of Operational Research*, 258(1):89 – 103, 2017.

[43] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[44] M. Grötschel and C.L. Monma. Integer polyhedra arising from certain network design problems with connectivity constraints. *SIAM J Discrete Mathematics*, 3:502–523, 1990.

[45] M. Grötschel, C.L. Monma, and M. Stoer. Polyhedral approaches to network survivability. *Series in Discrete Mathematics & Theoretical Computer Science*, 5:121–141, 1991.

[46] M. Grötschel, C.L. Monma, and M. Stoer. Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research*, 40:309–330, 1992.

[47] M. Grötschel, C.L. Monma, and M. Stoer. Polyhedral and computational investigations for designing communication networks with high survivability requirements. *Operations Research*, 43:1012–1024, 1995.

[48] A.R. Mahjoub H. Kerivin and C. Nocq. *(1,2)-Survivable Networks: Facets and Branch&Cut*, pages 121–152. MPS/SIAM Optimization, 2004.

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

[50] D. Huygens and A.R. Mahjoub. Integer programming formulations for the two 4-hop-constrained paths problem. *Networks*, 43:135–144, 2007.

[51] D. Huygens, A.R. Mahjoub, and P. Pesneau. Two edge-disjoint hop-constrained paths and polyhedra. *SIAM J Discrete Mathematics*, 18:287–312, 2004.

[52] S. Irnich and G. Desaulniers. *Shortest Path Problems with Resource Constraints*. Springer, Boston, MA, 2005.

[53] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. 1972.

[54] H. Kerivin and A. Ridha Mahjoub. Design of survivable networks: A survey. *Networks*, 46:1–21, 2005.

[55] C-W. Ko and C. L. Monma. Heuristics for designing highly survivable communication networks. In *New Jersey*, 1989.

[56] A. R. Mahjoub. *Polyhedral Approaches*, pages 261–324. Wiley Online Library, 2013.

[57] A. R. Mahjoub and C. Nocq. On the linear relaxation of the 2-node connected subgraph polytope. *Discrete Applied Mathematics*, 95:389–416, 1999.

[58] A. R. Mahjoub, L. Simonetti, and E. Uchoa. Hop-level flow formulation for the hop constrained survivable network design problem. *Lecture Notes in Computer Science*, 61:176–181, 2011.

[59] M. Mahjoub, I. Diarrassouba, A.R. Mahjoub, and R. Taktak. The survivable k-node-connected network design problem: Valid inequalities and branch-and-cut. *Computers & Industrial Engineering*, 112:690 – 705, 2017.

[60] K. Menger. Zur allgemeinen kurventhorie. *Fundamanta Mathematicae*, 10:96–115, 1927.

[61] Inge Røpke. Theories of practice - new inspiration for ecological economic studies on consumption. *Ecological Economics*, 68(10):2490 – 2497, 2009.

[62] M. W. P. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 45:831–841, 1997.

[63] A. Schrijver. *Combinatorial Optimization : Polyhedra and Efficiency. Algorithms and Combinatorics*, volume 24. Springer, 2003.

[64] François Vanderbeck and Laurence A. Wolsey. An exact algorithm for IP column generation. *Oper. Res. Lett.*, 19(4):151–159, 1996.

[65] Inken Wierstra and Jürgen Alves. The c-myc promoter: Still mystery and challenge. *Advances in Cancer Research*, 99:113 – 333, 2008.

Résumé

Dans un graphe non orienté, le problème du sous-graphe k-sommet connexe consiste à déterminer un sous-graphe de poids minimum tel que entre chaque paires de sommets, il existe k chemins sommet-disjoints. Ce modèle a été étudié dans la littérature en termes d'arête connexité. Cependant, le cas de la sommet connexité n'a pas été traité jusqu'à présent. Nous décrivons de nouvelles inégalités valides et nous présentons un algorithme de Coupes et Branchements ainsi qu'une large étude expérimentale qui montrent l'efficacité des contraintes utilisées. Nous proposons ensuite une formulation étendue pour le même problème pour une connexité $k = 2$, suivi d'un algorithme de Génération de Colonnes et Branchements pour résoudre cette formulation.
Nous étudions ensuite la version avec chemins bornés du problème. Le problème consiste à trouver un sous-graphe de poids minimum, tel que entre chaque paire d'origine-destination, il existe k chemins sommet-disjoints de longueur au plus L. Nous proposons une formulation linéaire en nombres entiers pour $L = 2, 3$. Nous présentons de nouvelles inégalités valides et nous proposons des algorithmes de séparation pour ces contraintes. Nous présentons ensuite un algorithme de Coupes et Branchements qu'on a testé sur différentes instances.

Mots Clés

Conception de réseaux, Polytope, Facette, Séparation, Algorithme de Coupes et Branchements, Algorithme de Génération de Colonnes et Branchements.

Abstract

Given a weighted undirected graph and an integer k, the k-node-connected subgraph problem is to find a minimum weight subgraph which contains k-node-disjoint paths between every pair of nodes. We introduce new classes of valid inequalities and discuss their facial aspect. We also devise separation routines, investigate the structural properties of the linear relaxation and discuss some reduction operations that can be used in a preprocessing phase for the separation. Using these results, we devise a Branch-and-Cut algorithm and present some computational results. Then we present a new extended formulation for the the k-node-connected subgraph problem, along with a Branch-and-Cut-and-Price algorithm for solving the problem.
Next, we investigate the hop-constrained version of the problem. The k node-disjoint hop-constrained network design problem is to find a minimum weight subgraph such that between every origin and destination there exist at least k node-disjoint paths of length at most L. We propose an integer linear programming formulation for $L = 2, 3$ and investigate the associated polytope. We introduce valid inequalities and devise separation algorithms. Then, we propose a B&C algorithm for solving the problem along with some computational results.

Keywords

k-node-connected graph, k-node-disjoint hop-constrained paths, Survivable Network, Polytope, Facets, Separation, Branch-and-Cut, Branch-and-Cut-and-Price.