Hierarchical Network Design

Tommy Thomadsen

Kongens Lyngby 2005
IMM-PHD-2005-149

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-PHD: ISSN 0909-3192

Summary

Communication networks are immensely important today, since both companies and individuals use numerous services that rely on them. This thesis considers the design of hierarchical (communication) networks. Hierarchical networks consist of layers of networks and are well-suited for coping with changing and increasing demands. Two-layer networks consist of one backbone network, which interconnects cluster networks. The clusters consist of nodes and links, which connect the nodes. One node in each cluster is a hub node, and the backbone interconnects the hub nodes of each cluster and thus the clusters. The design of hierarchical networks involves clustering of nodes, hub selection, and network design, i.e. selection of links and routing of flows.

Hierarchical networks have been in use for decades, but integrated design of these networks has only been considered for very special types of networks. The thesis investigates models for hierarchical network design and methods used to design such networks. In addition, ring network design is considered, since ring networks commonly appear in the design of hierarchical networks.

The thesis introduces hierarchical networks, including a classification scheme of different types of hierarchical networks. This is supplemented by a review of ring network design problems and a presentation of a model allowing for modeling most hierarchical networks. We use methods based on linear programming to design the hierarchical networks. Thus, a brief introduction to the various linear programming based methods is included. The thesis is thus suitable as a foundation for study of design of hierarchical networks.

The major contribution of the thesis consists of seven papers which are included

in the appendix. The papers address hierarchical network design and/or ring network design. The papers have all been submitted for journals, and except for two papers, are awaiting review. The papers are mostly concerned with optimal methods and, in a few cases, heuristics for designing hierarchical and ring networks. All papers develop bounds which are used in the optimal methods and for comparison. Finally, computational results are reported.

Resumé

Kommunikationsnetværk har enorm betydning i dag, da enkeltpersoner og virksomheder anvender utallige tjenester, som afhænger af kommunikationsnetværkene. Denne afhandling omhandler design af hierarkiske (kommunikations-) netværk. Hierarkiske netværk er lagdelte og er velegnede til at håndtere ændringer og øgede i krav til båndbredde. Netværk med to niveauer består af et backbone netværk som forbinder klynger af netværksknuder. Klyngerne består af netværksknuder og forbindelser mellem netværksknuderne internt i klyngen. I hver klynge er en af netværksknuderne udpeget som hoved-netværksknuden, dvs. den har forbindelse til backbone netværket. Backbone netværket forbinder hoved-netværksknuderne og dermed klyngerne. For at designe et hierarkisk netværk, skal der tages stilling til hvilke netværksknuder der er i hvilken klynge, hoved-netværksknuderne skal udvælges, netværksknuderne skal forbindes både i klyngerne og i backbone netværket, og trafik skal rutes i netværket.

Hierarkiske netværk har været anvendt i årtier, men design af disse netværk er kun blevet undersøgt for specielle netværkstyper. Denne afhandling undersøger modeller til design af hierarkiske netværk og metoder der kan anvendes til at designe hierarkiske netværk. Derudover undersøges ring netværk, da ring netværk ofte forekommer i design af hierarkiske netværk.

Afhandlingen introducerer hierarkiske netværk, inklusive et klassifikationsskema af de forskellige typer af hierarkiske netværk. Dette bliver suppleret med en gennemgang af ring netværk problemer og en præsentation af en generel model, der tillader modellering af de fleste hierarkiske netværk. Metoder baseret på lineær programmering introduceres, idet de anvendes til design af hierarkiske netværk. Afhandlingen kan således danne grundlag for et studie af design af hierarkiske netværk.

Afhandlings vigtigste bidrag består af syv artikler, der er inkluderet i appendiks. Artiklerne handler om design af hierarkisk netværk og ring netværk. Artiklerne er alle indsendt til videnskablige journaler og afventer bedømmelse, bortset fra to artikler, der allerede er blevet accepteret. Artiklerne beskriver oftest optimale metoder og i enkelte tilfælde heuristikker til at design hierarkiske netværk samt ring netværk. I alle tilfælde er der udviklet grænser, der kan anvendes enten til sammenligning eller indlejret i de optimale metoder. Endelig præsenteres resultater for testkørsler af metoderne.

Preface

This thesis was prepared at Informatics and Mathematical Modelling, the Technical University of Denmark in partial fulfillment of the requirements for acquiring the Ph.D. degree.

The thesis considers optimization of communication networks with more layers denoted hierarchical networks. The Ph.D. project has been supervised by professor Jens Clausen.

The thesis consists of an introduction to the project and a collection of seven research papers prepared during the period 2002–2005.

Lyngby, May 2005

Tommy Thomadsen

Papers included in the thesis

A Vicky Mak, Tommy Thomadsen. Facets for the Cardinality Constrained Quadratic Knapsack Problem and the Quadratic Selective Travelling Salesman Problem, 2004. Submitted for *J. of Combinatorial Optimization*.

B Tommy Thomadsen, Thomas Stidsen. A Branch-and-Cut Algorithm for the Quadratic Selective Travelling Salesman Problem, 2003. Submitted for *Telecommunication Systems*.

C Tommy Thomadsen, Thomas Stidsen. Hierarchical Ring Network Design Using Branch-and-Price, 2005. *Telecommunication Systems*, Volume 29, Issue 1, pp. 61–76.

D Thomas Stidsen, Tommy Thomadsen. Joint Routing and Protection Using p-cycles, 2004. Submitted for *European Journal of Operational Research*.

E Tommy Thomadsen, Thomas Stidsen. The Generalized Fixed-Charge Network Design Problem, 2004. Accepted for publication in *Computers and Operations Research*.

F Tommy Thomadsen, Jesper Larsen. The Two-Layered Fully Interconnected Network Design Problem – Models and an Exact Approach, 2005. Submitted for *Computers and Operations Research*.

G Tommy Thomadsen. Design of Two-Layered Fixed Charge Networks, 2005. Submitted for *Networks*.

Acknowledgments

I would like to thank my wife Hanne for supporting me during the 3 years of work, and for giving me the opportunity to occasionally spent way too much time on working. Also I thank my daughter Laura for giving me the reason and opportunity to take on parental leave for 3 months, when it was needed the most.

I thank my colleagues at the operations research section for creating an invaluable work environment. I thank my supervisor Jens Clausen for being there when needed and especially I thank Thomas K. Stidsen for helpful discussions and feedback. The sometimes loud-voiced discussions have been, if not necessary, very beneficial. Finally I thank other colleagues for enduring all the noise we made and for closing the door when we forgot to.

Contents

CHAPTER 1

Introduction

Communication networks have increasing importance in the modern society, since communication networks provide services that are widely used and highly valued. Services that rely directly on communication networks are the telephone and the Internet, but numerous other services rely on communication networks, either since they use the Internet or since they establish separate communication channels, e.g. using a telephone line. Most obvious are services where access to centralized databases are required, e.g. money transfers and services in the public sector. Also, various jobs consist mostly of communicating, and as a consequence many employees cannot carry out their job if communication networks are not available.

Thus, services that rely on communication networks are numerous and the services are often paramount for companies to do business. For that reason, companies and individuals expect communication networks to be available in much the same way they require e.g. water and electricity to be available around-the-clock.

This thesis considers the design of communication networks. In particular, we consider hierarchical communication networks, since these have not received very much attention so far. Hierarchical communication networks have, nevertheless, existed for decades and it is widely recognized that hierarchical networks are beneficial, if not necessary, to cope with changing and increasing traffic demands.

Hierarchical networks divide a network into manageable entities, the clusters and the backbone. These entities can to some extend be considered independent. Alternatively the hierarchical networks can be seen as consisting of a lower and an upper layer. The lower layer consists of the clusters of nodes and links inside each cluster. One node in each cluster is designated as hub node. The upper layer consists of the hub nodes and links interconnecting the hub nodes, i.e. the backbone. In hierarchical networks there is a natural distinction between high and low capacity links, since for a uniform traffic demand, the backbone links are higher loaded than the cluster links.

Design of hierarchical networks have often been done by subdividing the design problem into at least two subproblems. The first subproblem consists of determining the hierarchy (i.e. the clusters and the hubs) and the second subproblem consists of interconnecting the nodes in the clusters and the backbone. One theme of the work presented in the thesis is to consider models where determination of the hierarchy and interconnection of nodes is done simultaneously. A first step is to model hierarchical networks and a second step is to devise algorithms for carrying out the optimization of the models.

It is challenging to devise models describing the problem properly, and especially devising models with a suitable trade off between the model detail and the computational difficulty is not easy. Many of the problems considered generalize problems which are difficult to solve. As a consequence, the problems considered present sufficient computational difficulties to be challenging to solve to optimality.

This chapter contains an introduction to hierarchical networks, and the subproblems of hierarchical network design. Furthermore an outline of the thesis is presented.

1.1 Hierarchical Networks

A network consists of nodes and links interconnecting the nodes. A hierarchical network consists of disjoint sets of nodes denoted clusters. In addition, each cluster contains at least one hub node. The backbone consists of the hub nodes interconnected by backbone links. Similarly nodes in each cluster are interconnected by cluster links. An example of a hierarchical network is shown in figure 1.1.

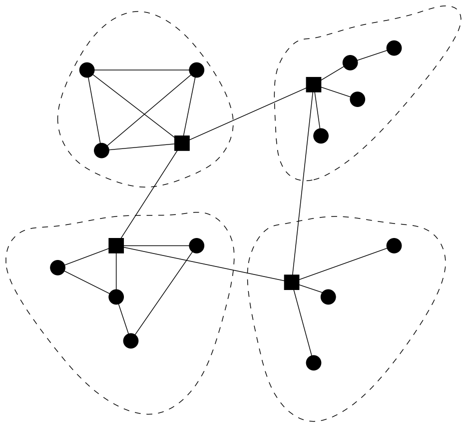


Figure 1.1: An example of a hierarchical network.

In the figure, each cluster contains one hub, which is indicated by a filled square. The backbone links are the links interconnecting the hubs and the cluster links are the links which interconnects nodes internally in a cluster. Note that there are no links between non-hub-nodes in different clusters. The backbone network is a ring, and the cluster networks are examples of four topologies: Fully interconnected, tree, star and mesh. The mesh topology allows for any selection of links and thus other topologies are special types of meshes.

Various topologies may be used in the clusters and in the backbone. Often the cluster networks are sparse, e.g. tree or star, whereas the backbone is usually denser, e.g. ring, mesh or even fully interconnected. This difference can be explained by the issue of protection. A broken link in the backbone potentially affects many more users than a broken link in the cluster networks. Thus, as a rule of thumb, the backbone should be better protected than the cluster networks, and thus the backbone is usually denser than the cluster networks.

1.2 Subproblems in Hierarchical Network Design

In designing hierarchical networks, interrelated subproblems have to be solved. The subproblems are listed below.

- Hub location (or selection)

- Clustering of nodes
- Interconnection of nodes in the backbone and cluster networks
- Routing

These subproblems have often been considered independently or only a few of them have been considered simultaneously. Since the subproblems are interrelated, this may lead to suboptimal designs. The problem considering all subproblems in an integrated fashion is denoted the *hierarchical network design problem*.

Papers dealing with hierarchical network design is reviewed in [26]. This paper also suggests a taxonomy for categorizing the networks. The taxonomy categorize problems depending on what topology the backbone and the cluster networks have. Thus, a network is identified by X-Y where X is the backbone topology and Y the cluster network topology. X and Y may be either ring, star, tree, fully interconnected or mesh. The taxonomy is used throughout this thesis.

The last of the four subproblems, routing, is trivial for most topologies, except ring and mesh. For the ring topology, the routing problem depends on the actual ring type. For the mesh topology, the last two subproblems are usually solved in an integrated fashion, by trading off routing costs and link establishment costs. This problem is denoted the fixed charge network design problem.

1.3 Outline of the Thesis

Following this introductory chapter, ring network design problems are surveyed in Chapter 2. Papers A, B, C and D address ring network design either solely or in the context of hierarchical Ring-Ring network design. Thus the chapter serves as an introduction to those papers and furthermore the papers and problems are put in context.

Chapter 3 introduces a model of the hierarchical network design problem. By adding constraints, topology requirements can be enforced on either the clusters, the backbone or both. The model is suitable for describing the hierarchical network design problem and to some extend for solving problems containing meshes, e.g. the mesh-mesh network design problem. However, for other problems, specialized models are most likely better. Introducing the model also serves as an introduction to notation and terminology used in papers on hierarchical network design, i.e. papers C, E, F and G.

Chapter 4 survey the optimization methods used. In all cases linear programming based methods have been used, thus these are the topic of the chapter.

In the context set up by the previous chapter, Chapter 5 give a brief introduction to each of the papers included in the thesis. The papers are included in the appendix.

Finally, Chapter 6 gives concluding remarks.

Chapter 2

Ring Network Design Problems

The ring network topology is widely used in communication networks. This is due to its inherent single link/node breakdown protection, its simple and fast restoration scheme, and its moderate cost. In many cases ring network design problems originate from other application areas and as a consequence have other names. The most fundamental and well known problem is the Travelling Salesman Problem (TSP). TSP and related problems have been widely studied and some of these problems are reviewed in this chapter. We consider symmetric problems, which has the property that the distance from node i to node j is the same as the distance from node j to node i.

In the papers included in the appendix, we have considered a generalization of the TSP, denoted the Quadratic Selective TSP (QSTSP). QSTSP is considered in Papers A, B, C and D either solely or as a subproblem. Thus, this chapter serves as an introduction to these papers by explaining TSP, related TSP generalizations, and QSTSP.

2.1 The Travelling Salesman Problem

TSP is the problem of determining a subset of links making up a cycle such that all nodes are visited at minimum cost. We denote any subset of edges making up a cycle, a ring. In the following we give a graph-theoretical description of TSP. As customary in such descriptions, let $V = \{1, \ldots, n\}$ be a set of nodes and $E \subseteq \{\{i,j\} : i \in V, j \in V\backslash\{i\}\}$ be a set of undirected edges representing the links of the network. Associate with each edge $\{i,j\} = e \in E$ a cost $c_e \geq 0$ incurred if e is chosen.

Furthermore, let $S \subseteq V$ and define $\delta(S) \subseteq E$ to be the set of edges with one endpoint in S and one endpoint not in S, i.e. $\delta(S) = \{\{i,j\} \in E : i \in S, j \notin S\}$. For notational convenience, $\delta(\{i\})$ is abbreviated by $\delta(i)$. Furthermore define $E(S) \subseteq E$ to be the set of edges with both endpoints in S, i.e. $E(S) = \{\{i,j\} \in E : i,j \in S\}$.

Let $x_e = 1$ if $e \in E$ is in the ring, 0 otherwise. Given this, TSP is formulated as follows.

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{e \in E} c_e x_e & & (1) \\
\text{s.t.} \quad & \sum_{e \in \delta(i)} x_e = 2 & \text{for } i \in V \quad & (2) \\
& \sum_{e \in E(S)} x_e \leq |S| - 1 & \text{for } \emptyset \subset S \subset V \quad & (3) \\
& x_e \in \{0,1\} & \text{for } e \in E \quad & (4)
\end{aligned}
$$

The objective (1) gives the cost or length of the ring, which has to be minimized. Constraint (2) ensures, that all nodes have an indegree of 2, and hence all nodes are in the ring. Constraint (3) ensures that the selected edge set is connected. The constraints are usually denoted the subtour elimination constraints. Finally constraint (4) ensures that an edge is either selected or not.

2.2 TSP with Optional Nodes

The generalizations we consider all have the property, that some or all nodes are optional, i.e. they do not necessarily have to be visited. In addition to the edge selection variables, a variable y_i is introduced for each node, which is 1 if

i is in the ring, 0 otherwise. In this case, the following constraint replaces constraints (2) and (3).

$$\sum_{e \in \delta(i)} x_e = 2y_i \qquad \text{for } i \in V \tag{5}$$

$$\sum_{e \in E(S)} x_e \leq \sum_{i \in S \setminus \{k\}} y_i - y_l + 1 \qquad \text{for } \emptyset \subset S \subset V, k \in S, l \notin S \tag{6}$$

$$y_i \in \{0, 1\} \qquad \text{for } i \in V \tag{7}$$

Constraint (5) ensures, that nodes have an indegree of 2 if node i is in the ring and 0 if it is not. Constraint (6) are the subtour elimination constraints which ensure that the selected edge set is connected. Finally constraint (7) ensures that a node is either in the ring or not. The following rewrite of (6) is easier to interpret.

$$\sum_{e \in E(S)} x_e \leq \sum_{i \in S} y_i + 1 - y_k - y_l \text{ for } \emptyset \subset S \subset V, k \in S, l \notin S \tag{8}$$

An interpretation is now, if a node k in S and a node l not in S are selected, and $c \leq |S|$ nodes in total are selected in S, then the number of edges selected in S (i.e. the left hand side) has to be less than or equal to $c - 1$. This ensures that the selected edge set is connected.

If this is the only modification of TSP and $c_e \geq 0$, the ring consisting of no nodes or edges is optimal with zero cost. Thus this modification usually comes with either a modification of the objective function or a constraint which ensures that at least a subset of the nodes have to be selected. These are discussed in the following sections.

2.2.1 Objective Modifications

Let c_i be a cost (possibly negative) incurred if $i \in V$ is not in the ring. Given this, the objective (1) may be modified to:

$$\text{Minimize } \sum_{i \in V} c_i y_i + \sum_{e \in E} c_e x_e \tag{9}$$

Thus a tradeoff is established between the edge costs incurred and a penalty incurred for not visiting node. Often such modifications are accompanied by a constraint on the length of the ring as described in the following.

2.2.2 Length and Price Constrains

In addition to the objective modifications, constraints on the nodes and/or edges visited may also exist. In the following, two types of constraints are shown. They are based on a revenue on the edge and/or nodes. They are shown in lower bound form, but since the revenues may be negative, they can also be seen as budget constraints. In this chapter, we distinguish between constants in the objective and constants in the constraints by consistently denoting the first costs and the latter revenues.

$$\sum_{e \in E} r_e x_e \geq R_x \tag{10}$$

$$\sum_{i \in V} r_i y_i \geq R_y \tag{11}$$

The constraints puts a lower bound on the revenues obtained from the edges and the nodes, respectively. The right hand sides, R_x and R_y are constants. An interesting special case of constraint (11) is the cardinality constraint in which $r_i = -1$ for all i and $-R_y$ is a number of nodes. This constraint puts a limit on the number of nodes in the ring. The costs in the objective and the revenues are often related (e.g. by $c_i = -r_i$)), but this is not always the case.

2.2.3 Depots

Some problems include one or more depots, i.e. mandatory nodes. In most cases it is important whether depot nodes exist, since the formulation may take advantage of the depot nodes. Thus problems with depot nodes may be computationally easer than problems without depots. Formulations which do not require depots, can usually incorporate depots by simply fixing y-variables corresponding to the depots to 1. Furthermore, if depots exist, the subtour elimination constraints (6) can be rewritten by assuming that S contains at least one depot node as in the following.

$$y_l + \sum_{e \in E(S)} x_e \leq \sum_{i \in S} y_i \text{ for } \emptyset \subset S \subset V, l \notin S, S \text{ contains depot} \tag{12}$$

2.2.4 Related Papers

A number of papers address generalizations of TSP with optional nodes. In [40], a generalized TSP is considered, with the only modification that a penalty is incurred if a node is not visited. The problem is transformed into standard TSP and solved.

The price collecting TSP is another generalization considered in [3, 4]. For the price collecting TSP, costs are incurred when an edge is in the ring and a penalty is incurred if a node is not visited. In addition, there is a lower bound on the node revenues that has to be obtained.

The selective TSP or the Orienteering problem is considered in [18, 20, 28]. The problem consists of minimizing the cost of nodes. In addition there is an upper bound on the length of the ring, measured in edge costs. It is assumed that a depot node exists.

The cardinality constrained circuit problem considered in [6] addresses the problem of minimizing the edge costs of the ring. There is an upper bound on the number of nodes that can be visited, i.e. a cardinality constraint. In addition there is a requirement, that the “ring” consisting of no edges is not a feasible solution, thus at least three edges has to be selected. No depot nodes exists.

Another generalization of TSP is discussed in [16, 17, 29]. In this problem, the nodes are assumed to be partitioned into disjoint clusters, and at least one node from each cluster has to be in the ring. The objective is as for the TSP to minimize the edge costs. Note that in the framework of hierarchical network design presented in Chapter 1, this problem can be considered as a hierarchical network design problem where the backbone is a ring. Furthermore, the clusters of nodes have been determined, and thus the hub location, interconnection of nodes and routing have to be carried out.

2.3 TSP with Quadratic Costs and Revenues

Some papers consider quadratic costs and/or revenues. Associate with all pairs of nodes a cost $c_{ij} \geq 0$ incurred if i and j are in the ring. Note that unlike for the cost of edges, c_e, costs are defined for *all* pairs of nodes.

Let z_{ij}, $i, j \in V, i < j$ be 1 if i and j are both in the ring, otherwise 0. For notational convenience z_{ji} is sometimes used instead of z_{ij}. Note that the definition of this variable is quite different than the definition of the variable for the edges, x_e. The variable z_{ij} are defined with respect to whether the *nodes* are in the ring, whereas x_e is defined with respect to whether the *edge* is in the ring.

The quadratic cost term may now be included in the objective:

$$\text{Minimize} \sum_{i \in V} c_i y_i + \sum_{e \in E} c_e x_e + \sum_{i,j \in V, i<j} c_{ij} z_{ij} \tag{13}$$

The y variables have to be linked to the z variables, which is accomplished by the following constraints.

$$z_{ij} \leq y_i \qquad \text{for } i, j \in V, i \neq j \tag{14}$$

$$z_{ij} \geq y_i + y_j - 1 \qquad \text{for } i, j \in V, i < j \tag{15}$$

$$z_{ij} \in \{0, 1\} \qquad \text{for } i, j \in V, i < j \tag{16}$$

In addition a quadratic revenue, r_{ij} may exists which is bounded similarly to bounds on edge and node revenues as given by the following constraint.

$$\sum_{i,j \in V, i<j} r_{ij} z_{ij} \geq R_z \tag{17}$$

2.3.1 Related Papers

The paper [19] considers a problem with quadratic costs and edge costs in the objective. In addition there is an upper bound on the length of the ring, measured in edge costs. Furthermore, no depot node is required. The paper presents heuristic for the problem.

A similar problem is considered in [22]. In this paper it is assumed, that at least one depot node exists. The authors suggest using alternative formulations for obtaining better bounds and solve the problem by a MIP solver.

Finally the two papers included in this thesis, Paper A and Paper B consider a problem with node, edge and quadratic costs. As mentioned, we denote this problem the Quadratic Selective TSP (QSTSP). Paper A considers a cardinality constraint on the nodes and in addition to the cardinality constraint, Paper B considers a bound on the edge costs.

Chapter 3

Hierarchical Network Design Problems

When describing hierarchical network design problems, it is beneficial to start by considering the fixed charge network design problem. The model for this problem is an integral part in the models we present for the hierarchical network design problems. Firstly, a model is presented for the fixed charge network design problem. This is then extended to describe hierarchical network design problems. In particular, various topology constraints are discussed, i.e. constraints that enforce a special structure on either the clusters, the backbone, or both.

3.1 The Fixed Charge Network Design Problem

The problem of designing cost efficient networks, taking into account both edge establishment, fixed costs and edge capacity costs, is denoted the Fixed Charge Network Design(FCND) problem.

Let V be the set of all nodes, the edges E be a set of unordered node-pairs and the arcs A be a set of ordered node-pairs. For each edge there exists exactly two arcs of opposite direction. We use s_a and t_a to denote the start and terminal node of arc a. In addition demands D for traffic between node-pairs exist. The set D is a set of unordered node-pairs, and we use s_d and t_d to denote the start and terminal node of demand d. The direction of the demands do not matter, since arcs always exist in both directions. For modeling purposes we only assume that some demands are given, and in some cases that the demands enforce a connected network, i.e. a disconnected network cannot serve the demands. For computational tests, test instances are investigated, in which a demand exist for each pair of nodes.

The binary decision variables y_e correspond to whether or not edge e between two hub-nodes should be established and the continuous decision variable x_a^d correspond to the fraction of demand d routed along arc a.

The communication demand volume for demand d is given by b_d. Furthermore, the capacity cost of arc a is given by c_a, which is the cost per unit of demand using that arc. Finally, the fixed cost of edge e is f_e. Given these definitions, the MIP model of the FCND problem is as follows.

$$\min \quad \sum_{e \in E} f_e y_e \quad + \sum_{d \in D, a \in A} c_a b_d x_a^d \tag{1}$$

$$\text{s.t.} \quad \sum_{a|s_a=i} x_a^d - \sum_{a|t_a=i} x_a^d = \begin{cases} 1 & \text{if } i = s_d \\ -1 & \text{if } i = t_d \\ 0 & \text{otherwise} \end{cases} \quad d \in D, i \in V \tag{2}$$

$$x_a^d + x_b^d \leq y_e \quad a, b \in A, s_a = t_b, t_a = s_b, \{s_a, t_a\} = e \in E, d \in D \tag{3}$$

$$y_e \in \{0, 1\} \tag{4}$$

$$x_a^d \in [0, 1] \tag{5}$$

The objective (1) contains the edge establishment costs and the edge capacity requirement costs. The flow constraints (2) ensure that the net out-flow is 1 if d starts in i, -1 if d terminates in i and zero otherwise. The constraints (3) ensure that flow is only allowed along established edges, constraints (4) ensure that edges are either established or not and finally constraints (5) ensure that the routed flows are between zero and one.

The FCND problem is NP-hard [23]. Its application to various problem areas is described in [31]. A number of optimization methods have been applied to the FCND problem: Benders decomposition e.g. [11, 30], dual ascent [2], cutting plane [1] and Lagrangian relaxation [23].

3.1.1 The Capacitated Fixed Charge Network Design Problem

The Capacitated Fixed Charge Network Design (CFCND) problem is obtained by adding capacities, c_e on the links:

$$\sum_{d \in D} b_d x_a^d + b_d x_b^d \leq c_e y_e \quad a, b \in A, s_a = t_b, t_a = s_b, \{s_a, t_a\} = e \in E \tag{6}$$

When capacities are present, it is important whether bifurcated (flows can split) or non-bifurcated flows are considered. If non-bifurcated flows are considered, it is in addition necessary to enforce that flow variables are integral:

$$x_a^d \in \{0, 1\} \tag{7}$$

A survey of CFCND is given in [21]. Lagrangian relaxation based methods are discussed in [23, 24] and bundle based methods for solving CFCND are discussed in [12]. Also Benders decomposition methods are surveyed in [11].

3.2 The Basic Model for Hierarchical Network Design Problems

Hierarchical network design problems enforce constraints on the network to be designed. As discussed in Chapter 1, when considering two layer networks, the nodes are divided into clusters. In addition to connections internally in clusters, a backbone network exists consisting of at least one node from each cluster, the hub nodes. These are the basic constraints necessary and ensure that the network designed is indeed hierarchical. In addition, topological constraints on the cluster networks and/or backbone networks may exists. We will return to such constraints later. Initially we also assume that each cluster contains exactly one hub node.

In addition to the constants, variables and constraints used in the FCND problem, we additionally define the following. Let c_{min} and c_{max} be a lower and an upper bound on the number of clusters, respectively and similarly let v_{min} and v_{max} be a lower and an upper bound on the number of nodes in each cluster, respectively. The variable h_i is 1 if node $i \in V$ is hub, 0 otherwise and z_{ij} is 1 if node $i \in V$ and $j \in V, i < j$ are in the same cluster, 0 otherwise. For notational convenience z_{ji} is sometimes used instead of z_{ij}. By adding the following constraints to the FCND problem, we obtain a hierarchical network.

$$
\begin{aligned}
y_e &\leq z_{ij} + h_k && \forall e = (i,j) \in E, k \in \{i,j\} && (8)\\
h_i + h_j + z_{ij} &\leq 2 && \forall i \in V, j \in V, i < j && (9)\\
z_{ik} + z_{jk} &\leq z_{ij} + 1 && \forall i,j,k \in V, i < j, k \neq i, k \neq j && (10)\\
v_{min} - 1 \leq \sum_j z_{ij} &\leq v_{max} - 1 && \forall i \in V && (11)\\
c_{min} \leq \sum_i h_i &\leq c_{max} && && (12)\\
z_{ij} \in \{0,1\} \quad & h_i \in \{0,1\} && && (13)
\end{aligned}
$$

Constraint (8) ensures that if a link between two nodes i and j are used, then the nodes are either in the same group, or both nodes are hub nodes. Constraint (9) ensures that if two nodes are in the same cluster, they cannot both be hubs. Constraint (10) ensures that if nodes i and k are in the same cluster, and nodes j and k are in the same cluster, then nodes i and j are in the same cluster as well. With constraints (13), constraints (8), (9) and, (10) ensure that the nodes are in disjoint clusters containing one hub node and links either connects nodes within clusters or hub nodes. Constraint (11) enforces bounds on the number of nodes in each cluster and similarly constraint (12) enforces bounds on the number of clusters.

3.3 An Extended Model for Hierarchical Network Design Problems

When expressing topological constraints on either the clusters or the backbone, it is beneficial to have variables distinguishing between whether links are in the backbone, y_e^1 or in clusters, y_e^2. These variables are linked to the existing variables by the expression $y_e = y_e^1 + y_e^2$. Thus they replace existing y_e variables.

It is often the case that links in clusters and in the backbone have different costs. Given fixed costs f_e^1 for the backbone links and by f_e^2 for the cluster links, this can now be expressed by modifying the objective. In addition the capacity cost may vary between the clusters links and the backbone links. To handle this, new variables x_a^d are defined, one for the backbone and one for the clusters, $x_a^{dl}, l \in \{1,2\}$. In addition two capacity costs are defined for each arc, $c_a^l, l \in \{1,2\}$. Thus the following objective replaces the previous objective (1).

$$\min \quad \sum_{e \in E, l \in \{1,2\}} f_e^l y_e^l \quad + \sum_{d \in D, a \in A} c_a^l b_d x_a^{dl} \tag{14}$$

So far, the cluster networks and backbone network are meshed fixed charge networks. By enforcing additional constraints, alternative topologies can be obtained in either the clusters, the backbone, or both. In most cases alternative models exist which are simpler and/or more suitable for computation. However, the fact that the various topologies can be enforced in the extend model, shows how general the extended model is.

The various topologies can either be present in the clusters or in the backbone, and in these cases require different constraints. The following section describes topological constraints on the cluster networks followed by a section describing topological constraints on the backbone.

3.4 Topology Constraints on the Clusters

When the cluster networks are trees, the number of cluster edges plus the number of hubs is exactly $|V|$. Furthermore assuming that the demands enforce a connected network, the following constraint ensure that cluster networks are trees.

$$\sum_{e \in E} y_e^2 + \sum_{i \in V} h_i = |V| \tag{15}$$

When the clusters are constrained to be stars, cluster edges can only be between nodes, where exactly one of the endpoint nodes is a hub. Constraint (9) ensures that at most one of the endpoint nodes of an edge is a hub and the following constraint ensures that at least one of the endpoint nodes is a hub.

$$y_e^2 \leq h_i + h_j \quad \forall e = (i,j) \in E \tag{16}$$

With constraint (15) these constraints ensure star networks in the clusters.

In the event that clustered are constrained to be rings, the indegree of all nodes must equal 2 ensured by the following constraints.

$$\sum_{e \in \delta(i)} y_e^2 = 2 \quad \forall i \in V \tag{17}$$

Note that subtour elimination constraints are not necessary in the single hub case. This is the case, since demands have to be handled and thus no node can be disconnected from hubs.

Finally fully interconnected cluster networks can be ensured by the following constraint.

$$y_e^2 = z_e \quad \forall e \in E \tag{18}$$

3.5 Topology Constraints on the Backbone

A backbone tree network can be ensured by requiring the number of backbone edges to be one less than the number of hubs.

$$\sum_{e \in E} y_e^1 = \sum_{i \in V} h_i - 1 \tag{19}$$

Ensuring that the backbone is a star seems non-trivial - the basic problem is that the "center" of the star is not know.

Enforcing a backbone ring, can be done by enforcing that nodes selected as hubs have indegree two, much the same way as described in Section 2.2 on TSP with optional nodes.

$$\sum_{e \in \delta(i)} y_e^1 = 2h_i \quad \forall i \tag{20}$$

Similarly to rings in the cluster networks, it is not necessary to have subtour elimination constraints. Finally a fully interconnected backbone network can be ensured by the following non-linear constraint.

$$y_e^1 = h_i h_j \quad \forall e = (i, j) \in E \tag{21}$$

The constraint can be linearized in a straightforward manner.

3.6 More Hubs

If more hubs are allowed or required, it may possibly be combined with an establishment cost for the hub. Let c_i be the cost of selecting node i as hub, then the establishment cost is included by adding the term $\sum_i c_i h_i$ to the objective.

The constraint (9) should be taken out of the formulation, since these constraints invalidates solutions with clusters containing more than one hub. If e.g. up to two hubs are allowed, constraint (9) can be generalized to cope with this, but the constraints cannot enforce that e.g. *at least* two hubs exist in each cluster.

The following non-linear constraint can cope with both cases, where H_l and H_u are lower and upper bounds on the number of hubs in each cluster.

$$H_l \leq h_i + \sum_{j|j \neq i} z_{ij} h_j \leq H_u \quad \forall i \in V \tag{22}$$

Since the constraints are non-linear, they are not suitable for use directly in a MIP solver. The constraints can, however, be linearized by introducing new variables $\overline{z}_{ij} = z_{ij} h_j$ and adding constraints linking $\overline{z}_{ij}$ with z_{ij} and h_j.

A better approach may be to replace the z_e variables with variables with cluster numbers as explained in the following. Define g_i^c to be 1 if node i is in cluster number c, $1 \leq c \leq c_{max}$. These variables replace the z_e variables, but in addition, the hub nodes need an additional index, such that they are defined as h_i^c is 1 if node i is hub in cluster c, 0 otherwise.

The important constraint in this context is that the case of more hubs can be handled by the following constraint.

$$H_l \leq \sum_i h_i^c \leq H_u \quad \forall c, 1 \leq c \leq c_{max} \tag{23}$$

Expressing the remaining constraints necessary to define the hierarchical network design problem is beyond the scope of this section.

3.7 Using the models

In many cases some variables are not needed, e.g. when a fully interconnected cluster is considered, $z_e = y_e^2$ and thus either of the variable can be left out to improve computational performance. However, the models serve the very important purpose of expressing, conceptualizing and formalizing the hierarchical network design problems. In most cases the above models cannot be used directly (i.e. by invoking a MIP solver) to solve larger instances in reasonable time. For various topology constraints on either the cluster networks, the backbone networks, or both, alternative models usually exist, possibly for slightly modified problems. Such specialized models are usually much more suitable for computations.

3.8 Related Papers

Given the various combinations of topologies on the clusters/backbone, numerous problem variants exists. Many of such problems are reviewed in [26]. Here we mention some recent problems of interest and the problems considered in this thesis.

A recent paper of interests in relation to hierarchical network design is the paper [33], which present a heuristic for solving a star-star problem. A recent paper [27] considers the ring-star problem, presents some strong valid inequalities and solve the problem to optimally using cutting plane methods. The paper [25] consider a problem where each cluster can consist of more rings. The problem is solved in steps. Initially clusters and hubs are determined using the methods described in [32] and the rings are designed in each cluster independently.

The ring-ring problem is considered by [34], [36], [37] which present a reformulation of the problem and present heuristics for solving it. The reformulation is considered in Paper C, and in this paper an optimal algorithm is presented.

The paper [15] reviews work on generalized network design problems. In such problems, it is assumed that clusters are known, hence hubs and backbone links have to be determined. The backbone network have various topologies. Paper E consider a similar problem, however, the backbone is a mesh network with fixed charges as for the FCND problem.

Finally the fully interconnected-fully interconnected network design problem is considered by Paper F and the mesh-mesh network design problem by Paper G.

CHAPTER 4

Linear Programming Based Methods

The papers included in this thesis all describe linear programming based methods and use a linear program solver as an integrated component. The solver used in experiments is CPLEX using either primal simplex or dual simplex, whichever is appropriate. In this chapter, we survey linear programming based methods.

4.1 The Linear Programming Relaxation

The linear programming relaxation (LPR) is used in all the methods considered, hence it is defined for a general MIP in the following. Assume that the constants a_{ij}, b_j, and c_i are given. Then, consider the following general MIP:

$$\min \quad \sum_i c_i x_i \tag{1}$$

$$\sum_i a_{ij} x_i \geq b_j \qquad \forall j \in J \tag{2}$$

$$x_i \in \{0, 1\} \qquad \forall i \in I \tag{3}$$

The MIP consists of an objective and constraints. Constraints are sometimes denoted cuts. The LPR is obtained by replacing the integrality constraints (3) with bounds on the variables. Thus the LPR is as follows:

$$\min \sum_i c_i x_i \tag{4}$$

$$\sum_i a_{ij} x_i \geq b_j \qquad \forall j \in J \tag{5}$$

$$x_i \in [0, 1] \qquad \forall i \in I \tag{6}$$

In addition some strengthening constraints may be added. The strengthening constraints are valid, thus all feasible solutions to the MIP are maintained. However, the strengthening constraints invalidates some solutions which are feasible to the LPR. As a consequence, the value of the LPR may increase, which is the purpose of adding the strengthening constraints. A facet is a valid constraint that cannot be strengthened any further. Unless explicitly stated, we do not distinguish between strengthening constraints and the constraints that define the problem.

4.2 Branch-and-Bound

In order to solve the MIP, branch-and-bound is applied, with the value of the LPR used as the bound. Branching is usually done on the integer variables. Often, an off the shelf MIP-solver such as CPLEX can be used with success. The MIP-solver manages the branch-tree and selects branching variables. Furthermore, it often adds strengthening constraints to improve the performance. We have mostly studied problems, where it is not possible or beneficial to apply an off the shelf MIP-solver directly. In this case the problems are complex and have theoretical and algorithmic interest.

4.3 Cutting Plane and Branch-and-Cut

Consider the LPR, possibly with some additional strengthening constraints included. Generally, it is beneficial to have as few constraints as possible, since the computation time of the linear programming solver increases with the number of constraints. Observe that for the optimal solution, $x_i^*, i \in I$, some constraints may not be binding, i.e. for some j, $\sum_i A_{ij} x_i^* > b_j$. A constraint that is not binding for the optimal solution, is actually not needed in the formulation. Thus a speedup is in principle possible, if the non-binding constraints can be determined prior to solving the problem. However, to determine whether a constraint is binding for the optimal solution, the optimal solution is needed.

This leads to the idea of constraint generation or cutting plane methods. Cutting plane methods initially set up a problem with very few constraints and solve this problem. Then alternately, violated constraints are added and the problem is resolved. This is continued until no violated constraint exists and thus the optimal solution is obtained. In a sense, all constraints are considered implicitly. This approach often leads to significant speedups and allows for solving problems with a huge, often exponential number of constraints in the problem size. This is especially the case, if the number of binding constraints for the optimal solution is low.

Branch-and-cut is a branch-and-bound algorithm where the cutting plane method is used to strengthen the LPR bound in each node of the branch-tree. Today, branch-and-cut is much more often used than cutting plane. Furthermore, the challenges experienced when developing branch-and-cut methods include all the challenges experienced when developing cutting plane methods. Thus, only branch-and-cut is discussed in the following.

Since much computation time is spent on resolving linear programming problems, it is very important to do this efficiently. It turns out to be crucial to use the information given in the previously obtained solution and start solving from this solution. This is denoted warm-start. This holds for both cutting plane and branch-and-cut, with the additional caveat that when branching or backtracking in the branch-tree, the stored solution may not give as high a speed up as intended. To obtain best possible match between the stored solution and the problem to solve, depth-first-search is always preferred over best-bound-first in branch-and-cut.

It is non-trivial to figure out which constraints to add during each iteration of the branch-and-cut method. In case a polynomial number of constraints of some type exists, it is often a good idea to explicitly evaluate all constraints and add the ones that are violated. In case an exponential number of constraints exist, an alternative is to solve the problem of determining whether any violated constraint exists. If so, determine at least one constraint to add. This is denoted the separation problem.

In some cases, the separation problem for a particular type of constraint can be solved to completion, i.e. a violated constraint is identified if any violated constraint exists. However, if the constraints are used to strengthen the formulation, it is not necessary to solve the separation problem to completion. If the separation problem is instead solved heuristically, the bound may, however, not be as strong as it could have been.

For the constraints that define the problem, the separation problem should be solved to completion, otherwise the following difficulty may be experienced. An integer, nevertheless, infeasible solution appears, but no violated constraint is generated since the separation routine is heuristic. Since the solution is integer, branching is not possible. Thus, the heuristic separation routine should for integer solutions produce at least one violated constraint if any violated constraint exists. This is most often the case for the heuristic separation routines, regardless that this has not been an issue when developing the routine.

Generally speaking, there is a tradeoff between solving the separation problem, resolving linear programs and the additional time required to do branching if the separation problem is solved heuristically. A typical approach to take is to solve the separation problem heuristically and, if necessary, solve it to completion, when the heuristic does not determine any violated constraints. Often it is not beneficial to include all violated constraints that can be determined, since too many constraints increase the computation time. A typical approach is to put a limit on the number of constraints that are added during each iteration. If more constraints are determined, some are discarded either randomly or by ranking the constraints in order of "how much they are violated", i.e. by $b_j - \sum_i A_{ij}x_i^*$. However, it is also of importance how the constraints interact. If two constraints address the same aspects, e.g. they almost include the same variables, it may not be beneficial to add both constraints. Thus, it is most often advantageous to use a heuristic separation routine that generates constraints that are in some way different rather than ranking generated constraints based on the violation. An example of such a heuristic separation routine is given in [17].

4.4 Column Generation and Branch-and-Price

Similarly to including only some of the constraints in cutting plane methods, it is possible to include only some of the variables. This is denoted column generation, and if used in a branch-and-bound setting, branch-and-price or integer column generation [5, 39]. The motivation to include only some of the variables is, similarly to cutting plane methods, that the computation time of the linear programming solver increases with the number of variables. Furthermore, a substantial number of the variables may be zero in the optimal solution, and they are thus not needed when solving the problem. In column generation and branch-and-price methods, initially, a problem is set up with some of the variables, only. Then alternately, variables with negative reduced costs are added and the problem is resolved until no variables with negative reduced costs exist.

Column generation is most often used when an exponential number of variables are present. In this case, it is necessary to solve a column generation problem - too many variables exist to just check the reduced cost of all variables. Similarly to cutting plane methods, it is important to use warm-start to get an efficient algorithm.

Unlike the separation problem in cutting plane methods, it is necessary to solve the column generation problem to completion to get a bound. Thus, it is of little value to have a column generation problem which cannot be solved to completion in practice. Conversely, it is certainly possible and common for cutting plane methods to use constraints with a separation problem that is only solved heuristically.

In this context, it is worth mentioning the concept of stabilized column generation [13]. The observation is that in many cases the convergence of the column generation algorithm is slow, i.e. many of the columns generated have the value zero in the optimal solution obtained in the end. By examining the dual variables at each iteration of a run of a column generation algorithm, it can be seen that the dual variables often have extreme values and change radically from one iteration to the next. This is especially seen during the first iterations of a column generation algorithm [38]. Since the columns are generated based on the values of the dual variables, the generated columns will be very different in structure from iteration to iteration and the columns generated may be far from the columns that turns out to be in the optimal solution. To circumvent this, stabilized column generation seeks to avoid large changes in the dual variables from iteration to iteration, thus hopefully generating more suitable columns. Computational experiments support that this is a good idea, at least for set packing and set covering problems [38].

Using branch-and-price pose some additional challenges regarding the branching, compared with branch-and-cut. In particular it is not a good idea to use variable branching if an exponential number of columns exist. If variable branching is used, the column that is fixed to 0 may, and often will, be generated again. Thus the column generation problem has to take branches into account.

In special cases, Ryan-Foster branching [35] can be used and generally constraint branching as described in [5, 39] is applicable. The constraints added during constraint branching also give rise to dual variables, which have to be included in the column generation problem. In some cases this may be difficult. A short intuitive description of such branching schemes is included in Paper C.

4.5 Branch-Cut-and-Price

When coding linear programming based methods adding variables and constraints are similar operations; they simply correspond to adding rows or columns to a matrix. Thus it makes sense to integrate the methods in a branch-cut-and-price method allowing for both adding variables and constraints. Such functionality is included in the "branch-cut-and-price"-framework from coin[10]. This framework has been used in most papers included in the thesis.

Chapter 5

Papers in the Thesis

In this chapter, the seven papers included in the appendix is presented one by one. For each paper, the problems, the models, the methods, and the results are briefly discussed and related to the other papers.

5.1 Paper A: Facets for the Cardinality Constrained Quadratic Knapsack Problem and the Quadratic Selective Travelling Salesman Problem

Two related problems are considered in this paper. The Cardinality Constrained Quadratic Knapsack Problem (CCQKP) and the Quadratic Selective Travelling Salesman Problem (QSTSP). The paper presents strengthening constraints for the two problems and gives proofs that the constraints are indeed facets.

The QSTSP is a ring network design problem with quadratic costs as described in Section 2.3. QSTSP generalizes TSP, and is much more difficult than TSP due to the quadratic terms in the objective function. Furthermore, QSTSP generalizes the Quadratic Knapsack Problem (QKP).

The QKP is an extension of the classical knapsack problem. The classical knapsack problem seeks to maximize the revenue obtained for selected elements, subject to a budget constraint on the weight of the elements. The QKP extends this formulation by allowing for quadratic revenues, i.e. revenues obtained if pairs of elements are selected. The QKP is considered in e.g. [7, 8, 9]. The CC-QKP is the special case of the QKP where the budget constraint is a cardinality constraint, i.e. the elements have unit weights.

Strong valid inequalities and facets in particular are useful in a branch-and-cut method, since they can be used to improve the bound of the linear programming relaxation. Thus the contribution of the paper is the constraints presented and the proofs that they are indeed facets.

5.2 Paper B: A Branch-and-Cut Algorithm for the Quadratic Selective Travelling Salesman Problem

This paper presents heuristics and a branch-and-cut algorithm for the QSTSP. Two different budgets are considered. The first is a budget on the number of nodes and the second is a budget on the length of the ring. Computational results are reported based on generated test instances. The test instances are fully interconnected and a demand exists for all pairs of nodes. The computational experiments show that instances can be solved to optimality but also that the running time can be substantial, up to almost 30 minutes for 50 node instances. However, the running time depends heavily on the test instance. In particular the running time depends heavily on which of the two budget types are considered and how restrictive it is.

5.3 Paper C: Hierarchical Ring Network Design Using Branch-and-Price

The design of a hierarchical ring network have received some attention previously and is the topic of this paper. In hierarchical ring networks, both the backbone and the cluster networks consist of rings, i.e. a ring-ring network design problem. The algorithm presented determines the clusters, hubs and the rings in the clusters. The routing cost of the backbone ring is only estimated. The algorithm is based on a set partitioning formulation with a column for each possible ring. Branch-and-price is used, and it turns out that the column generation subproblem is the QSTSP. Optimal solutions are obtained for general fully connected networks with up to 20 nodes in just about one hour and in special cases for networks with up to 36 nodes in one and a half hour.

5.4 Paper D: Joint Routing and Protection Using p-cycles

The p-cycle protection method is recognized as a promising way to protect networks. This paper address a problem which consists of routing demands in a network and simultaneously protect the routed demands with p-cycles. The p-cycles protect the routed demands by protecting each link of the demand route. A p-cycle consists of a set of links making up a cycle. In addition to being able to protect links on the cycle, the p-cycle can also protect chords of the cycle. As a consequence, p-cycles have the advantage of being much more capacity efficient than ring protection and at the same time the simple and fast restoration scheme of ring protection can be used. In fact the capacity efficiency is remarkably close to that of meshed protection schemes.

The algorithm used to route and protect demands is very similar to the algorithm used in Paper C. It is a column generation method with two types of generated columns. One column type corresponds to the routes and the column generation subproblem is the shortest path problem. The other column type corresponds to the p-cycles and in this case, the column generation subproblem is the QSTSP.

The computational results use real-world networks with a generated demand. Results are obtained for networks with up to 43 nodes, 71 links, and demands between all pairs of nodes. Solutions are in most cases within 1% of the optimal solution obtained with a worst case computation time of 6 minutes. This emphasizes the computational efficiency of the proposed method when compared to existing heuristic methods.

5.5 Paper E: The Generalized Fixed-Charge Network Design Problem

Consider a network design problem in which some nodes have to be connected in a specified topology (e.g. a tree or a ring) by edges. A generalization of such a network design problem is the following. Assume that nodes are allocated to node disjoint clusters and only one node (the hub) has to be selected from each cluster. Determine the hub nodes and interconnect the hub nodes by edges in the specified topology. Such generalized network design problems are reviewed in [15].

This paper considers the generalized fixed charge network design problem in which the topology mentioned above is a fixed charge network design. The paper presents a model and an algorithm for solving the problem. The generalized fixed charge network design problem is a hierarchical network design problem, since it consists of determining hubs, the interconnection of the hubs, and routing in the backbone.

A branch-cut-and-price algorithm for solving the problem is presented. A polynomial number of variables and constraints exists, thus pricing and separation problems are carried out by evaluating the reduced costs of variables and evaluating constraints to see if they are violated. The algorithm is much more efficient than straightforward solution using a MIP solver, since very few variables are non-zero and very few cuts are binding.

Optimal results are obtained for networks with up to 300 nodes and 10 clusters, 150 nodes and 20 cluster, and 50 nodes and 30 clusters within 10 hours of computation time. The results are quite remarkable, since the number of variables and constraints are $O(n^4)$, where n is the number of nodes.

5.6 Paper F: The Two-Layered Fully Interconnected Network Design Problem – Models and an Exact Approach

This paper presents the problem of designing a hierarchical network where both the backbone and the clusters consist of fully interconnected networks. Two integer programming models are presented. One model gives poor bounds, whereas a column generation model gives much better bounds. The column generation model is similar to models for the set-partitioning problem, with the exception that two types of interacting columns exists. One type of column corresponds to the backbone network and the other type of column corresponds to the cluster networks. The column generation problem can be solved by solving a series of QKPs. Solving a series of QKPs has the advantage that more columns are generated during each iteration.

The column generation model is embedded into a branch-and-bound procedure and is thus solved by branch-and-price. The computational results show that this is indeed a viable way to solve the problem. A substantial amount of time is, nevertheless, spent on solving the column generation problem to optimality. This is the case, regardless that optimal methods are only used if heuristics do not produce any solutions to the column generation problem. The computational results are based on networks with up to 25 nodes and show that the running time depends heavily on the way the test instances are generated.

5.7 Paper G: Design of Two-Layered Fixed Charge Networks

In this paper, the hierarchical network design problem is studied. The problem consists of designing a two-layered network where the backbone and the clusters consist of fixed charge networks. The paper presents two integer programming models that are extensions of the models presented in Paper F. The additional complication is to cope with flow of demands. The linear programming bounds of the two models are investigated, and it is shown that the column generation model achieves the best bounds. However, neither of the bounds are tight enough to be used for an optimal method. Instead, a GRASP heuristic [14] is developed, which supplies heuristic solutions. Heuristic solutions are obtained for complete networks with up to 50 nodes and a full demand matrix, albeit at the cost of significant running times.

CHAPTER 6

Conclusion

This thesis has presented problems and algorithms in the area of hierarchical network design. Such problems combine hub location, clustering and network design. Furthermore, ring network design problems have been surveyed.

The thesis contains an introduction and a description of models and related work in the area of ring network design and hierarchical network design. Furthermore linear programming (LP) based methods are introduced. The majority of the contribution consists of the seven papers included as appendices. The papers are briefly described in Chapter 5.

6.1 Summary

The ring network design problems discussed in Chapter 2 are of great importance since ring networks are widely used. In addition, ring network design problems appear as subproblems in many real-life applications. We have considered one particular problem, the quadratic selective TSP in Paper A and Paper B, and use the results obtained here in Paper C and Paper D.

When considering hierarchical network design, one common approach is to divide the problems into subproblems which are solved separately, leading to suboptimal designs. The approach taken in this thesis is instead to consider integrated problems and devise methods for these. As a consequence, we usually face computationally challenging problems. In some cases, the problems presented have not been considered previously, and in other cases we present reformulated models of existing problems that are more suitable for computation.

The hierarchical network design model presented in Chapter 3 is quite general and includes the problems considered in papers C, E, F, and G. The model is not very suitable for direct computation, but serves the purpose of formalizing the problems. The LP relaxation of the model is used as bound in Paper G.

The algorithms presented are mostly LP based methods and are examples of well performing methods. The methods are surveyed in Chapter 4, which is also a short introduction to LP based methods, including references for related material.

In LP based methods, studies of the underlying polyhedra are of great importance. We have used such results whenever possible and present polyhedral results for two related problems in Paper A.

The computational results are ample. Papers B, C, E, F, and G present results for optimal methods. The running times are in most cases quite high, but given the complexity of the problems, such running times are to be expected and acceptable. Papers B, D, and G present results for heuristics, but in all cases bounds are presented as well. The benefit of the bounds is to allow for an evaluation of the quality of the heuristic solutions. In Paper D, the bound is excellent, in Paper G the bound is not very good, and in Paper B the bound is somewhere in between. The heuristic solutions on the other hand seems to be reasonable in all three cases.

6.2 Main Contributions

The problems that are addressed have applications in communication network design and are of both practical and theoretical interest. The most important contribution of the work reported is the models developed for hierarchical network design problems and the linear programming based methods devised for solving the problems. Through computational experiments, it is shown that the methods work in practice. During the development of the methods, a number of challenging subproblems have been identified and solved, most notably, the novel quadratic selective TSP. The quadratic selective TSP is in itself an interesting problem, since it resembles ring network design.

Finally, a spin-off result is the development of the algorithm to jointly route and protect networks using p-cycles in Paper D. The method has proven superior to previously suggested methods.

6.3 Future Work

Several extensions to problems and/or related problems may be considered. Obvious extensions are capacitated networks, more hubs per cluster, inclusion of equipment costs, and more than two layers. All extensions are motivated by real-world networks, and are likely to increase the problem complexity even further.

Alternative models are of interests, and in particular constraints which strengthens existing formulations may prove an important tool to reduce computation times.

Numerous algorithmic improvements are possible and described in detail in the papers. Here we mention some of the general algorithmic improvements that are of interests. One such example is the stronger constraints mentioned above. Another example is the stabilization of the set-partitioning-like models considered in papers C, D, F, and G.

Since the focus has been on optimal solutions, one obvious line of research to explore is construction heuristics. Heuristics are interesting, either for obtaining solutions fast or as input to speed up the branch-and-cut and branch-and-price algorithms. Furthermore, a heuristic algorithm decomposing the hierarchical network design problem into subproblems based on the ideas in Paper E is of interests. For the purpose of benchmarking, the results presented in this thesis will be valuable in studies of heuristics.

Finally computational experiments are largely based on generated data. Thus, computational experiments based on real-world data would be of great significance. An alternative would be to consider sparser networks, resembling real world networks.

Appendix A

Facets for the Cardinality Constrained Quadratic Knapsack Problem and the Quadratic Selective Travelling Salesman Problem

Submitted for *J. of Combinatorial Optimization*

Facets for the Cardinality Constrained Quadratic Knapsack Problem and the Quadratic Selective Travelling Salesman Problem

Vicky Mak[1] and Tommy Thomadsen[2]

Abstract

This paper considers the Cardinality Constrained Quadratic Knapsack Problem (QKP) and the Quadratic Selective Travelling Salesman Problem (QSTSP). The QKP is a generalization of the Knapsack Problem and the QSTSP is a generalization of the Travelling Salesman Problem. Thus, both problems are NP hard.
The QSTSP and the QKP can be solved using branch-and-cut methods, and in doing so, good bounds can be obtained if strong constraints are used. Hence it is important to identify strong or even facet-defining constraints. This paper presents the polyhedral combinatorics of QSTSP and QKP, i.e. amongst others identify facet-defining constraints for the QSTSP and the QKP, and provide mathematical proofs that they do indeed define facets.

Keywords: Quadratic Knapsack, Quadratic Selective Travelling Salesman, Polyhedral Analysis, Facets

A.1 Introduction

A well-known extension of the Travelling Salesman Problem (TSP) is the Selective (or Prize-collecting) TSP: In addition to the edge-costs, each node has an associated reward (denoted the node-reward) and instead of visiting all nodes, only profitable nodes are visited. The Quadratic Selective TSP (QSTSP) has the additional complications: (1) each *pair* of nodes have an associated reward (denoted the edge-reward) which can be gained only if both nodes are visited; and (2) a constraint on the number of nodes selected is imposed, which we refer to as the cardinality constraint. The objective of an QSTSP is to maximize

[1]This paper was written mostly when Vicky Mak was working at Department of Mathematics and Statistics, University of Melbourne, Australia. She now works at School of Information Technology, Dearkin University. Email:vmak@ms.unimelb.edu.au

[2]Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark. Email: tt@imm.dtu.dk

the total node-reward and edge-rewards gained minus the edge-costs incurred subject to the satisfaction of the cardinality constraint.

Conceptually the QSTSP consists of two interacting problems, a cardinality-constrained min-cost circuit problem with respect to the edge-costs and a cardinality-constrained max-reward clique problem with respect to the edge-rewards.

The cardinality constrained circuit problem (CCCP) is considered in [3] where polyhedral results are presented and in [4] where a branch and cut algorithm is discussed. The max-reward clique problem is a special case of the quadratic knapsack problem where the knapsack constraints have unit coefficients. We denote this problem the cardinality constrained quadratic knapsack problem (QKP). The quadratic knapsack problem (when coefficients are not necessarily unit) is considered in e.g. [15], [5] and [6]. If edge-rewards are non-negative, the cardinality constraint will be met with equality. This is similar to the p-dispersion problem considered in [7] wherein the objective is to maximize the minimum edge-reward. The p-dispersion problem is considered in [18] with an objective equivalent to the one considered here.

Various TSP-like problems are similar to QSTSP in the way that a subset of nodes has to be selected. E.g. the Prize-collecting TSP [1, 2], Selective TSP [12, 16], the Orienteering problem [10], and the Generalized TSP [8, 9]. Problems that consider the combination of a clique problem and a cycle problem has been studied in [11] and [13]. Gendreau, Labbe, and Laporte [11] study a problem where instead of imposing the cardinality constraint, an upper bound on the sum of the edge-costs are imposed. Gouveia and Manuel Pires [13] study a QSTSP-like problem with the additional requirement that some nodes must be in the ring.

In this paper we study the polyhedral combiatorics of the QKP and the QSTSP. Our interest in studying the QSTSP is due to the fact that this problem arose as a subproblem from another combinatorial optimisation problem which deals with designing hierarchical ring networks (see [19]). Naturally, the faster we can solve the QSTSP, the better. The QKP, however, is an interesting problems in its own right, but we study the QKP mostly for its relevance in understanding the QSTSP. Since QKP is a generalization of the Knapsack Problem and QSTSP is a generalization of the Travelling Salesman Problem, both problems are NP hard.

A promising approach in solving these combinatorial optimisation problems is the branch-and-cut method. A significant factor in the success of the method is the use of strong constraints that at least partially describe the convex hull of the incidence vectors of all feasible solutions, in other words, the use of facet-defining cuts.

The contribution of this research is therefore the identification of some of the facet-defining cuts, the mathematical proofs that these cuts are indeed facet-defining, and the various mathematical techniques used in proving these results.

We begin with, in Section A.2, giving an integer programming model for QSTSP and define the polyhedra of the QKP, CCCP, and the QSTSP. In Sections A.3 and A.4, we present our polyhedral results on the QKP and the QSTSP polytopes. Finally, in Section A.5, we conclude our findings.

A.2 Integer Programming Model and the Polyhedra

We consider QSTSP defined on the undirected graph $G = (V, E)$. We assume that G is a complete undirected graph. This is not restrictive, since we can always introduce high costs for edges that do not exist.

For notational convenience, we use (U_1, U_2) to denote $\{(i,j) \in E \mid i \in U_1, j \in U_2\}$, for any $U_1, U_2 \subseteq V$; use $\delta(S)$ to denote $\{(i,j) \in E \mid i \in S, j \in V \setminus S\}$; and use $E(S)$ to denote $\{(i,j) \in E \mid i,j \in S\}$.

To give the model, we use r_i for the node-reward, w_e for the edge-reward, c_e for the edge-cost, and b for the maximum number on nodes allowed in the ring. Let x_e be the decision variable with $x_e = 1$ if $e \in E$ is chosen in the ring and 0 otherwise; y_i be the decision variable with $y_i = 1$ if $i \in V$ is on the ring, 0 otherwise; and z_e be the decision variable with $z_e = 1$, for $(i,j) \in E$ if node i and j are both on the ring, 0 otherwise. If $(i,j) = e \in E$, then z_{ij} is sometimes used in place of z_e. Given these, the QSTSP is formulated as follows.

$$\max \quad \sum_{i \in V} r_i \cdot y_i + \sum_{e \in E} w_e \cdot z_e - \sum_{e \in E} c_e \cdot x_e \tag{1}$$

$$\text{s.t.} \quad \sum_{e \in \delta(i)} x_e = 2y_i, \ \forall i \in V \tag{2}$$

$$z_e \le y_i, \ \forall i \in V, e \in \delta(i) \tag{3}$$

$$z_{ij} \ge y_i + y_j - 1, \ \forall (i,j) \in E, i < j \tag{4}$$

$$\sum_{e\in\delta(S)} x_e \geq 2(y_k + y_l - 1),$$
$$\forall \emptyset \subset S \subset V, k \in S, l \notin S \quad (5)$$
$$\sum_{i\in V} y_i \leq b \quad (6)$$
$$x_e \in \{0,1\}, \forall e \in E \quad (7)$$
$$y_i \in \{0,1\}, \forall i \in V \quad (8)$$
$$z_e \in \{0,1\}, \forall e \in E \quad (9)$$

Constraints (2) makes sure that if and only if a node is selected, the indegree of the node is 2. Constraints (3) and (4) establishes that $z_{ij} = 1$ if and only if $y_i = y_j = 1$. Constraints (5) is the subtour elimination constraints and (6) is the cardinality constraint. Finally (7), (8) and (9) are the binary constraints. Let $n = |V|$. The Quadratic Selective Travelling Salesman(QSTS) polytope is defined to be

$$P_{QS}^{nb} = \text{ conv } \{(x,y,z) \in \mathcal{R}^{2|E|+n} | (x,y,z) \text{ satisfies } (2)-(9)\}. \quad (10)$$

We identify two related polytopes. The cardinality constrained quadratic knapsack(QK) polytope,

$$P_{QK}^{nb} = \text{ conv } \{(y,z) \in \mathcal{R}^{|E|+n} | (y,z) \text{ satisfies } (3),(4),(6),(8) \text{ and } (9)\}, \quad (11)$$

and the cardinality constrained circuit polytope,

$$P_{C}^{nb} = \text{ conv } \{(x,y) \in \mathcal{R}^{|E|+n} | (x,y) \text{ satisfies } (2),(5)-(8)\}\}. \quad (12)$$

Note that P_{QS}^{nb} is contained in the intersection of P_{QK}^{nb} and P_{C}^{nb}. Thus any valid inequality for either P_{QK}^{nb} or P_{C}^{nb} is valid for P_{QS}^{nb}. Note also that for the CCCP and QSTSP we consider in this paper, we assume that the empty cycle is considered as a feasible solution, whereas in Bauer [3], it is not considered as a feasible solution. No feasible cycles with one or two nodes exists.

The contribution of this paper is to study the QK polytope and a QSTS polytope modelled without the y variables, denoted by $\tilde{P}_{QS}^{nb}$. We show that $\tilde{P}_{QS}^{nb}$ and P_{QS}^{nb} are in fact describing the same set of feasible solutions for the QSTSP, and that any facet-defining inequality defined for $\tilde{P}_{QS}^{nb}$ is also facet-defining for P_{QS}^{nb}. Then we work on $\tilde{P}_{QS}^{nb}$ and P_{QK}^{nb}: we establish the dimensions of these polytopes, and for each of them, we develop several classes of constraints and prove that they are facet-defining.

A.3 Polyhedral results for the QK polytope

In this section, we present our polyhedral results on the dimension of P_{QK}^{nb} and that four classes of constraints are facet-defining. The first class is the non-negativity constraint on z_e, the following two classes are generalizations of (3) and (4) respectively and the last class of constraints are obtained by modifying (6).

In what follows, we use incidence vectors $(y, z) \in \{0,1\}^{|V|+|E|}$, for $y \in \{0,1\}^{|V|}$ and $z \in \{0,1\}^{|E|}$ to represent our solutions. Each element in a vector corresponds to a node $j \in V$ or an edge $(i,j) \in E$. We also use $e_j \in \{0,1\}^{|V|+|E|}$, for any $j \in V$, to represent a vector with the value of the element corresponding to node j equals 1 and the values of all other elements equal 0; and use $e_{ij} \in \{0,1\}^{|V|+|E|}$, for any $(i,j) \in E$, to represent a vector with the value of the element corresponding to edge (i,j) equals 1 and the values of all other elements equal 0.

Theorem A.1 *Given any $G = (V, E)$, $2 \leq b \leq |V|$, the dimension of the QK polytope, P_{QK}^{nb}, is $|E| + |V|$, i.e., it is full dimensional.*

Proof. Consider the following feasible solutions:

1. $(y, z)^0 = 0$;
2. $(y, z)^1 = e_j$, for all $j \in V$; and
3. $(y, z)^2 = e_i + e_j + e_{ij}$, for all $(i,j) \in E$.

Clearly, these give us $|E| + |V| + 1$ affinely independent feasible solutions, and therefore the dimension of the QK polytope is $|E| + |V|$. □

Theorem A.2 *Given any $G = (V, E)$, $2 \leq b \leq |V|$ the constraints, given as*

$$z_f \geq 0, \quad \forall f \in E, \tag{13}$$

are facet-defining for P_{QK}^{nb}.

Proof. We need to show that the dimension of $F = P_{QK}^{nb} \cap \{z_f = 0\}$ is $|E| + |V| - 1$. First of all, it is trivially true that F defines a proper face and therefore $\dim(F) \leq |E| + |V| - 1$. Now consider the following feasible solutions:

1. $(y, z)^0 = 0$;
2. $(y, z)^1 = e_j$, for all $j \in V$; and
3. $(y, z)^2 = e_i + e_j + e_{ij}$, for all $(i, j) \in E \setminus \{f\}$.

Clearly, these give us $|E| + |V|$ affinely independent feasible solutions, and therefore $\dim(F)$ is $|E| + |V| - 1$. $\square$

Theorem A.3 *Given any $G = (V, E)$, $3 \leq b \leq |V|$ the constraints, given as:*

$$\sum_{e \in (i,S)} z_e \leq y_i + \sum_{e \in E(S)} z_e, \forall i \in V, S \subseteq V \setminus \{i\}, |S| \geq 2. \tag{14}$$

are facet-defining for P_{QK}^{nb}.

Note that (3) is a special case of (14).

Proof. Let $F = P_{QK}^{nb} \cap \left\{ \sum_{e \in (i,S)} z_e = y_i + \sum_{e \in E(S)} z_e \right\}$. Since $(y, z) = e_j$, for any $j \in V \setminus \{i\}$, does not satisfy the constraint at equality, $\dim(F) \leq \dim(P_{QK}^{nb}) - 1$. Now, we show that $\dim(F) \geq \dim(P_{QK}^{nb}) - 1$ by finding exactly $\dim(P_{QK}^{nb}) = |V| + |E|$ affinely independent feasible solutions that satisfy the constraints at equality. We do so by sequentially introducing the following vectors, each representing a feasible solution.

1. $(y, z)^1 = \{0\}$;
2. $(y, z)^2 = \{(y, z)_j^2 \mid \forall j \in V \setminus \{i\}\}$ where $(y, z)_j^2 = e_j$, (we have $|V| - 1$ of these solutions);
3. $(y, z)^3 = \{(y, z)_{ij}^3 \mid \forall j \in S\}$ where $(y, z)_{ij}^3 = e_i + e_j + e_{ij}$, for all $j \in S$, (we have $|S|$ of these solutions);
4. $(y, z)^4 = \{(y, z)_{jk}^4 \mid \forall j \in S, k \in \bar{S} \setminus \{i\}\}$ where $(y, z)_{jk}^4 = e_j + e_k + e_{jk}$, (we have $|(S, \bar{S} \setminus \{i\})|$ of these solutions);
5. $(y, z)^5 = \{(y, z)_{jk}^5 \mid \forall j, k \in \bar{S} \setminus \{i\}, j < k\}$ where $(y, z)_{jk}^5 = e_j + e_k + e_{jk}$, (we have $|E(\bar{S} \setminus \{i\})|$ of these solutions);
6. $(y, z)^6 = \{(y, z)_{jk}^6 \mid \forall j, k \in S, j < k\}$ where $(y, z)_{jk}^6 = e_i + e_j + e_k + e_{ij} + e_{ik} + e_{jk}$, (we have $|E(S, S)|$ of these solutions); and

7. $(y,z)^7 = \{(y,z)^7_{jk} \quad | \quad j \in S, \forall k \in \bar{S} \setminus i\}$, where $(y,z)^7_{jk} = e_i + e_j + e_k + e_{ij} + e_{ik} + e_{jk}$, (we have $|\bar{S}| - 1$ of these solutions).

Hence, we have $|V|+|E|$ affinely independent feasible solutions in total and thus the theorem is proved. □

Theorem A.4 *Given any $G = (V, E)$, $3 \le b \le |V|$, the constraints, given as:*

$$\sum_{e \in E(S)} z_e + 1 \ge \sum_{i \in S} y_i, \quad \forall S \subseteq V, |S| \ge 2 \tag{15}$$

are facet-defining for the QK polytope.

Note that (4) is a special case of (15).

Proof. Let $F = P^{nb}_{QK} \cap \left\{ \sum_{e \in E(S)} z_e + 1 = \sum_{i \in S} y_i \right\}$. Since $(y,z) = 0$ does not satisfy the constraint at equality, $\dim(F) \le \dim(P^{nb}_{QK}) - 1$. Now, we show that $\dim(F) \ge \dim(P^{nb}_{QK}) - 1$ by finding exactly $\dim(P^{nb}_{QK}) = |V| + |E|$ affinely independent feasible solutions that satisfy the constraints at equality. We do so by taking the following steps.

1. $(y,z)^1 = \{(y,z)^1_i \quad | \quad \forall i \in S\}$, where $(y,z)^1_i = e_i$ (we have $|S|$ of these solutions);
2. $(y,z)^2 = \{(y,z)^2_{ij} \quad | \quad \forall i \in S,\ j \in V\}$, where $(y,z)^2_{ij} = e_i + e_j + e_{ij}$, (we have $|(S,S)| + |(S,\bar{S})|$ of these solutions);
3. $(y,z)^3 = \{(y,z)^3_k \quad | \quad \forall k \in \bar{S}\}$, where $(y,z)^3_k = e_i + e_j + e_k + e_{ij} + e_{ik} + e_{jk}$, for a fixed $i \in S$, and a fixed $j \in S \setminus \{i\}$, (we have $|\bar{S}|$ of these solutions); and
4. $(y,z)^4 = \{(y,z)^4_{jk} \quad | \quad \forall j,k \in \bar{S}\}$, where $(y,z)^4_{jk} = e_i + e_j + e_k + e_{ij} + e_{ik} + e_{jk}$, for a fixed $i \in S$, (there are $|(\bar{S},\bar{S})|$ of these solutions).

It is obvious that the $|S| + |(S,S)| + |(S,\bar{S})|$ feasible solutions introduced in Step 1 and Step 2 are affinely independent. We now show, by contradiction, that the solutions introduced in Step 3 are in fact affinely independent to any of the previously introduced solutions. We assume that, w.l.o.g., the first solution introduced in Step 3 is $(y,z)^3_l$, for any $l \in \bar{S}$, and that $(y,z)^3_l = \sum_i \lambda_i (y,z)^1_i +$

$\sum_{ij} \mu_{ij}(y,z)^2_{ij}$, for some $\lambda \in \mathbb{R}^{|S|}$, $\mu \in \mathbb{R}^{|(S,S)|+|(S,\bar{S})|}$, $(\lambda,\mu) \neq 0$. Now, to obtain the elements in $(y,z)^3_l$ corresponding to the z variables, we need $\mu_{ij} = \mu_{il} = \mu_{jl} = 1$, and $\mu_f = 0$ for all $f \in E \setminus \{(i,j),(i,l),(j,l)\}$. Observe that in $(y,z)^1$, the value of the elements corresponding to variable y_l is always 0, (since $l \in \bar{S}$), so y^3_l has a value of 2 instead of 1. Hence there is a contradiction. Clearly $(y,z)^3$ are independent as elements in $\bar{S}$ are all distinct, we conclude that the incidence vectors in $(y,z)^3$ are all affinely independent. Last of all, the solutions introduced in Step 4, i.e. $(y,z)^4$ are obviously affinely independent to all the previously introduced solutions. Thus the theorem is proved. □

Theorem A.5 *Given any $G=(V,E)$, $3 \leq b \leq |V|-1$, the constraints, given as:*

$$\sum_{e\in\delta(i)} z_e \leq (b-1)y_i, \quad \forall i \in V \tag{16}$$

are facet-defining for P^{nb}_{QK}.

Constraint (16) can be obtained by multiplying (6) with y_i and noting that $y_iy_i = y_i$ and $y_iy_j = z_{ij}$.

Proof. We need to show that the dimension of $F = P^{nb}_{QK} \cap \{\sum_{e\in\delta(i)} z_e = (b-1)y_i\}$ is $|E|+|V|-1$. Since $(y,z)=e_i$ does not satisfy constraint (16) at equality, $\dim(F) \leq |E|+|V|-1$. Now consider the following feasible solutions:

1. $(y,z)^0 = 0$;
2. $(y,z)^1 = e_k$, for all $k \in V\setminus\{i\}$, (we have $|V|-1$ of these solutions);
3. $(y,z)^2 = e_k + e_l + e_{kl}$, for all $\{k,l\} \subseteq V\setminus\{i\}$, (we have $|E|-(|V|-1)$ of these solutions); and

Clearly, these $|E|+1$ points are affinely independent, and satisfy (16) at equality.

Now, we are left with finding $|V|-1$ affinely independent feasible solutions. We do so by inspecting the set of all feasible solutions that selects exactly b nodes: the node i plus $b-1$ other nodes from the set $V\setminus\{i\}$. We define such a set to be $\Omega^{V,b} = \{I_1,\dots,I_m \mid I_l = \{i\}\cup U, |I_l| = b, \quad \forall l = 1,\dots,m\}$, where $U \subset V\setminus\{i\}$. (Note that m is finite). We denote I_l by $\{i, j^l_1, \dots, j^l_{b-1}\}$ for each $l = 1,\dots,m$.

Our inductive hypothesis is that $\Omega^{V,b}$ contains precisely $|V|-1$ affinely independent feasible solutions. Our proof takes the following steps: Step 1 concerns the initial case for $|V|=4$ and $b=3$; Step 2 concerns induction on $|V|$ while holding b constant; and Step 3 concerns induction on both b and $|V|$.

Step 1. We have found precisely 3 affinely independent feasible solutions for the case when $|V|=4$ and $b=3$.

Step 2. We assume that our inductive hypothesis is true for $|V|=4,\dots,s$ and $b=t$. We now show that it is true for $|V|=s+1$ and $b=t$. Consider the QKP defined on $\tilde{G}=(\tilde{V},\tilde{E})$, for $\tilde{V}=V\cup\{q\}$, $\tilde{E}=(q,V)\cup E(V)$. We show that $\Omega^{\tilde{V},t}$ contains exactly s affinely independent feasible solutions. By our inductive hypothesis, there exists $\Omega^{V,t}$ that contains $s-1$ affinely independent feasible solutions, and w.l.o.g., let these $s-1$ solutions be $I_1,\dots,I_{s-1}$. As b was held constant at t, these $s-1$ points are also feasible for $\tilde{G}$ and satisfy (16) at equality. Now consider a new solution $I_s=\{i,j_1^1,\dots,j_{t-2}^1,q\}$. Clearly, I_s is affinely independent to any of the previously introduced solutions (wherein q is never used), and it satisfy (16) at equality.

Step 3. We assume that our inductive hypothesis holds for $|V|=4,\dots,s$, $b=3,\dots,t$, for $t\leq s-1$, and prove that it holds for $|V|=s+1$ and $b=t+1$. Recall $I_1,\dots,I_{s-1}$ defined in Step 2. First, consider $I'_s=I_1\cup\{k\}$, for $k\in V\setminus I_1$, which uses exactly $t+1$ nodes, and is affinely independent to $(y,z)^1$ and $(y,z)^2$ (as the node i is never selected therein). Then we define $I'_l=I_l\cup\{q\}$, for all $l=1,\dots,s-1$, and thus obtain $s-1$ affinely independent feasible solutions each selecting $t+1$ nodes. These are affinely independent to $(y,z)^1$, $(y,z)^2$, and I'_s due to the use of node q. Thus completes the proof. □

A.4 Polyhedral results for the QSTS polytope

In this section, we present our polyhedral results for the QSTS polytope, $\tilde{P}^{nb}_{QS}$. (Recall that this concerns the formulation without the y variables). We first establish the dimension of $\tilde{P}^{nb}_{QS}$ and establish the links between $\tilde{P}^{nb}_{QS}$ and P^{nb}_{QS}. We then prove that five classes of constraints are facet-defining for $\tilde{P}^{nb}_{QS}$. The first class of constraints concerns the relationship between x_e and z_e; the second class of constraints is a strengthened version of the subtour elimination constraints (5); and the last three classes of constraints are also facets for the QK polytope, except that herein we use $\frac{1}{2}\sum\limits_{e\in\delta(i)} x_e$ in place of y_i.

In what follows, we use incidence vectors $(x, z) \in \{0,1\}^{2|E|}$, for $x, z \in \{0,1\}^{|E|}$ to represent our solutions. We also define $(\lambda, \mu) \in \mathbb{R}^{2|E|}$, for $\lambda, \mu \in \mathbb{R}^{|E|}$, with each element in λ and μ representing an edge $e \in E$. Furthermore, when we refer to p-cycles, we refer to cycles in G that contain p nodes.

We will use the following result frequently.

Proposition A.6 *Given an undirected graph $G = (V, E)$, $|V| = 5$, let X be the matrix generated by incident vectors of all 3- and 4-cycles in G. Under the assumption that G is complete, if $X(\lambda, \mu)^T = 0$, then $\lambda_e = \mu_e = 0$ for all $e \in (E)$.*

Proof. It can be verified that X is of rank $2|E| = 20$, hence the result follows immediately. □

Theorem A.7 *Given any QSTSP defined on an undirected graph $G = (V, E)$, with $|V| \geq 5$ and $4 \leq b \leq |V|$, under the assumption that G is complete, the dimension of the QSTS polytope, $\tilde{P}^{nb}_{QS}$, is $2|E|$.*

Proof. We show, by contradiction, that the dimension of $\tilde{P}^{nb}_{QS}$ is $2|E|$. We first assume that $\tilde{P}^{nb}_{QS}$ is not full-dimensional, and hence there must be at least one equality constraint, $\lambda \cdot x + \mu \cdot z = \lambda_0$, satisfied by all feasible solutions in the polytope. Then we establish that this is true only when $\lambda_e = \mu_e = \lambda_0 = 0$, for all $e \in E$, thus implying that there is no equality constraint satisfied by all feasible solutions in the polytope and hence the polytope is full dimensional. Consider the 0-cycle defined by $(x, z) = 0$. We have $\lambda \cdot 0 + \mu \cdot 0 = \lambda_0$. Hence we get $\lambda_0 = 0$. To show that $\lambda_e = \mu_e = \lambda_0 = 0$, for all $e \in E$, consider any arbitrary subgraph $\tilde{G} = (\tilde{V}, \tilde{E})$ for $\tilde{V} \subseteq V$, $|\tilde{V}| = 5$, and $\tilde{E} = E(\tilde{V})$. Under the assumption that G is complete, $\tilde{G}$ is also complete. Now, consider a matrix X generated by the incident vectors of all the 3-cycles and the 4-cycles in $\tilde{G}$. Since $\lambda_0 = 0$, by result of Proposition A.6, we have $\lambda_e = \mu_e = 0$ for all $e \in \tilde{E}$. As $\tilde{G}$ is arbitrary in G, we have that $\lambda_e = \mu_e = 0$, for all $e \in E$. Hence the theorem is proved. □

Next, we discuss the relation between $\tilde{P}^{nb}_{QS}$ and P^{nb}_{QS}. Essentially, we try to establish that the two polytopes represent the same set of feasible solutions, and that facets found for one are facets for the other (with slight modifications). Hence, all facets of $\tilde{P}^{nb}_{QS}$ we propose in this paper are also facets for P^{nb}_{QS}. These results are echos of similar results of Bauer *et a.* [4] for the CCCP.

Proposition A.8 *For any QSTSP defined on $G = (V, E)$ where $|V| \geq 5$, and $4 \leq b \leq |V|$, we have that $dim(\tilde{P}^{nb}_{QS}) = dim(P^{nb}_{QS})$.*

Proof. Each incidence vector $(x, z) \in \mathbb{R}^{2|E|} \cap \tilde{P}_{QS}^{nb}$ can be represented by an incident vector $(x, y, z) \in \mathbb{R}^{2|E|+|V|} \cap P_{QS}^{nb}$. For any set of $2|E| + 1$ affinely independent incident vectors that spans $\tilde{P}_{QS}^{nb}$, we can get $2|E|+1$ affinely independent incident vectors in P_{QS}^{nb}. Thus $dim(P_{QS}^{nb}) \geq 2|E|$. As the rank of the degree constraints, (2), is $|V|$, clearly $dim(P_{QS}^{nb}) \leq 2|E| + |V| - |V|$, and thus $dim(P_{QS}^{nb}) = 2|E|$. $\square$

Remark A.4.1 *Since* $dim(\tilde{P}_{QS}^{nb}) = dim(P_{QS}^{nb})$, *and each incidence vector* $(x, z) \in \mathbb{R}^{2|E|} \cap \tilde{P}_{QS}^{nb}$ *can be represented by an incident vector* $(x, y, z) \in \mathbb{R}^{2|E|+|V|} \cap P_{QS}^{nb}$, *the two polytopes describe the same set of feasible solutions for the QSTSP.*

Proposition A.9 *For any QSTSP defined on* $G = (V, E)$ *where* $|V| \geq 5$, *and* $4 \leq b \leq |V|$, *if* $ax + bz \leq a_0$ *defines a facet for* $\tilde{P}_{QS}^{nb}$, *then it also defines a facet for* P_{QS}^{nb}.

Proof. The same $2|E|$ affinely independent incidence vectors $(x, z) \in \mathbb{R}^{2|E|} \cap \tilde{P}_{QS}^{nb}$ that satisfy $ax + bz \leq a_0$ at equality can be converted to $2|E|$ affinely independent incidence vectors $(x, y, z) \in \mathbb{R}^{2|E|+|V|} \cap P_{QS}^{nb}$. Hence the result. $\square$

Proposition A.10 *For any QSTSP defined on* $G = (V, E)$ *where* $|V| \geq 5$, *and* $4 \leq b \leq |V|$, *if* $\alpha x + \beta y + \gamma z \leq \alpha_0$ *defines a facet for* P_{QS}^{nb}, *then* $\tilde{\alpha} x + \gamma z \leq \alpha_0$ *also defines a facet for* $\tilde{P}_{QS}^{nb}$, *where* $\tilde{\alpha}_{ij} = \alpha_{ij} + \frac{1}{2}(\beta_i + \beta_j)$.

Proof. Suppose $\Omega = \{(x^1, y^1, z^1), \ldots, (x^{|E|}, y^{|V|}, z^{|E|})\}$ defines $2|E|$ affinely independent feasible solutions that satisfy $\alpha x + \beta y + \gamma z \leq \alpha_0$ at equality, then $\tilde{\Omega} = \{(\tilde{x}^1, 0, z^1), \ldots, (\tilde{x}^{|E|}, 0, z^{|E|})\}$, where $\tilde{x}_{ij} = x_{ij} + \frac{1}{2}(y_i + y_j)$ for all $(i, j) \in E$, (which is essentially obtained from Ω by simple linear row operations), are also affinely independent. Hence the result. $\square$

Theorem A.11 *Given any QSTSP defined on an undirected graph* $G = (V, E)$, *with* $|V| \geq 5$ *and* $4 \leq b \leq |V|$, *the constraints given below, are facet-defining for the QSTS polytope,* $\tilde{P}_{QS}^{nb}$.

$$x_e \leq z_e, \quad \forall e \in E. \tag{17}$$

Proof. We first show that the result holds for $|V| \geq 6$ and $b \geq 4$. (For $|V| = 5$ and $5 \geq b \geq 4$, one can easily prove this by generating $2|E|$ affinely independent feasible points that satisfy (17) at equality). First we show that $\tilde{P}_{QS}^{nb} \cap \{x_e = z_e\}$ defines a proper face. This is achieved by showing that there is at least one each of a solution that satisfies the constraint at equality and a solution that does not. Let $e = (i,j)$. Consider a 4-cycle given by (l,i,m,j), for $l,m \in V \setminus \{i,j\}$, clearly $x_e = 0$ and $z_e = 1$, hence the constraint is not satisfied at equality. Now consider a 3-cycle given by (l,i,j), for $l \in V \setminus \{i,j\}$, clearly $x_e = z_e = 1$ and the constraint is satisfied at equality. Thus F defines a proper face.

Now, using Theorem 3.6 in Part I.4 of Nemhauser and Wolsey [17], we need to show that if $\lambda \cdot x + \mu \cdot z = \lambda_0$ for all $x \in \tilde{P}_{QS}^{nb} \cap \{x_e = z_e\}$, then

$$\lambda_f = \begin{cases} \alpha, & \text{if } f = e, \\ 0, & \text{otherwise;} \end{cases} \qquad \lambda_0 = 0; \text{ and } \mu_f = \begin{cases} -\alpha, & \text{if } f = e, \\ 0, & \text{otherwise;} \end{cases}$$

for some $\alpha \in \mathbb{R}$.

By considering the 0-cycle, we obtain $\lambda_0 = 0$. Now consider any arbitrary subgraph $\tilde{G} = (\tilde{V}, \tilde{E})$ for $\tilde{V} \subseteq V \setminus \{i\}$, $e = (i,j)$, $|\tilde{V}| = 5$, and $\tilde{E} = E(\tilde{V})$. Obviously $e \notin \tilde{E}$ thus (17) holds with equality for all cycles in $\tilde{G}$. By Proposition A.6, we thus have $\lambda_f = \mu_f = 0$, for all $f \in \tilde{E}$. As $\tilde{G}$ is arbitrary, we have $\lambda_f = \mu_f = 0$, for all $f \in E \setminus \{e\}$. Then, consider any arbitrary 3-cycle that contains the edge e, we get $\lambda_e + \mu_e = 0$. Let $\lambda_e = \alpha$ for some $\alpha \in \mathbb{R}$, we have $\mu_e = -\alpha$ and thus the theorem is proved. $\square$

To eliminate subtours (for the QSTSP), we propose a class of constraints which strengthens (5), given as:

$$\sum_{e \in \delta(S)} x_e \geq 2z_{kl}, \ \forall \emptyset \subset S \subset V, k \in S, l \notin S. \tag{18}$$

Theorem A.12 *Given any QSTSP defined on an undirected graph $G = (V,E)$, with $|V| \geq 10$, $|V| - 5 \geq |S| \geq 5$ and $4 \leq b \leq |V|$, the constraints given by (18), are facet-defining for the QSTS polytope, $\tilde{P}_{QS}^{nb}$.*

Proof. $\tilde{P}_{QS}^{nb} \cap \left\{ \sum_{e \in (S,\bar{S})} x_e = 2z_{kl} \right\}$ defines a proper face, since (18) holds with equality for the 0-cycle while it does not for the 3-cycle (k,p,q), for $p,q \in \bar{S} \setminus \{l\}$, $p \neq q$.

Now, we are left to show that if $\lambda \cdot x + \mu \cdot z = \lambda_0$ for all $x \in \tilde{P}_{QS}^{nb} \cap \left\{ \sum_{e \in (S,\bar{S})} x_e = 2z_{kl} \right\}$, then

$$\lambda_e = \begin{cases} \alpha, & \text{if } e \in (S, \bar{S}), \\ 0, & \text{otherwise;} \end{cases} \qquad \lambda_0 = 0; \text{ and } \mu_e = \begin{cases} -2\alpha, & \text{if } e = (k, l), \\ 0, & \text{otherwise;} \end{cases}$$

for some $\alpha \in \mathbb{R}$.

By considering the 0-cycle, we have $\lambda_0 = 0$. Now, consider any arbitrary subgraph $\tilde{G} = (\tilde{S}, \tilde{E})$ for $\tilde{S} \subseteq S$, $|\tilde{S}| = 5$, and $\tilde{E} = E(\tilde{S})$. Constraint (18) holds with equality for all cycles in $\tilde{G}$. Thus, by Proposition A.6, we have $\lambda_f = \mu_f = 0$, for all $f \in \tilde{E}$. As $\tilde{G}$ is arbitrary, we have $\lambda_f = \mu_f = 0$, for all $f \in E(S)$. Analogously it can be obtained that $\lambda_f = \mu_f = 0$, for all $f \in E(\bar{S})$.

Now we obtain values for all the remaining elements in (λ, μ), i.e., we find λ_e and μ_e for all $e \in (S, \bar{S})$, by comparing cycles with 3 or 4 nodes for which (18) holds with equality. In the following, we assume arbitrary i, j, m, for $i, j \in S \setminus \{k\}, i \neq j$ and $m \in \bar{S} \setminus \{l\}$.

Let (x^1, z^1) and (x^2, z^2) be the incidence vectors of the 4-cycle defined by (k, i, j, l) and the 3-cycle defined by (k, i, j) respectively. We get:

$$\lambda \cdot x^1 + \mu \cdot z^1 - (\lambda \cdot x^2 + \mu \cdot z^2) = \lambda_{jl} + \lambda_{kl} - \lambda_{jk} + \mu_{kl} + \mu_{il} + \mu_{jl} = 0. \quad (19)$$

Note that $\lambda_{jk} = 0$ since $k, j \in S$. Analogously let (x^3, z^3) be the incidence vectors of the 4-cycle defined by (k, j, i, l). We get:

$$\lambda \cdot x^3 + \mu \cdot z^3 - (\lambda \cdot x^2 + \mu \cdot z^2) = \lambda_{kl} + \lambda_{il} - \lambda_{ik} + \mu_{kl} + \mu_{il} + \mu_{jl} = 0. \quad (20)$$

Note that $\lambda_{ik} = 0$ since $k, i \in S$. By comparing (19) with (20), we get $\lambda_{il} = \lambda_{jl}$. Let $\lambda_{il} = \alpha$, by symmetry, we get $\lambda_{il} = \alpha$ for all $i \in S \setminus \{k\}$. Now by comparing the 3-cycle (k, j, l) with (19) it follows that $\mu_{il} = 0$ for all $i \in S \setminus \{k\}$.

Comparing the 4-cycle (k, i, l, j) with the 3-cycle (k, i, j), we get $\mu_{kl} = -2\alpha$ and by comparing the 3-cycle (k, j, l) with the 4-cycle (k, j, l, i), we get $\lambda_{kl} = \alpha$. Given this and by symmetry, $\lambda_{km} = \alpha$ and $\mu_{km} = 0$ for all $m \in \bar{S} \setminus \{l\}$.

By comparing the 3-cycle (i, l, k) and the 4-cycle (i, l, m, k), we get $\mu_{im} = 0$ for all $i \in S \setminus \{k\}$ and all $m \in \bar{S} \setminus \{l\}$. Last of all, by comparing the 3-cycle (k, i, l) and the 4-cycle (k, i, m, l), we obtain $\lambda_{im} = \alpha$, for all $i \in S \setminus \{k\}$, $m \in \bar{S} \setminus \{l\}$. □

Theorem A.12 does not hold for $7 \leq |V| \leq 9$, but it actually holds for $|V| = 6$, $|S| = 5$ and $4 \leq b \leq |V|$ (and for $|S| = 1$ which is the symmetric case). This can be verified by generating $2|E|$ affinely independent feasible points that satisfy (18) at equality.

Theorem A.13 *Given any $G = (V, E)$, $|V| \geq 6$, $b \geq 4$, the constraints, given as:*

$$\sum_{e\in(i,S)} z_e \leq \frac{1}{2}\sum_{e\in\delta(i)} x_e + \sum_{e\in E(S)} z_e, \forall i \in V, S \subset V\setminus\{i\}, 1 \leq |S| \leq |V|-5, \quad (21)$$

are facet-defining for $\tilde{P}^{nb}_{QS}$.

Constraint (21) is obtained by replacing y_i by $\frac{1}{2}\sum_{e\in\delta(i)} x_e$ in (14) and is a generalization of (3).

Proof. $\tilde{P}^{nb}_{QS} \cap \left\{\sum_{e\in(i,S)} z_e = \frac{1}{2}\sum_{e\in\delta(i)} x_e + \sum_{e\in E(S)} z_e\right\}$ defines a proper face since the 0-cycle satisfies the constraint at equality whereas the 3-cycle (i, p, q), for $p, q \in \bar{S}\setminus\{i\}$, for $p \neq q$, does not. Now, we need to show that if $\lambda\cdot x + \mu\cdot z = \lambda_0$ for all $x \in \tilde{P}^{nb}_{QS} \cap \left\{\frac{1}{2}\sum_{e\in\delta(i)} x_e + \sum_{e\in E(S)} z_e = \sum_{e\in(i,S)} z_e\right\}$, then

$$\lambda_e = \begin{cases} \frac{1}{2}\alpha, & \text{if } e \in \delta(i),\\ 0, & \text{otherwise;}\end{cases} \quad \lambda_0 = 0; \text{ and } \mu_e = \begin{cases} \alpha, & \text{if } e \in E(S),\\ -\alpha, & \text{if } e \in (i,S),\\ 0, & \text{otherwise;}\end{cases}$$

for some $\alpha \in \mathbb{R}$.

By considering the 0-cycle we get $\lambda_0 = 0$. W.l.o.g. let R, k be arbitrary for $R \subseteq \bar{S}\setminus\{i\}$, $|R| = 4$ and $k \in S$. Consider the subgraph $\tilde{G} = (\tilde{V}, \tilde{E})$, $\tilde{V} \subseteq V$, $\tilde{V} = R \cup \{k\}$ and $\tilde{E} = E(\tilde{V})$. Constraint (21) holds with equality for all cycles in $\tilde{G}$, hence by Proposition A.6, $\lambda_e = \mu_e = 0$ for all $e \in \tilde{E}$. Since R and k are arbitrary, $\lambda_e = \mu_e = 0$ for all $e \in E(\bar{S}\setminus\{i\}) \cup (S, \bar{S}\setminus\{i\})$.

Let $k \in S$ and $p, q \in \bar{S}\setminus\{i\}$, $p \neq q$ be arbitrary. By comparing the cycles (k, i, p, q) and (k, i, p), we obtain $\lambda_{pq} + \lambda_{kq} - \lambda_{kp} + \mu_{kq} + \mu_{iq} + \mu_{pq} = 0$. Since $\lambda_{pq} = \lambda_{kq} = \lambda_{kp} = \mu_{kq} = \mu_{pq} = 0$, $\mu_{iq} = 0$. Since k, p and q are arbitrary, $\mu_{ip} = 0$ for all $p \in \bar{S}\setminus\{i\}$.

Let $k \in S$ and $p, q \in \bar{S}\setminus\{i\}$, $p \neq q$ be arbitrary. By comparing the cycles (k, p, i, q) and (k, p, i), we obtain $\lambda_{kq} + \lambda_{iq} - \lambda_{ki} + \mu_{kq} + \mu_{pq} + \mu_{iq} = 0$. Since $\lambda_{kq} = \mu_{kq} = \mu_{pq} = \mu_{iq} = 0$, $\lambda_{iq} = \lambda_{ki}$ are constant and let the constant be $\frac{1}{2}\alpha$. Since k, p and q are arbitrary, $\lambda_e = \frac{1}{2}\alpha$ for all $e \in \delta(i)$.

Let $k \in S$ and $p \in \bar{S}\setminus\{i\}$ be arbitrary. Consider the cycle (i, k, p) to obtain $\mu_{ik} = -\alpha$ for all $k \in S$.

If $|S| = 1$, we are done. Otherwise, let $k, l \in S$, $k \neq l$ and $p \in \bar{S} \backslash \{i\}$ be arbitrary. By comparing the cycles (k, l, i, p) and (k, i, l, p), we obtain $\lambda_{kl} + \lambda_{ip} = \lambda_{ki} + \lambda_{lp}$. Since $\lambda_{lp} = 0$ and $\lambda_{ip} = \lambda_{ki} = \frac{1}{2}\alpha$, $\lambda_{kl} = 0$. Since k, l and p are arbitrary, $\lambda_e = 0$ for all $e \in E(S)$.

Finally, let $k, l \in S$, $k \neq l$ be arbitrary. Consider the cycle (i, k, l) to obtain $\mu_{kl} = \alpha$. Since k and l are arbitrary, $\mu_e = \alpha$ for $e \in E(S)$. □

Theorem A.14 *Given any $G = (V, E)$, $|V| \geq 5$, $b \geq 5$, the constraints, given as:*

$$\sum_{e \in E(S)} z_e + 1 \geq \sum_{e \in E(S)} x_e + \frac{1}{2} \sum_{e \in \delta(S)} x_e, \forall S \subset V, 2 \leq |S| \leq |V| - 3, \tag{22}$$

are facet-defining for $\tilde{P}^{nb}_{QS}$.

Constraint (22) is obtained by replacing y_i by $\frac{1}{2} \sum_{e \in \delta(i)} x_e$ in (15). Note that (4) is a special case of (22).

Proof. $\tilde{P}^{nb}_{QS} \cap \Big\{ \sum_{e \in E(S)} z_e + 1 = \sum_{e \in E(S)} x_e + \frac{1}{2} \sum_{e \in \delta(S)} x_e \Big\}$ defines a proper face since the 3-cycle (i, j, k), $i \in S$, $j, k \in \bar{S}$ satisfies the constraint at equality and the 0-cycle does not.

Now, we need to show that if $\lambda \cdot x + \mu \cdot z = \lambda_0$ for all $x \in \tilde{P}^{nb}_{QS} \cap \Big\{ \sum_{e \in E(S)} z_e + 1 = \sum_{e \in E(S)} x_e + \frac{1}{2} \sum_{e \in \delta(S)} x_e \Big\}$, then

$$\lambda_e = \begin{cases} -\alpha, & \text{if } e \in E(S), \\ -\frac{1}{2}\alpha, & \text{if } e \in \delta(S), \\ 0, & \text{otherwise;} \end{cases} \qquad \lambda_0 = \alpha; \text{ and } \mu_e = \begin{cases} \alpha, & \text{if } e \in E(S), \\ 0, & \text{otherwise;} \end{cases}$$

for some $\alpha \in \mathbb{R}$.

W.l.o.g. let $R \subseteq S$, $|R| = 2$ and $T \subseteq \bar{S}$, $|T| = 3$ be arbitrary. Consider the subgraph $\tilde{G} = (\tilde{V}, \tilde{E})$, $\tilde{V} \subseteq V$, $\tilde{V} = R \cup T$, (so $|\tilde{V}| = 5$) and $\tilde{E} = E(\tilde{S})$. Let $\lambda_0 = \alpha$. Let the matrix X be generated by the incident vectors of all the cycles in $\tilde{G}$ for which (22) holds with equality. X is found to be of rank $2|\tilde{E}| = 20$, thus $X(\lambda, \mu)^T = \alpha$ has an unique solution. The solution is $\lambda_e = -\alpha$ for all $e \in E(R)$, $\lambda_e = -\frac{1}{2}\alpha$ for all $e \in \delta(R)$, and $\lambda_e = 0$ for all $e \in E(T)$; $\mu_e = \alpha$ for all $e \in E(R)$ and $\mu_e = 0$ for all $e \in \delta(R) \cup E(T)$. Since R is arbitrary in S, T is arbitrary in $\bar{S}$, and each $e \in E$ is in this arbitrarily chosen $\tilde{G}$, $\lambda_e = -\alpha$ for all $e \in E(S)$, $\lambda_e = -\frac{1}{2}\alpha$ for all $e \in \delta(S)$ and $\lambda_e = 0$ for all $e \in E(\bar{S})$, $\mu_e = \alpha$ for all $e \in E(S)$ and $\mu_e = 0$ for all $e \in \delta(S) \cup E(\bar{S})$. □

The following constraints are found to be very effective in practise when solving QSTSPs using a branch-and-cut method (see [19]):

$$\sum_{e\in\delta(i)} z_e \leq \frac{b-1}{2} \sum_{e\in\delta(i)} x_e,\ \forall i \in V. \tag{23}$$

Constraint (23) is obtained by replacing y_i with $\frac{1}{2}\sum_{e\in\delta(i)} x_e$ in constraint (16).

Theorem A.15 *Given any QSTSP defined on an undirected graph $G = (V, E)$, with $|V| \geq 6$ and $4 \leq b \leq |V|-1$, the constraints given by (23) are facet-defining for the QSTS polytope, $\tilde{P}^{nb}_{QS}$.*

Proof. $\tilde{P}^{nb}_{QS} \cap \left\{ \sum_{e\in\delta(i)} z_e = \frac{b-1}{2} \sum_{e\in\delta(i)} x_e \right\}$ defines a proper face, since any 3-cycle (i, j, k), $j, k \in V \setminus \{i\}$, $j \neq k$ does not satisfy the constraint at equality whereas the 0-cycle does.

Now, we are left to show that if $\lambda \cdot x + \mu \cdot z = \lambda_0$ for all $x \in \tilde{P}^{nb}_{QS} \cap \left\{ \sum_{e\in\delta(i)} z_e = \frac{b-1}{2} \sum_{e\in\delta(i)} x_e \right\}$, then

$$\lambda_e = \begin{cases} \frac{\alpha(b-1)}{2}, & \text{if } e \in \delta(i), \\ 0, & \text{otherwise;} \end{cases} \quad \lambda_0 = 0; \text{ and } \mu_e = \begin{cases} -\alpha, & \text{if } e \in \delta(i), \\ 0, & \text{otherwise;} \end{cases}$$

for some $\alpha \in \mathbb{R}$.

By considering the 0-cycle, it can be obtained that $\lambda_0 = 0$. Now, consider any arbitrary subgraph $\tilde{G} = (\tilde{V}, \tilde{E})$ for $\tilde{V} \subseteq V \setminus \{i\}$, $|\tilde{V}| = 5$, and $\tilde{E} = E(\tilde{V})$. Since all cycles in $\tilde{G}$ satisfies constraint (23) at equality, it follows from Proposition A.6 that $\lambda_f = \mu_f = 0$, for all $f \in \tilde{E}$. As $\tilde{G}$ is arbitrary, we have $\lambda_f = \mu_f = 0$, for all $f \in E(V \setminus \{i\})$.

Let $\{i_1, \ldots, i_{b-1}\} \subseteq V \setminus \{i\}$ be arbitrary. Now compare the two b-cycles $(i, i_1, i_2, i_3, \ldots, i_{b-1})$ and $(i, i_2, i_1, i_3, \ldots, i_{b-1})$. This gives $\lambda_{ii_1} + \lambda_{i_2 i_3} = \lambda_{ii_2} +$

$\lambda_{i_1 i_3}$. Since $\lambda_{i_2 i_3} = \lambda_{i_1 i_3} = 0$, λ_{ii_a} is constant for $a = 1, \ldots, b-1$ and let the constant be $\frac{\alpha(b-1)}{2}$. Since $\{i_1, \ldots, i_{b-1}\} \subseteq V \setminus \{i\}$ is arbitrary, $\lambda_{ij} = \frac{\alpha(b-1)}{2}$ for all $j \in V \setminus \{i\}$. Finally compare the b-cycle $(i, i_1, i_2, i_3, \ldots, i_{b-1})$ with the $(b-1)$-cycle $(i_1, i_2, i_3, \ldots, i_{b-1})$ to obtain $\lambda_{ii_1} + \lambda_{ii_{b-1}} - \lambda_{i_1 i_{b-1}} + \sum_{k=1}^{b-1} \mu_{ik} = 0$. Since $\lambda_{i_1 i_{b-1}} = 0$ and $\lambda_{ii_1} = \lambda_{ii_{b-1}} = \frac{\alpha(b-1)}{2}$, $\mu_{ii_a} = -\alpha$ for $a = 1, \ldots, b-1$. Since $\{i_1, \ldots, i_{b-1}\} \subseteq V \setminus \{i\}$ is arbitrary, $\mu_{ij} = -\alpha$ for all $j \in V \setminus \{i\}$ and the theorem is proved. □

A.5 Conclusion

In this paper, we studied the polyhedra of the Quadratic Knapsack Problem and the Quadratic Selective Travelling Salesman Problem. For each of these polytopes, we established its dimension, identified a number of strong constraints, and proved that these constraints are indeed facet-defining cuts. Various mathematical techniques were used in proving these results.

These results are of great significance in the implementation of a branch-and-cut method for obtaining exact solutions. The benefit of using such facet-defining cuts is that it improves the quality of the linear programming relaxation bounds.

Bibliography

[1] Balas, E.(1989) The prize collecting traveling salesman problem. *Networks* **19** 621–636

[2] Balas, E.(1995) The prize collecting traveling salesman problem. II. Polyhedral results. *Networks* **25** 199–216

[3] Bauer, P. (1997) The circuit polytope: Facets. *Mathematics of Operations Research* **22** 110–145

[4] Bauer, P., Linderoth J.T., Savelsbergh M.W.P. (2002) A branch and cut approach to the cardinality constrained circuit problem. *Mathematical Programming Ser. A* **91** 307–348

[5] Billionnet, A., Calmels, F.(1996) Linear programming for the 0-1 quadratic knapsack problem. *European Journal of Operational Research* **92** 310–325

[6] Caprara, A., Pisinger, D., Toth, P.(1999) Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing* **11** 125–137

[7] Erkut, E. (1990) Discrete p-dispersion problem. *European Journal of Operational Research* **16** 48–60

[8] Fischetti, M., Salazar Gonzalez, J.J., Toth, P.(1995) The symmetric generalized traveling salesman polytope. *Networks* **26** 113–123

[9] Fischetti, M., Salazar Gonzalez, J.J., Toth, P.(1997) A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research* **45** 378–394

[10] Fischetti, M., Salazar Gonzalez, J.J., Toth, P.(1998) Solving the Orienteering Problem through Branch-and-Cut. *INFORMS Journal on Computing* **10** 133–148

[11] Gendreau, M., Labbe, M., Laporte, G. (1995) Efficient heuristics for the design of ring networks. *Telecommunication Systems - Modeling, Analysis, Design and Management* **4** 177–188

[12] Gendreau, M., Laporte, G., Semet, F. (1998) A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks* **32** 263–273

[13] Gouveia, L., Manuel Pires, Jose(2001) Models for a Steiner ring network design problem with revenues. *European Journal of Operational Research* **133** 21–31

[14] Grötschel, M., Padberg, M.W. (1985) “Polyhedral Theory” in E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, *The Travelling Salesman Problem* Wiley

[15] Johnson, E.L., Mehrotra, A., Nemhauser, G.L.(1993) Min-cut clustering. *Mathematical Programming* **62** 133–151

[16] Laporte, G., Martello, S.(1990) The selective travelling salesman problem. *Discrete Applied Mathematics* **26** 193–207

[17] Nemhauser, G.L., Wolsey, L.A. (1998) *Integer and Combinatorial Optimization* Wiley.

[18] Pisinger, D.(1999) Exact Solution of “p”-dispersion Problems. *DIKU-rapport 99/14*

[19] Stidsen, T., Thomadsen, T. Optimal Design of Hierarchical Ring Networks using Branch-and-Price. *Work in progress.*

Appendix B

A Branch-and-Cut Algorithm for the Quadratic Selective Travelling Salesman Problem

Submitted for *Telecommunication Systems*

A Branch-and-Cut Algorithm for the Quadratic Selective Travelling Salesman Problem

Tommy Thomadsen[1] and Thomas Stidsen [2]

Abstract

A well-known extension of the Travelling Salesman Problem (TSP) is the Selective TSP (STSP). In the STSP, each node has an associated revenue and instead of visiting all nodes, the most profitable nodes, taking the travelling costs into account, are visited. The Quadratic STSP (QSTSP) possesses the additional complication that a revenue is associated with each pair *of nodes, which can be gained only if* both *nodes are visited. The QSTSP resemble ring network design and is a subproblem when designing hierarchical ring networks.*
We describe an integer linear programming model for the QSTSP. The QSTSP is solved both by applying two construction heuristics and by applying a branch-and-cut algorithm.
Computational results are presented for two types of budget constraints limiting the length of the ring and the number of nodes in the ring, respectively. In addition, more or less restrictive budgets are considered. The construction heuristics are fast, but obtain solutions which are far from optimal. One heuristic is best when restrictive budgets are considered, the other heuristic is best when ample budgets are considered. The branch-and-cut algorithm determines optimal solutions at much higher computation times than the heuristics. The computation time depends on the budget and is highest for budgets that are neither ample nor restrictive. All problems with up to 50 nodes are solved within one hour of computation time.

Keywords: Ring Networks, (Selective) Traveling Salesman Problem, Branch-and-Cut, Integer Programming

[1]Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark. Email: tt@imm.dtu.dk
[2]Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark. Email: tks@imm.dtu.dk

B.1 Introduction

When building telecommunication networks, the ring topology is widely used due to its inherent single link/node breakdown protection and its simple and fast restoration scheme. When a service provider chooses to build a ring, it is of major importance to build a ring which affords a good tradeoff between the revenue obtained and the node- and link-costs of building the ring. The ring should also be implementable in the technology chosen, e.g. the number of nodes and/or the length of the ring may be limited due to transmission delay.

Single-ring design can be generalized to multi-ring design where the service provider is contracted to provide service to a set of customers using one or more rings. In that case the same tradeoff as for single-ring design is apparent, but for more rings at the same time. Evaluating the quality of single rings is a good start when evaluating such multi-ring designs. In particular one could imagine generating a set of potentially good rings and choosing a subset of those which covers all customers.

The single-ring design can be seen as an extension to the classical Travelling Salesman Problem (TSP) and it has been termed, both the Orienteering problem, Prize collecting TSP, and Selective TSP (STSP). We will use the term STSP. In the STSP, a revenue is associated with each customer and the idea is then to establish a good tradeoff between the design cost and the revenue obtained for including nodes in the ring. The STSP is studied in e.g. [2], [3], [9], [11], and [12]. A related problem is the Generalized TSP (GTSP) which is considered in [6] and [7]. In GTSP the set of nodes is divided into disjoint subsets denoted clusters and the shortest ring which passes at least one node in each cluster must be determined.

The most notable difference between the different versions of the STSP and the GTSP is how the size of the ring is limited. The limit may be a minimum size or a maximum size determined directly or indirectly. Some of the possibilities, which are sometimes combined are: 1) A tradeoff in design cost and revenue collected, i.e. both the design cost and the revenue are present in the objective [2, 3]; 2) A lower bound on the revenue collected [2, 3]; 3) An upper bound on the design cost [9, 11, 12]; and 4) Controlled by clusters of nodes as in GTSP [6, 7].

Another notable difference between the STSP and the GTSP is whether a particular node, usually denoted the depot, is required to be in the ring [9, 11, 12] or not [2, 3, 6, 7]. In [9], a non-empty subset of nodes is required to be in the ring. Finally GTSP [6, 7] falls a bit out of category. Here, a set of clusters of nodes are given, and for each of the clusters one node in the cluster has to be in the ring, but which node is to be determined.

The Quadratic STSP (QSTSP) possesses the additional complication that a revenue is associated with each pair of nodes, which can be gained only if *both* nodes are in the ring. The QSTSP is considered in [8] and [10]. In [8] a model for the QSTSP with quadratic objective is presented and heuristics are developed and tested. The size of the ring is controlled by an upper limit on the design cost combined with a tradeoff in the objective between the design cost and the revenue associated with each pair of nodes. No nodes are required to be in the ring. In [10] alternative models for the QSTSP are considered. The size of the ring is controlled by a bound on the number of nodes in the ring and also some nodes are required to be in the ring. Three integer linear models are evaluated by solving test instances with up to 80 nodes using Cplex. For the test instances, all pairs of nodes have a demand, but only a fixed number of the shortest edges are considered. For the largest instances with 80 nodes, only the 200 shortest edges are considered. Some instances cannot be solved within 24 hours. Finally [13] presents a polyhedral study of the QSTSP.

We present an integer linear programming model for the QSTSP and suggest a branch-and-cut algorithm. Given the similarities with the GTSP, cuts and corresponding separation-routines in [7] have been found very useful.

The outline of the paper is as follows. We present the model for the QSTSP in Section B.2. In Section B.3 we describe the branch-and-cut algorithm and in Section B.4 we describe two construction heuristics. In Section B.5 we test the algorithms with two budget constraints limiting the length of the ring and the number of nodes in the ring respectively. Finally we give conclusions in Section B.6.

B.2 The Model

In this section we present an integer linear programming model for the QSTSP. The model is in essence a linearization of the model presented in [8]. We restrict ourselves to the undirected/symmetric version of QSTSP as is the case in [8] and [10].

Let $V = \{1, \ldots, n\}$ be a set of vertices and $E = \{\{i, j\} : i \in V, j \in V \backslash \{i\}\}$ be a set of node-pairs corresponding to undirected edges. Let r_i be the revenue (possibly negative) obtained if $i \in V$ is in the ring. Associate with each node-pair $\{i, j\} = e$ a cost $c_e \geq 0$ incurred if e is chosen and a revenue $r_e \geq 0$ obtained if i and j are in the ring.

Let $S \subseteq V$ and define $\delta(S) \subseteq E$ to be the set of edges with one endpoint in S and one endpoint not in S, i.e. $\delta(S) = \{\{i,j\} \in E : i \in S, j \notin S\}$. For notational convenience, $\delta(\{i\})$ is abbreviated by $\delta(i)$. Furthermore define $\gamma(S) \subseteq E$ to be the set of edges with both endpoints in S, i.e. $\gamma(S) = \{\{i,j\} \in E : i,j \in S\}$.

Let $x_e = 1$ if $e \in E$ is in the ring, 0 otherwise. Let $y_i = 1$ if $i \in V$ is in the ring, 0 otherwise. And finally let $z_e = 1$, $\{i,j\} = e \in E$ if i and j are both in the ring. Given this, the QSTSP is formulated as follows.

$$\text{Maximize} \qquad \sum_{i \in V} r_i y_i + \sum_{e \in E} r_e z_e - \sum_{e \in E} c_e x_e \tag{1}$$

Subject To

$$\sum_{e \in \delta(i)} x_e = 2y_i \text{ for } i \in V \tag{2}$$

$$z_e \leq y_i \text{ for } i \in V, e \in \delta(i) \tag{3}$$

$$z_e \geq y_i + y_j - 1 \text{ for } \{i,j\} = e \in E \tag{4}$$

$$\sum_{e \in \gamma(S)} x_e \leq \sum_{i \in S \setminus \{k\}} y_i - y_l + 1 \text{ for } \emptyset \subset S \subset V, k \in S, l \notin S \tag{5}$$

$$x_e \in \{0,1\} \text{ for } e \in E \tag{6}$$

$$y_i \in \{0,1\} \text{ for } i \in V \tag{7}$$

$$z_e \in \{0,1\} \text{ for } e \in E \tag{8}$$

The objective (1) is the sum of the revenues obtained for nodes and pairs of nodes in the ring subtracted the cost of edges in the ring. Constraints (2) ensure that if i is on the ring, then exactly two edges are incident to i, otherwise no edges are incident to i. Constraints (3) and (4) connect the y and z variables. Constraints (3) ensure that $z_e = 0$ if one of the endpoint nodes is not in the ring and constraints (4) ensure that if both endpoint nodes are in the ring, then $z_e = 1$.

Constraints (5) are the Generalized Subtour Elimination Constraints (GSECs) in inner form [6, 7]. The GSECs ensure that at most one subtour exists. Assume S contains the nodes of a subtour and assume that a node in another subtour $l \notin S$ exists. For any $k \in S$, the corresponding GSEC cut is violated, since $\sum_{e \in \gamma(S)} x_e = |S| > \sum_{i \in S \setminus \{k\}} y_i - y_l + 1 = |S| - 1 - 1 + 1 = |S| - 1$ and thus cuts off such solutions. Thus, GSECs always exist which cut off any solution with more than one subtour. Finally constraints (6), (7), and (8) are the integer domain constraints.

The GSECs can be reformulated in outer form, leading to the following constraints.

$$\sum_{e\in\delta(S)} x_e \geq 2(y_k + y_l - 1) \text{ for } \emptyset \subset S \subset V, k \in S, l \notin S \tag{9}$$

The reformulation is obtained by using constraints (2) and (5). The GSEC constraints are easier to interpret in outer form than in inner form. If a node is selected on both sides of a cut, then at least two edges crossing the cut must be selected. Actually it suffices to consider GSECs such that $|S| \leq \lfloor n/2 \rfloor$. To see why, assume that a GSEC with $|S'| > \lfloor n/2 \rfloor, S \subset V$, $k' \in S'$, and $l \notin S'$ exists. A corresponding GSEC is $S = V \backslash S'$, hence $|S| \leq \lfloor n/2 \rfloor$, $k = l'$, and $l = k'$. Obviously this GSEC is violated if and only if the original GSEC is violated, hence it suffices to consider GSECs such that $|S| \leq \lfloor n/2 \rfloor$.

For $|S| \leq \lfloor n/2 \rfloor$, the number of non-zero variables are smaller for GSECs in inner form than in outer form. Hence GSECs in inner form are more suitable for cutting-plane approaches [6] and are thus used.

As noted earlier, it is customary to limit the ring-size in some way, at least for the STSP. We will consider two budget constraints, one which limits the length of the ring, and one which limits the number of nodes in the ring. Assuming b_x is the max length of the ring and b_y is the maximum number of nodes in the ring, the budget constraints are given below.

$$\sum_{e\in E} c_e x_e \leq b_x \tag{10}$$

$$\sum_{i\in V} y_i \leq b_y \tag{11}$$

Constraint (10) limits the length of the ring and is the same budget constraint used in [8]. Constraint (11) limits the number of nodes in a ring and corresponds exactly to a subproblem arising when using column generation to solve a hierarchical multi-ring network design problem as described in [15].

We note that the c_e's in (10) are the same c_e's as in the objective (1). The following constraint generalizes both (10) and (11).

$$\sum_{e\in E} b_e x_e \leq b \tag{12}$$

Replacing b_e by c_e we obtain (10) and by applying (2), we obtain (12) with $b_e = \frac{1}{2}$ for all $e \in E$. If the budget constraint (11) is used and arbitrary coefficients were multiplied on the y_i's in (11), then QSTSP is actually a generalization of the Quadratic Knapsack Problem considered in e.g. [4] and [5].

The mathematical model is similar to several other models suggested in e.g. [2, 6, 7, 9, 11, 12] with the exception of the quadratic revenue. In [8] a model is presented with a quadratic revenue and budget constraint (10). The presented model has a quadratic objective and the model is obtained by replacing z_e with $y_i y_j$ and leaving out the now redundant constraints (3) and (4). Also no revenue exists for nodes, i.e. $r_i = 0$.

Integer requirements on z_e are unnecessary, since z_e is in effect a bookkeeping variable. Furthermore, since $r_e \geq 0$ and we maximize the demand-revenue ($\sum_{e \in E} r_e z_e$), constraints (4) are not binding and thus redundant.

The advantage of the suggested model over the one given in [8] is clearly that the model is linear, hence the usual tools for solving integer programs can be used. The disadvantages are the additional $(n^2 - n)/2$ variables (z_e), the additional $n^2 - n$ constraints of type (3), and the additional $(n^2 - n)/2$ constraints of type (4).

In [6] the polytope of the GTSP is studied and it is proved that the GSECs are facet defining for the GTSP. Additionally [6] studies generalized comb constraints which are proven to be facet defining for the GTSP. A computational study of a branch-and-cut algorithm using these cuts are presented in [7], and shows that the GSECs are far more important than the generalized comb constraints. In most problems solved, no or very few (≤ 5) generalized combs are generated. With this in mind, we will not consider the generalized combs any further in this paper.

B.2.1 Additional Cuts

In order to strengthen the LP-relaxation, we introduce additional valid cuts which cuts off fractional and hence infeasible solutions. The following constraints strengthen the LP-relaxation.

$$x_e \leq y_i \text{ for } i \in V, e \in \delta(i) \tag{13}$$

Constraints (13) are valid, since if an edge is in the ring, then both endpoint nodes are selected. Also, constraints (13) dominate the GSEC constraints (5) for $|S| = 2$ as can be seen from the following. Let $S = \{i, j\}$, $e = \{i, j\}$, $k = j$, and $l \notin S$, then the following GSEC is obtained:

$$x_e \leq y_i - y_l + 1 \tag{14}$$

This constraint is valid whenever constraints (13) are valid since $y_l \leq 1$ and thus $x_e \leq y_i \leq y_i - y_l + 1$. Furthermore, values exist for which constraint (14) is valid but constraints (13) are not (e.g. $x_e = 1$, $y_i = 0$, and $y_l = 0$). Thus constraints (13) dominate constraint (14). Computational experiments indicate that constraints (13) improve the performance only marginally.

The following constraints are valid only when the number of nodes in the ring is limited i.e. constraint (11) is used.

$$\sum_{j:e=\{i,j\}\in E} z_e \leq (b_y - 1)y_j \text{ for } i \in V \tag{15}$$

The constraints are described in e.g. [4] and [5] for the quadratic knapsack problem. The constraints improve the value of the LP-relaxation considerably for both the quadratic knapsack problem and for the QSTSP. The constraints can be seen to be valid by multiplying (11) by y_j and replacing $y_i y_j$ with z_e for $e = \{i, j\}$ and $y_j y_j$ by y_j.

B.3 Branch-and-Cut Algorithm

We solve the QSTSP using a branch-and-cut algorithm. A branch-and-cut algorithm is a branch-and-bound algorithm where the bounds are obtained by solving the LP-relaxation of the problem, possibly including some additional cuts. We have implemented the branch-and-cut algorithm using the Branch-Cut-and-Price framework (BCP) which is part of COIN [1]. BCP is, as its name indicates, a framework for developing branch-cut-and-price algorithms. The choice of using BCP instead of implementing from scratch was made to speed up development. We have used BCP standard setups and classes whenever possible. The branch-and-cut algorithm is shown in Figure B.1.

An initial feasible solution is generated by selecting the best solution from the two heuristics described in Section B.4. The algorithm maintains a set of subproblems, initially containing the root node problem only. The root node problem consists of (1), (2), bounds on variables and one of the budget constraints (10) or (11). If constraint (11) is used, we also include (15).

A subproblem is selected and the LP-relaxation of the subproblem including additional cuts are solved. If any violated cuts can be determined, the cuts are added to the subproblem and the subproblem is resolved. This is done until no more violated cuts can be determined. We generate (3), (13), and the GSECs (5). Generation of constraints (3) and (13) is done by checking whether any of the constraints are violated. Since there are $(n^2 - n)/2$ of each of those, the computation time is $O(n^2)$. Generation of GSECs is described in the following section.

```
INCUMBENT = A feasible solution obtained heuristically.
SET_OF_SUBPROBLEMS = {Root node problem}
while SET_OF_SUBPROBLEMS ≠ ∅
   Select subproblem and remove it from SET_OF_SUBPROBLEMS
   do
      Solve LP-relaxation of subproblem including generated cuts
      Generate and add violated cuts
   while New cuts added
   Let OBJ_VAL = objective value of LP-relaxation
   if LP solution is feasible, hence integer and OBJ_VAL > INCUMBENT:
      Update incumbent: INCUMBENT = OBJ_VAL
   else if OBJ_VAL ≤ INCUMBENT or subproblem is infeasible:
      Continue
   else Branch:
      Add two subproblems to SET_OF_SUBPROBLEMS
end while
```

Figure B.1: *The branch-and-cut algorithm*

Given a bound on the subproblem, three possibilities exists. 1) The subproblem is feasible, hence a candidate solution has been identified. If the candidate is better than the incumbent, the candidate takes the place of the incumbent. 2) The bound is worse than the incumbent. The subproblem is not considered any further, since it will not lead to any better solutions. 3) Nothing can be determined, so we have to branch. Create two new subproblems and continue.

Subproblems are considered depth first. This allows for reuse of the obtained LP-tableau in the following iteration. Variable branching is applied. Note that the z variables are integer if the y variables are integer, thus the z variables are not considered for branching. The variable to branch on is determined as follows. First the y variables are considered. If any fractional y variable exists, the y variable with a value closest to $1/2$ is chosen. If no fractional y variable exists, the x variables are considered. Similarly the x variable with a value closest to $1/2$ is chosen. Two new subproblems are created. In one subproblem we set the chosen variable equal to one and in the other subproblem we set the chosen variable equal to zero. The subproblem with the chosen variable equal to one is considered first.

B.3.1 Separation of GSECs

Separation of GSECs is due to [7] which presents an optimal and a heuristic separation routine. Optimal separation can be achieved in $O(n^4)$ time, but proves to be ineffective for GTSP. For completeness the optimal separation routine is, however, described.

Assume (x_e^*, y_i^*, z_e^*) is the value of an optimal solution to the LP-relaxation of the problem. Set up an undirected graph with n nodes and edge capacities equal to x_e^*.

Optimal separation of GSECs is best described based on constraints (9). Recall the interpretation of the constraints: If a node is selected on both sides of a cut, then at least two edges crossing the cut must be selected. Thus separation can be done by for all pairs of nodes k, l computing a minimum cut separating k and l. Given the minimum cut, i.e. a set of nodes S, evaluate (9) and check whether it is violated. This can be achieved in $O(n^5)$ time, since a minimum cut can be computed in $O(n^3)$ time and $O(n^2)$ pairs of nodes exists.

By using the following observation, the computation time can be reduced to $O(n^4)$. For a given $S' \subset V$, the most violated cut is obtained for $k' \in S'$ and $l' \notin S'$ such that $k' = \arg\max_{i \in S'}\{y_i^*\}$ and $l' = \arg\max_{i \notin S'}\{y_i^*\}$. Assume without loss of generality that $y_{k'}^* \geq y_{l'}^*$. For any GSEC defined by S, k, and l, either $k' \in S$ or $k' \notin S$. But in that case one of the most violated GSECs will have $k = k'$ if $k' \in S$ or $l = k'$ if $k' \notin S$. Thus it suffices to pick any such k' and consider pairs for this k' and all other nodes l. Thus only $O(n)$ pairs need to be considered and thus the complexity is only $O(n^4)$ in total.

In summary the separation algorithm is defined as follows. Pick a node k such that $k = \arg\max_{i \in V}\{y_i^*\}$. For all nodes $l \neq k$ compute a minimum cut between k and l. Evaluate (9) to check whether the cut is violated and if so add it in the form of (5). Thus up to $n - 1$ cuts may be added.

Computational experiments show that substantial time is spent on optimal separation, however, the most important problem is that the generated cuts do not span the graph [7]. This is explained in the following. Suppose that three disjoint sets $S_a \subset V, a = 1, 2, 3$, exist for which $\delta(S_a) = 0$. Assume the node k, chosen during optimal separation, is in S_1. Then at least two GSECs will be generated with $S = S_1$ and $l \in S_2$ or $l \in S_3$. The cut with $S = S_2$, $k \in S_2$, and $l \in S_3$ will *not* be generated. This last cut may be needed to get the best bound, and thus may have to be generated in later iterations. Thus additional iterations may be needed, thereby increasing the computation time. An alternative is to use heuristic separation as described in [7] and in the following.

Again, set up an undirected graph with n nodes and capacity of edges equal to x_e^*. The idea is then to generate a minimum spanning forest using Kruskal's algorithm. Each time two components are merged, a cut is identified and added if it is violated. Initially all nodes constitute a component. Consider all edges $e = \{i, j\}$, $x_e^* > 0$ in non-increasing order. If i and j are in different components, the corresponding components are merged. When two components are merged, let the resulting component be S. Furthermore, let $k = \arg\max_{i \in S}\{y_i^*\}$ and $l = \arg\max_{i \notin S}\{y_i^*\}$ and add the corresponding GSEC if it is violated.

Checking whether a GSEC is violated can be done in $O(n^2)$ time and this may be done at most once for each merge, i.e. $n - 1$ times, in total $O(n^3)$ time. However, this can be reduced to $O(n^2)$ time in total by doing the following. Maintain for each component the weight internally in the component and the weight to all other components. For each merge of two components, update the weights. Updating can be accomplished in linear time in n. Since at most n merges are carried out the total complexity is $O(n^2)$. The complexity of Kruskal's algorithm is $O(m \log m)$ where m is the number of edges. Since m may be up to $O(n^2)$ the complexity expressed in terms of n is $O(n^2 \log n)$. Thus the complexity of the heuristic separation is $O(n^2 \log n)$.

As mentioned, the primary advantage of the heuristic separation routine is that it generates GSECs that span the graph. This is confirmed by computational experiments. The heuristic separation routine is used in all computational experiments presented in Section B.5.

B.4 Heuristics

We have developed two types of greedy heuristics. One builds up a ring from scratch similar to the heuristics used in [8] and one removes nodes from a full ring initially including all nodes. To distinguish the two heuristics, we will denote them the construction heuristic and the deconstruction heuristic, respectively. The heuristics are used for comparison and for generating an initial solution for the branch-and-cut algorithm. Generating an initial solution for the branch-and-cut algorithm does not give a significant speed up, though.

The construction heuristic is shown in Figure B.2. For notational convenience we use the following notation: $r_{ij} = r_e$ for $\{i, j\} = e$ and $c_{ij} = c_e$ for $\{i, j\} = e$. Let Δ_{bud} be the change in budget, equal to 1 when the number of nodes is limited and equal to $c_{ik} + c_{jk} - c_{ij}$ when the length of the ring is limited.

The construction heuristic builds the ring by greedily identifying a feasible ring

Let $\{i,j\} = \arg\max\{r_{ij} + r_i + r_j - c_{ij} | \text{cost within budget}\} \in E$
Let $k = \arg\max\{r_{ik} + r_{jk} + r_k - c_{ik} - c_{jk} | \text{cost within budget}\} \in (V \backslash \{i,j\})$
Let $R = \{i,j,k\} \subseteq V$
Let $F = \{\{i,j\},\{j,k\},\{k,i\}\} \subseteq E$
while a node exists which can be added without exceeding the budget
  Compute $k \in V\backslash R$ and $ij \in F$:
  $k, i, j = \arg\max$
    $\{(\sum_{i \in R} r_{ik} + r_k + c_{ij} - c_{ik} - c_{jk})/\Delta_{bud} | \text{cost within budget}\}$
  $R = R \cup \{k\}$
  $F = F \backslash \{\{i,j\}\} \cup \{\{i,k\},\{j,k\}\}$
end while
Run 2-opt on the final set of nodes R.

Figure B.2: *The construction heuristic*

with three nodes (first four lines). The ring is then extended until the budget is entirely spent. This is done by picking, during each iteration, the node which yields the largest revenue minus additional cost relative to the increase in budget. Finally the ring is improved by running 2-opt on the selected set of nodes. Running 2-opt is defined as follows: For all non-adjacent edges ij and kl in the ring, take out ij and kl and obtain two paths. Assume i and k are on the same path. Connect the two paths by inserting edge il and jk, thus obtaining a new ring. If the new ring is shorter than the old ring, accept the new ring, otherwise keep the old ring.

The construction heuristic is aborted if it is not possible to compute a ring with three nodes. In that case the trivial solution consisting of no nodes and a value of 0 is reported.

The deconstruction heuristic is shown in Figure B.3. The deconstruction algorithm initially builds a ring containing all nodes. This is done using farthest insertion followed by running 2-opt. Farthest insertion constructs a ring as follows: Pick three nodes at random, obtaining an initial ring. Add the remaining nodes one by one by doing the following. For all nodes not in the ring, determine the shortest distance to a node in the ring and denote this value $short_i$ where i is a node not in the ring. For all nodes not in the ring, select the node which has the *highest* $short_i$ value. Insert this node i in the ring such that the increase in the ring length is as low as possible.

The initial ring is now shortened by removing one node during each iteration. The node removed is picked greedily as the node which decreases the objective the least relative to the decrease in budget. Finally 2-opt is run on the remaining set of nodes.

Compute a ring containing all nodes using farthest insertion and 2-opt.
Let $R = V$
Let F = Set of edges in the ring obtained
while Budget is exceeded
  Compute $k \in R$ and let i and j be neighbours, i.e. $\{i,k\}$, $\{j,k\} \in F$:
  $k = \arg\min\{(\sum_{i \in R \setminus \{k\}} r_{ik} + r_k + c_{ij} - c_{ik} - c_{jk})/\Delta_{bud}\}$
  $R = R \setminus \{k\}$
  $F = F \setminus \{\{i,k\},\{j,k\}\} \cup \{\{i,j\}\}$
end while
Run 2-opt on the final set of nodes R.

Figure B.3: *The deconstruction heuristic*

The idea of starting out from a solution with all nodes selected and then removing one node at a time come from Quadratic Knapsack algorithms which use the same method [4, 5].

In some cases the algorithms return solutions of negative value. These are rejected and the trivial solution consisting of no nodes and a value of 0 is reported.

B.5 Computational Results

In this section computational results are presented. It is investigated how the branch-and-cut algorithm performs and how the heuristics perform. We investigate how the budget constraint influence the problems. In particular a problem is expected to be easier if the budget is either very restrictive or very ample.

If the budget is very restrictive, the set of feasible rings is smaller than for an average restricted budget. Hence selecting one node will have more affect on the bound and hence branching may finish faster.

When the budget is ample, almost all nodes will be selected. Thus the optimal revenue is close to the total possible revenue. Such problems reduce to TSP plus a constant corresponding to the revenue. In essence, the branch-and-cut algorithm solves these problems as one would solve TSP, and in particular the bound on the optimal distance cost corresponds to the value of the LP-relaxation of the TSP. The LP-relaxation of the TSP is strong, and so a strong bound is obtained on the optimal solution for the QSTSP. Thus problems with ample budgets should be easy. The most difficult problems seems to be problems with budgets that are neither ample nor restrictive and thus have optimal solutions with approximately $n/2$ nodes.

B.5.1 Generation of Test Instances

The test instances are generated similarly to what was suggested in [8]. The original test instances are not available to us, and in addition revenues on the nodes are generated, which is not considered in [8]. n points are randomly located in the plane with coordinates uniformly distributed between 0 and 100. Coefficient c_e is the Euclidean distance rounded to integer between points i and j, $e = \{i, j\}$.

When limiting the length of the ring, set $b_x = 0.75\sqrt{n/2}$ which will have the effect that somewhat more than $n/2$ nodes will be in the optimal ring [8]. Since c_e is the coefficient in both the objective and in the budget constraint limiting the ring length, the distance-cost incurred is at most b_x. Given this, an expected total revenue of similar size is constructed. To make sure that no instances are generated where it does not pay off to have any nodes, the revenues are constructed such that the total expected revenue R is $1.15b_x$. The aim is to distribute the total revenue equally over node-revenues and demand-revenues. Thus the test instances have the property that the expected node-revenue and demand-revenue are both $R/2$.

Given the total node- and demand-revenue and the expected number of nodes in the ring ($= n/2$), the average node- and demand-revenue can be determined. The average node-revenue $\overline{r}_n$ is:

$$\frac{n}{2}\overline{r}_n = \frac{R}{2} = \frac{1.15b_x}{2} \Leftrightarrow \overline{r}_n = \frac{1.15b_x}{n} \tag{16}$$

The number of demands for l nodes is $(l^2 - l)/2$, thus the total demand revenue is:

$$R/2 = \overline{r}_d((n/2)^2 - n/2)/2 \approx \overline{r}_d((n/2)^2)/2 \tag{17}$$

Isolating $\overline{r}_d$, an expression for the average demand-revenue is obtained:

$$\overline{r}_d \approx \frac{R}{(n/2)^2} = \frac{4 \cdot 1.15b_x}{n^2} = \frac{4.6b_x}{n^2} \tag{18}$$

The revenue of a node is now uniformly distributed between 0 and $2\overline{r}_n$ and the revenue of an edge is uniformly distributed between 0 and $2\overline{r}_d$. The total revenue turns out to be higher than the expected revenue, since the number of nodes in optimal solutions is on average higher than $n/2$. 10 instances of networks with 10, 20, 30, 40 and 50 nodes have been generated.

B.5.2 Results

In this section computational results for the branch-and-cut algorithm and for the two heuristics are presented. As mentioned, BCP [1] is used for the branch-and-cut algorithm and Cplex 7.5 is used for solving LP problems. The tests are carried out on a PC with a 1.6 Ghz Intel Xeon processor running Linux.

It is investigated how the type of budget and the more or less restrictive budgets affect computation times. When the length of the ring is limited, tests are carried out for a budget of 0.2, 0.6 1.0, 1.4, and 1.8 times the b_x described in Section B.5.1. When limiting the number of nodes in the ring, b_y is set equal to 5, 10, 20, 30, 40, and 50. Results for instances where $b_y > n$ are not reported, since the results are the same as for $b_y = n$. Results given are averages over 10 random instances. All tests were completed within one hour. The results when limiting the length of the ring are given in Table B.1.

The first and second columns of Table B.1 contain the number of nodes in the test instance and the budget relative to the normal budget. The third column contains the average number of nodes in solutions, and the fourth column contains the average optimal solution. Columns five to seven contain the components of the objective value corresponding to x, y, and z, see (1). The columns eight to eleven contain the root node value after adding cuts, the gap between the root node and the optimal solution relative to the optimal solution, the number of subproblems and the maximum depth of the branch-tree. Finally the last three columns contain computation times. The total time and the two most time consuming operations, the time spent on generating cuts, and the time spent on solving LPs are reported.

The results when limiting the number of nodes in the ring are given in Table B.2. The table contains the same columns as Table B.1, except the second column which contains the maximum number of nodes in the ring, b_y. Note that distance-costs and the node- and demand-revenues all contribute significantly to the objective. This corresponds to how the test instance were generated.

The results show that the budget has considerable impact on the gap between the root node and the optimal solution. The gap is low for ample budgets, but high for restrictive budgets. The computation times are not always higher for instances with larger gaps. The instances with very restrictive budgets are solved faster than instances with average restrictive budgets. For instances with very restrictive budgets, branching on a variable has more impact on the bound than for other instances. This explains why the computation time is low regardless of the large gap and as a consequence, the number of subproblems is low. In summary, the computation times are highest when neither ample nor restrictive

n	Budget	Nodes in Solution	Optimal value	Distance cost	Node revenue	Demand revenue	Root LP-value	Gap (%)	Num. of subp.	Tree depth	Time (seconds) Total	Cut	LP
10	0.2	0.6	7.5	5.9	8.4	5.0	43.8	482.8	9.2	4.1	0.0	0.0	0.0
10	0.6	3.9	54.6	75.7	82.3	48.0	117.3	115.0	21.6	5.0	0.1	0.0	0.0
10	1.0	6.2	113.8	144.6	128.2	130.1	172.4	51.5	20.2	5.2	0.1	0.0	0.0
10	1.4	8.3	182.2	219.4	163.2	238.4	220.3	20.9	12.4	4.9	0.0	0.0	0.0
10	1.8	9.5	227.9	259.4	181.3	305.9	239.4	5.1	5.0	1.8	0.0	0.0	0.0
20	0.2	3.5	32.7	41.7	61.1	13.2	75.5	131.2	24.6	6.5	0.4	0.0	0.3
20	0.6	8.7	104.5	127.3	137.3	94.6	207.1	98.1	66.8	8.9	0.9	0.0	0.7
20	1.0	13.6	201.0	231.8	197.4	235.3	308.9	53.7	133.0	10.9	1.4	0.1	1.1
20	1.4	17.9	361.7	316.9	260.6	418.0	393.3	8.7	49.0	8.8	0.6	0.0	0.4
20	1.8	20.0	418.1	377.0	279.9	515.2	419.1	0.2	2.8	0.4	0.0	0.0	0.0
30	0.2	4.7	36.4	46.1	67.8	14.7	95.5	162.2	82.8	12.6	3.1	0.1	2.5
30	0.6	13.4	120.2	167.4	162.3	125.3	264.2	119.7	326.8	16.2	11.1	0.3	9.2
30	1.0	22.0	316.8	286.2	259.5	343.5	396.7	25.2	233.2	16.9	7.7	0.3	6.3
30	1.4	28.3	498.3	396.8	325.0	570.0	510.6	2.5	36.2	10.5	1.5	0.1	1.1
30	1.8	30.0	530.2	454.6	341.3	643.5	531.7	0.3	8.6	1.6	0.2	0.0	0.1
40	0.2	6.5	38.1	56.3	75.8	18.6	111.3	192.0	304.4	20.1	23.2	0.5	19.6
40	0.6	18.9	166.6	197.7	196.9	167.4	303.5	82.1	475.2	16.2	44.7	1.2	38.6
40	1.0	30.1	387.4	331.0	293.3	425.1	453.8	17.1	303.4	22.3	26.2	0.9	22.5
40	1.4	37.8	568.9	462.5	358.4	672.9	581.6	2.2	285.8	21.8	20.3	1.2	15.1
40	1.8	40.0	608.5	519.8	377.6	750.7	610.1	0.3	11.4	2.6	0.3	0.0	0.2
50	0.2	9.0	50.0	69.2	92.3	26.9	136.8	173.5	566.4	27.0	86.3	1.9	74.7
50	0.6	24.4	215.1	221.5	234.9	201.8	368.6	71.3	1459.0	23.2	283.4	7.0	249.3
50	1.0	38.7	490.3	372.3	354.4	508.2	556.3	13.5	438.6	28.1	83.6	2.5	73.6
50	1.4	48.0	694.2	511.5	429.9	775.8	707.5	1.9	229.2	28.3	35.2	1.8	28.7
50	1.8	50.0	725.5	559.4	443.8	841.2	727.6	0.3	25.6	3.2	1.1	0.1	0.8

Table B.1: *Results for the branch-and-cut algorithm, limit on the length of the ring.*

n	Budget	Nodes in Solution	Optimal value	Distance cost	Node revenue	Demand revenue	Root LP-value	Gap (%)	Num. of subp.	Tree depth	Time (seconds) Total	Cut	LP
10	5	4.9	70.0	122.2	110.6	81.6	91.2	30.3	12.6	3.6	0.0	0.0	0.0
10	10	10.0	242.4	285.2	188.1	339.5	242.4	0.0	1.0	0.0	0.0	0.0	0.0
20	5	4.8	44.4	70.5	87.0	27.8	59.9	35.0	18.6	4.4	0.3	0.0	0.2
20	10	9.9	122.5	165.7	163.7	124.5	169.2	38.1	65.6	7.9	1.2	0.0	1.1
20	20	20.0	418.1	377.0	279.9	515.2	419.1	0.2	2.6	0.5	0.1	0.0	0.0
30	5	4.5	36.8	47.5	71.0	13.4	47.4	28.6	17.0	4.2	0.6	0.0	0.5
30	10	9.9	75.0	131.8	139.4	67.4	116.9	55.9	129.4	10.4	5.9	0.1	5.3
30	20	20.0	271.4	270.4	252.2	289.5	328.5	21.1	352.0	14.1	17.4	0.4	15.8
30	30	30.0	530.2	454.6	341.3	643.5	532.1	0.4	5.8	1.4	0.2	0.0	0.1
40	5	4.7	28.9	40.3	61.1	8.1	38.8	34.1	19.4	5.9	1.2	0.0	1.0
40	10	10.0	56.6	106.6	117.5	45.7	92.4	63.4	159.0	10.1	16.0	0.3	14.6
40	20	20.0	189.5	229.6	231.0	188.1	250.4	32.2	1123.8	19.0	139.9	2.0	129.1
40	30	30.0	393.8	349.3	314.8	428.4	441.0	12.0	693.6	23.2	86.1	1.1	80.7
40	40	40.0	608.5	519.8	377.6	750.7	610.1	0.3	9.8	2.8	0.5	0.0	0.4
50	5	4.4	28.7	30.7	53.5	6.0	38.1	32.9	22.2	6.0	1.9	0.0	1.6
50	10	9.8	54.2	84.9	107.1	32.0	84.7	56.3	134.8	10.7	24.4	0.4	22.6
50	20	20.0	157.3	197.6	218.3	136.6	216.3	37.5	947.6	18.9	273.6	3.3	259.2
50	30	30.0	317.8	300.4	312.6	305.6	387.5	21.9	5320.8	26.0	1525.9	16.1	1431.7
50	40	40.0	520.3	406.1	382.7	543.7	571.4	9.8	2257.0	32.6	541.5	4.4	516.7
50	50	50.0	725.5	559.4	443.8	841.2	727.6	0.3	30.2	3.3	1.9	0.0	1.6

Table B.2: *Results for the branch-and-cut algorithm, limit on the number of nodes.*

budgets are considered.

The computation times are highest for instances with a limit on the number of nodes compared with instances with a limit on the length of the ring. Furthermore, note that the main part of the computation time is spent on solving LPs.

The two heuristics are tested for the same instances as the branch-and-cut algorithm. Computation times of the heuristics are insignificant and always less than 1 second for the instances considered. The results are given in Table B.3 and Table B.4.

Columns one to three in the Tables shows the number of nodes, the budget and the optimal solution obtained by the branch-and-cut algorithm. Columns four to eight presents the results for the construction heuristic and columns nine to thirteen present the results for the deconstruction heuristic. For each heuristic, the solution, the gap relative to the optimal solution and the components of the objective are reported.

The construction algorithm performs best if the budget is restrictive whereas the deconstruction algorithm performs best if the budget is ample. Comparing the heuristic solutions obtained with the proven optimal solutions, there are substantial gaps.

B.5.3 Discussion of Results

As noted in the previous section, the main part of the computation time is spent on solving LPs. This has three explanations. 1) The bound obtained is weak, so the number of processed subproblems is rather high. 2) For each subproblem the LP-solver is invoked several times in order to strengthen the bound from the newly added cuts and to be able to generate more cuts. 3) The LPs are of considerable size.

Improving on the first point can be done by identifying new cuts which improve the bound. Constraints (15) are examples of this and such constraints are important for improving the performance in the case where the number of nodes in the ring is limited. Improving on the second point can be done by generating more cuts and more importantly generate the "right" cuts [14]. For each subproblem, the LP is solved, additional cuts are generated, and the LP is resolved and so on until no more violated cuts exist. The right cuts are the cuts that are effective at the end of this process. If it were possible to identify them, only two calls to the LP solver would be necessary for each subproblem. One initial call

n	Budget	Optimal solution value	Construction Heuristic Solution value	Gap (%)	Distance cost	Node revenue	Demand revenue	Deconstruction Heuristic Solution value	Gap (%)	Distance cost	Node revenue	Demand revenue
10	0.2	7.5	3.7	50.2	3.2	4.2	2.7	0.0	100.0	0.0	0.0	0.0
10	0.6	54.6	40.0	26.6	72.5	73.3	39.3	27.0	50.4	52.3	50.7	28.7
10	1.0	113.8	89.5	21.4	130.3	110.1	109.6	85.7	24.6	133.5	107.8	111.4
10	1.4	182.2	173.3	4.9	213.7	155.0	232.0	162.3	10.9	210.4	151.5	221.2
10	1.8	227.9	193.1	15.2	245.4	168.3	270.2	221.7	2.7	262.3	180.0	304.0
20	0.2	32.7	23.4	28.3	34.0	49.0	8.4	12.4	62.2	17.3	24.7	5.0
20	0.6	104.5	69.9	33.1	117.2	112.7	74.4	39.6	62.1	74.0	74.7	38.9
20	1.0	201.0	135.4	32.6	219.0	172.5	181.9	116.2	42.2	205.8	160.5	161.5
20	1.4	361.7	260.3	28.0	320.0	230.8	349.5	339.8	6.1	314.2	256.0	398.0
20	1.8	418.1	363.8	13.0	371.1	267.3	467.6	413.7	1.1	381.4	279.9	515.2
30	0.2	36.4	27.3	25.0	42.5	57.6	12.2	24.4	33.0	31.5	47.9	8.0
30	0.6	120.2	84.1	30.0	148.6	138.0	94.7	22.5	81.3	116.3	87.6	51.2
30	1.0	316.8	188.0	40.6	283.9	220.4	251.5	213.5	32.6	284.2	225.4	272.3
30	1.4	498.3	332.2	33.3	394.3	283.9	442.7	466.4	6.4	390.7	316.0	541.1
30	1.8	530.2	452.6	14.6	474.6	329.7	597.5	523.8	1.2	461.0	341.3	643.5
40	0.2	38.1	23.1	39.3	55.9	64.0	15.0	2.0	94.7	10.5	9.9	2.6
40	0.6	166.6	95.7	42.6	193.4	167.8	121.3	40.3	75.8	130.6	99.3	71.7
40	1.0	387.4	241.6	37.6	323.5	255.2	309.9	272.4	29.7	327.5	260.9	338.9
40	1.4	568.9	392.8	31.0	450.5	316.6	526.6	526.2	7.5	458.7	348.2	636.7
40	1.8	608.5	530.1	12.9	546.8	370.1	706.8	596.9	1.9	531.4	377.6	750.7
50	0.2	50.0	22.6	54.9	63.5	68.8	17.2	14.7	70.5	38.1	44.5	8.3
50	0.6	215.1	119.3	44.6	218.7	196.5	141.4	68.4	68.2	172.6	146.0	95.0
50	1.0	490.3	309.4	36.9	368.3	308.3	369.4	362.0	26.2	366.8	322.0	406.8
50	1.4	694.2	532.7	23.3	516.4	394.5	654.6	647.7	6.7	513.9	420.4	741.2
50	1.8	725.5	624.4	13.9	619.2	435.8	807.8	703.8	3.0	581.1	443.8	841.2

Table B.3: *Results for heuristics, limit on the length of the ring.*

n	Budget	Optimal solution value	Construction Heuristic: Solution value	Gap (%)	Distance cost	Node revenue	Demand revenue	Deconstruction Heuristic: Solution value	Gap (%)	Distance cost	Node revenue	Demand revenue
10	5	70.0	61.6	12.0	115.2	100.0	76.8	53.8	23.1	126.8	104.1	76.5
10	10	242.4	192.6	20.5	260.7	172.2	281.0	240.1	0.9	287.5	188.1	339.5
20	5	44.4	35.2	20.7	71.9	80.2	26.9	20.0	55.0	43.4	49.4	14.0
20	10	122.5	93.1	24.0	193.1	157.9	128.3	87.1	28.9	230.4	180.8	136.7
20	20	418.1	357.6	14.5	376.4	267.5	466.9	413.7	1.1	381.4	279.9	515.2
30	5	36.8	26.0	29.4	54.3	65.8	14.5	1.6	95.6	19.9	18.4	3.1
30	10	75.0	52.2	30.3	126.5	116.5	62.2	34.4	54.1	168.8	140.8	62.4
30	20	271.4	216.9	20.1	305.7	232.6	289.8	246.9	9.0	311.7	267.3	291.2
30	30	530.2	458.7	13.5	468.4	328.6	600.1	523.8	1.2	461.0	341.3	643.5
40	5	28.9	22.0	24.0	42.0	54.8	9.1	8.7	70.0	13.3	19.1	2.8
40	10	56.6	37.6	33.6	99.3	96.4	39.6	13.3	76.4	96.5	82.4	27.4
40	20	189.5	125.9	33.6	236.8	191.8	166.7	144.9	23.5	297.4	249.1	193.2
40	30	393.8	314.1	20.2	407.9	297.5	417.8	362.0	8.1	385.1	318.3	428.8
40	40	608.5	522.9	14.1	566.3	370.6	714.5	596.9	1.9	531.4	377.6	750.7
50	5	28.7	19.7	31.4	50.1	62.5	7.3	2.6	91.0	19.5	19.6	2.5
50	10	54.2	35.4	34.6	86.9	93.5	28.9	5.6	89.7	75.2	65.1	15.7
50	20	157.3	110.4	29.8	220.9	199.7	131.5	93.5	40.6	281.3	238.0	136.8
50	30	317.8	240.1	24.5	339.1	279.0	300.2	270.7	14.8	363.7	325.5	308.9
50	40	520.3	427.2	17.9	479.0	365.4	540.8	489.7	5.9	446.4	390.0	546.1
50	50	725.5	632.6	12.8	612.7	436.5	809.4	703.8	3.0	581.1	443.8	841.2

Table B.4: *Results for heuristics, limit on the number of nodes.*

and one call after the problem has been resolved. In general it is not possible to identify the right cuts without actually carrying out the iterations. However, heuristics may perform better than simply computing the most violated cuts or even all violated cuts. In particular the heuristic separation routine used to generate GSECs is an example of a routine which generates better cuts than the cuts generated using the optimal separation routine.

Finally at least two possible ways of improving on the third point exists. One possibility is to evaluate the bound or alternative bounds combinatorially. This has been done successfully for the Quadratic Knapsack Problem [5]. The other possibility is to try to reduce the number of constraints in the formulation and the number of non-zero variables in the constraints. One way of doing this could be to remove ineffective cuts and generate them again if needed. It may also be possible to use alternative representations of cuts with fewer non-zero variables as suggested in [14].

As an example of this, let the current LP solution be x_e^*, y_i^*, and z_e^*. Assume that a violated GSEC is given by $S_1 \subset V$, $|S_1| \leq |V|/2$, $k_1 \in S_1$, $l \notin S_1$, and $\sum_{e\in\delta(S_1)} x_e = 0$. Assume further that S contains two nonempty disjoint subsets $S_2 \cup S_3 = S_1$ such that $\sum_{e\in\delta(S_2)} x_e = \sum_{e\in\delta(S_3)} x_e = 0$. Instead of the GSEC corresponding to S_1, one can include two GSECs with S_2, $k_2 \in S_2$, $l \notin S$, and S_3, $k_3 \in S_3$, $l \notin S$. The nodes k_2 and k_3 are picked such that $y_{k_2}^* = \max_{i\in S_2}\{y_i^*\}$ and $y_{k_3}^* = \max_{i\in S_3}\{y_i^*\}$. The GSECs are used in inner form (5) and thus the number of variables for a GSEC constraint is $(|S|^2 - |S|)/2 + |S|$. It can be checked that for $|S_1| \geq 3$ fewer non-zero variables are required to represent the two GSECs corresponding to S_2 and S_3 than the GSEC corresponding to S_1.

We believe that the most promising approach is to obtain new cuts related to the z_e variables which cause the highly fractional solutions. In order for better bounds to be really useful, better heuristics may be needed. In order to determine the maximum speed-up which could be expected if alternative heuristics were used, the optimality verification process has been isolated, i.e. the following has been carried out. Obtain the optimal solution by branch-and-cut and measure the computation time. Rerun the branch-and-cut algorithm but initialize the incumbent to the optimal solution. Doing this, the average reduction in computation time is approximately 40% for the instances which require more than 10 seconds. This is the absolutely best improvement that can be realized due to improved heuristics. However, if the bounds are improved, the computation time may decrease even further.

B.6 Conclusions

The QSTSP has been presented as an extension of the TSP. An integer linear programming model for the QSTSP has been presented. The problem is solved both by applying branch-and-cut and by applying construction heuristics. Computational results are highly dependent on the budget, but in summary the construction heuristics are fast, but obtain solutions far from optimal. The branch-and-cut algorithm computes provably optimal solutions to all test problems with up to 50 nodes within one hour.

In order for the branch-and-cut algorithm to be really useful, it is necessary to reduce the computation time. The main part of the computation time is spent on solving LPs. Hence, to get a substantial overall reduction in computation time, the time spent on solving LPs should be reduced. One way of doing this is to identify better cuts. Alternatively, advanced heuristics could be considered, which would probably obtain much better solutions than the construction heuristics much faster than the branch-and-cut algorithm.

Acknowledgments

The authors would like to thank Jens Clausen, Jesper Larsen and Camilla Schaumburg-Müller for helpful discussions and comments.

Bibliography

[1] COIN-OR: COmputational INfrastructure for Operations Research, `www.coin-or.org`

[2] Balas, E. (1989) The prize collecting traveling salesman problem, *Networks* 19(6), 621-636.

[3] Balas, E. (1989) The prize collecting traveling salesman problem. II. Polyhedral results, *Networks* 25(4), 199-216.

[4] Billionnet, Alain and Calmels, Frederic. (1996) Linear programming for the 0-1 quadratic knapsack problem, *European Journal of Operational Research* 92(2), 310-325.

[5] Caprara, A. and Pisinger, D. and Toth, P. (1999) Exact solution of the quadratic knapsack problem, *INFORMS Journal on Computing* 11(2), 125-137.

[6] Fischetti, M. and Salazar Gonzalez, J.J. and Toth, P. (1995) The symmetric generalized traveling salesman polytope, *Networks* 26(2), 113-123.

[7] M. Fischetti, J.J. Salazar Gonzalez, P. Toth. (1997) A branch-and-cut algorithm for the symmetric generalized traveling salesman problem, *Operations Research* 45(3), 378-394.

[8] Gendreau, M., Labbe, M. and Laporte, G. (1995) Efficient heuristics for the design of ring networks, *Telecommunication Systems - Modeling, Analysis, Design and Management* 4(3-4), 177-188.

[9] Gendreau, M., Laporte, G and Semet. (1998) A branch-and-cut algorithm for the undirected selective traveling salesman problem, *Networks* 32(4), 263-273.

[10] Gouveia, Luis and Manuel Pires, Jose. (2001) Models for a Steiner ring network design problem with revenues, *European Journal of Operational Research* 133(1), 21-31.

[11] Laporte, G. (1986) Generalized subtour elimination constraints and connectivity constraints, *Journal of the Operational Research Society* 37(5), 509-514.

[12] Laporte, G. and Martello, S. (1990) The selective travelling salesman problem, *Discrete Applied Mathematics* 26(2-3), 193-207.

[13] Mak, Vicky and Thomadsen, Tommy. (2004) Facets for the Cardinality Constrained Quadratic Knapsack Problem and the Quadratic Selective Travelling Salesman Problem, *IMM-Technical Report-2004-19.*

[14] Padberg, Manfred and Rinaldi, Giovanni. (1991) A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems, *SIAM Review* 33(1), 60-100.

[15] Thomadsen, Tommy and Stidsen, Thomas. (2005) Hierarchical Ring Network Design Using Branch-and-Price, *To appear in Telecommunication Systems.*

Appendix C

Hierarchical Ring Network Design Using Branch-and-Price

Appears in *Telecommunication Systems*, Volume 29, Issue 1, pp. 61–76, 2005

Hierarchical Ring Network Design Using Branch-and-Price

Tommy Thomadsen[1] and Thomas Stidsen[2]

Abstract

We consider the problem of designing hierarchical two layer ring networks. The top layer consists of a federal-ring which establishes connection between a number of node disjoint metro-rings in a bottom layer. The objective is to minimize the costs of links in the network, taking both the fixed link establishment costs and the link capacity costs into account.

Hierarchical ring network design problems combines the following optimization problems: Clustering, hub selection, metro ring design, federal ring design and routing problems. In this paper a branch-and-price algorithm is presented for jointly solving the clustering problem, the metro ring design problem and the routing problem. Computational results are given for networks with up to 36 nodes.

Keywords: Ring network design, Hierarchical network design, Branch-and-Price.

C.1 Introduction

Design of survivable communication networks is important for at least two reasons. First of all there is a growing reliance on electronic communication in society. Secondly failures (e.g. a link failure) may have a large impact, given the high capacity of links.

Self Healing Rings (or rings for short) have been widely used to ensure survivable communication for several reasons. First of all, the rings are pre-configured such that the only nodes that need to do re-routing in case of a link failure are the two endpoint nodes of the failed link. Thus no communication with other nodes is necessary making ring protection fast. Furthermore the node equipment is

[1]Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark. Email: tt@imm.dtu.dk

[2]Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark. Email: tks@imm.dtu.dk

cheap to build and protection does not require the involvement of an expensive network management system.

Larger networks consist of several interconnected rings, since it is neither possible nor beneficial to restrict the entire network topology to a single ring. One possible way to interconnect the rings is in a hierarchy. Hierarchical networks have existed for decades and were introduced because of the limited switching capabilities in the telephone systems. Hierarchies are still used since they divide the network in sub-networks which can to some extend be treated independently, easing maintenance and upgrade.

In this paper we consider the design of hierarchical ring networks (HRNs), i.e. hierarchical networks where each sub-network is a ring. We assume that communication demands are given and determine a HRN which satisfies the communication demands as cheaply as possible. We present models and algorithms for two layers only, but both models and algorithms can be generalized to more layers. We denote the ring in the top layer the federal-ring, and the node disjoint rings in the bottom layer, the metro-rings. See Figure C.1 for an example of a HRN. We consider single homing, i.e. exactly one node from each metro-ring is in the federal-ring. This node is called the hub node.

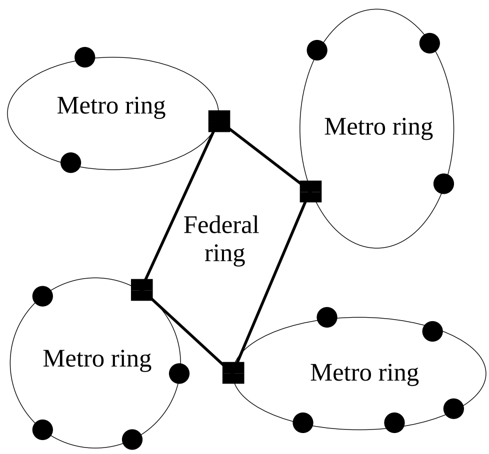


Figure C.1: A two layer hierarchical ring network

The HRN design problem belongs to the more general class of hierarchical network design problems which jointly considers hub location and network design. For an excellent survey of this area we refer to [9]. In [9] the hierarchical network design problem is decomposed into a number of smaller optimization problems:

Clustering: Decide which nodes should belong to the same metro-network.

Hub Selection: For each metro-network select hub nodes to connect the metro-networks to the federal-network.

Metro-Network Design: Determine the best network to connect the nodes in each metro-network.

Federal-Network Design: Determine the best federal-network connecting the hub nodes.

Routing: Route the communication demands, minimizing the capacity usage in nodes and links.

An approach to the hierarchical network design problem is to solve it step by step by solving one or more of the above smaller optimization problems in each step. This is what most papers suggest, including this paper. This means that the hierarchical network design problem is solved by first solving e.g. the clustering problem and the metro-network design problem, then the hub selection problem and the federal-network design problem and finally the routing problem. The optimization process is much simpler but a suboptimal solution of the hierarchical network design problem may be obtained.

The main contribution of the paper is the implementation of a branch-and-price algorithm which can be used to solve to optimality a modified model which includes the clustering problem, the metro-ring design problem and the routing problem. We refer to this problem as *the modified HRN problem*. The hub selection problem and the federal-ring design problem can be solved jointly afterwards and is a Generalized Travelling Salesman Problem considered in e.g. [3]. We discuss the modified HRN problem and point out under what circumstances an optimal solution for the modified problem is optimal for the original optimization problem. The problem modification has previously been put forward and used for implementing heuristics but it has not been analysed in detail. Optimal solutions have previously been obtained for networks with up to 12 nodes and used for comparison with heuristic values. Our branch-and-price algorithm can in general solve instances with 20 nodes and for problems with special structure up to 36 nodes.

The outline of the paper is as follows. In Section C.2, we discuss related papers. In Section C.3 we consider the problem modification of the clustering problem and the metro-ring design problem. In Section C.4, the integer linear formulation of the modified HRN problem is given and in Section C.5 we describe how the integer linear model can be solved using a branch-and-price algorithm. In Section C.6 we give some computational results and suggest some directions for future research. Finally we give some concluding remarks in Section C.7.

C.2 Previous work

Because of the importance of ring protection, a significant amount of work has been carried out regarding the design of ring networks. In this section some related papers are briefly discussed.

In [1] a two layer HRN design problem is studied. The hubs are assumed given, thus eliminating the hub selection problem and reducing the federal-network design problem to a Travelling Salesman Problem. The focus is on the clustering problem and the metro-network design problem. For the optimization of the metro-network design problem, a heuristic with guaranteed worst case performance is suggested. By constructing the rings, the clustering problem is implicitly solved. Finally, the routing problem is ignored, i.e. the capacity costs of the network is not considered.

In [6] a more real-world version of the HRN design problem is studied. Besides requiring some metro-networks to be rings, some metro-networks are allowed to be tree-like. This is achieved by dividing nodes into two groups, some which require ring protection and some which do not. The communication demand pattern assumes that each demand node only requires to communicate with one or a few service nodes. This assumption simplifies the routing problem. The problem is solved in steps using heuristics to first determine the clusters, select hubs, design the metro-networks and design the federal-network. For each problem tailored heuristics are used, which makes the combined optimization algorithm fast and applicable to what-if analyses.

In [8] another type of hierarchical network design problem is considered. Each sub-network is allowed to consist of several rings, which are connected to the same hub nodes. Again the problem is solved in steps. First the clusters are determined and the hubs selected, using the column generation method from [11]. The objective is to maximize the traffic within the metro-networks. Secondly the metro-network design problem is solved for each of the clusters separately by a column generation algorithm. The algorithm is used to minimize the cost of the rings which connects all the nodes in each of the metro-networks to the hubs. The costs includes both link costs and node costs. The federal-network design problem and the routing problem is not considered. Finally the developed algorithm is tested on a real world telecommunication network from Korea Telecom.

Another type of the hierarchical network design problem is suggested in [7]. The focus in this article is the clustering problem. The nodes are clustered in order to minimize the number of necessary clusters, i.e. metro-networks, constrained by the capacity of the metro-rings and the federal-ring. The cost of the metro-ring structures and the federal-ring structures are thus not considered.

The hardness of the routing problem in combination with one or more other problems is illustrated in the paper [5]. Here the routing costs are considered in combination with the logical design of the rings, i.e. which nodes in the rings should contain add-drop equipment. An integer linear program is presented, but as it is pointed out, it is clearly not suitable to exact solution approaches. Instead a tabu search heuristic for the combined problem is presented. The main problem is that given a complete ring-network design, the routing problem is itself a multicommodity optimization problem. Hence, for each iteration in the tabu-search heuristic, many such multicommodity optimization problems needs to be solved. The algorithm is tested on real world examples with up to 48 nodes.

In [14] the modified HRN problem was introduced and this was further developed in [15, 16, 17, 18]. We use the same idea of a modified HRN problem in this paper, and the problem modification is described in greater detail in Section C.3. In the papers [14, 15, 16, 17, 18] both a heuristic and an enumerative scheme is described, but the number of possible networks grows exponentially, making the enumerative scheme useless except for small and trivial instances (less than 10 nodes). The heuristic on the other hand is able to handle large networks, but gives no guarantee regarding the quality of the solutions obtained.

In [12] an integer linear program of the modified HRN problem is presented. Optimal solutions can in some cases be obtained using the model, for networks with up to 12 nodes and a maximum of 4 nodes in the metro-rings. The focus of the paper is a “partition, construct and perturb” heuristic. This heuristic is compared with optimal solutions when these can be obtained and with the heuristics from [14]. It is concluded that better results than Shi and Fonseka are in general obtained.

C.3 The Modified HRN Problem

Let the network $G(V, E)$ where V is the set of nodes and E is the set of possible bidirectional links. Let D be the set of demands, let R^{met} be the set of possible metro-rings and R^{fed} the set of possible federal-rings. For $r \in R^{fed}$ or $r \in R^{met}$, $r \subseteq E$, i.e. r is a subset of links, and the links induce a ring. Let d'_{ij}, $ij \in D$ denote the demand for communication flow between node $i \in V$ and $j \in V$. Also let c_e be the fixed cost for establishing link e and correspondingly let the cost per capacity unit on link e be b_e.

The purpose of modifying the problem is to obtain a formulation which includes the cluster problem, the metro-network design problem and implicitly includes the routing problem. Thus no routing variables are necessary. This is possible, since we consider unidirectional self-healing rings. The modification also allows a decomposition of the total cost into costs for each ring which can be measured independently.

The cost of a HRN is assumed to depend solely on the links used by the rings in the network and the capacity of these links, i.e. the *fixed cost* and the *capacity cost* respectively. Thus the cost of a HRN is as given in equation (1), where $r^{fed} \in R^{fed}$ is the federal-ring, $\overline{R}^{met} \subset R^{met}$ is the set of node disjoint metro-rings covering all nodes and finally CAP_r is the minimal capacity required on each link of ring r to service the traffic flow.

$$\sum_{e \in r^{fed}} c_e + CAP_{r^{fed}} \cdot \sum_{e \in r^{fed}} b_e + \sum_{r \in \overline{R}^{met}} (\sum_{e \in r} c_e + CAP_r \cdot \sum_{e \in r} b_e) \tag{1}$$

The fixed cost of the federal-ring is left as a separate optimization problem, i.e. the HRN cost is initially approximated by the fixed cost of the metro-rings and the capacity cost:

$$\sum_{r \in \overline{R}^{met}} \sum_{e \in r} c_e + CAP_{r^{fed}} \cdot \sum_{e \in r^{fed}} b_e + \sum_{r \in \overline{R}^{met}} CAP_r \cdot \sum_{e \in r} b_e \tag{2}$$

We consider unidirectional self-healing rings, for which it holds that communication flow in the ring takes up capacity in all links in the ring. Thus if a demand $ij \in D$ traverse a ring, it takes up capacity d'_{ij} in all links on the ring. Assume that B is the average capacity cost per ring per unit of demand. An estimate of the capacity cost for satisfying the demand d'_{ij} is Bd'_{ij} if i and j are in the same metro-ring and $3Bd'_{ij}$ if i and j are in different metro-rings, since three rings are in that case traversed (two metro-rings and the federal-ring). Also the capacity cost can be expressed as a worst case cost, $K = 3B \sum_{ij \in D} d'_{ij}$ corresponding to that all demands traverse three rings minus a savings obtained by handling communication demands within metro-rings. Denote by $D_r^{met} \subset D$ the set of demands handled within metro-ring r. In that case the capacity cost can be estimated as follows.

$$CAP_{r^{fed}} \cdot \sum_{e \in r^{fed}} b_e + \sum_{r \in \overline{R}^{met}} CAP_r \cdot \sum_{e \in r} b_e \approx K - 2B \sum_{r \in \overline{R}^{met}} \sum_{ij \in D_r^{met}} d'_{ij} \tag{3}$$

The total HRN cost is then estimated by the following.

$$K + \sum_{r \in \overline{R}^{met}} \sum_{e \in r} c_e - 2B \sum_{r \in \overline{R}^{met}} \sum_{ij \in D_r^{met}} d'_{ij} \tag{4}$$

The intuition behind this rewrite is, that minimizing the capacity cost corresponds to maximizing the communication demand handled within metro-rings.

This is in good agreement with previous recommendations [4] and what has been done in e.g. [11]. Note that $2B$ will have to be experimentally determined. Different values of $2B$ will result in different cost structures, e.g. a low $2B$ will correspond to the case where the capacity cost is higher in the federal-ring than in the metro-rings.

The cost per ring per unit of demand may be far from constant (i.e. deviate considerably from B). However if the capacity cost reflects a cost of node-equipment rather than a cost proportional to the distance between nodes, B is thus proportional to the number of nodes in the rings. In that case it makes much more sense to have a known, fixed B corresponding to a known fixed number of nodes in the rings, and in particular [14, 15, 16] study such networks. For HRNs where the capacity cost per ring per unit of demand is not B in all cases, optimal solutions for the modified problem may not be optimal in the original problem.

Note that the cost can now be decomposed into costs minus a reward for each metro-ring plus a constant K, which can be measured *independently*. Thus the cost for metro-ring $r \in R^{met}$ is:

$$c_r = \sum_{e \in r} c_e - 2B \sum_{ij \in D_r^{met}} d'_{ij} \tag{5}$$

Consider the demand d'_{ij} where i and j are in different metro-rings, i is in the federal-ring and r is the metro-ring including i. In that case equation (4) includes a cost for routing d'_{ij} in r, but d'_{ij} need not be routed "from i via r to i" - there is no need to route it in r at all. Thus additional savings should be included if i is in the federal-ring. This saving is included as a reward on nodes when the federal-ring is designed. The node reward is the sum of all demands starting or ending in the node.

C.4 The Problems

Given the modification of the problem, the idea is now to select the lowest cost set of metro-rings, which includes nodes exactly once, i.e. a set-partitioning problem. However, since there are too many metro-rings to pregenerate all, we generate metro-rings when needed. Thus what we describe is actually a column generation algorithm or, since branching is needed to get integer solutions, an integer programming column generation algorithm, also known as branch-and-price [2, 20].

In this section we will describe the two problems we need to solve; the ring-partitioning problem (which is a set-partitioning problem) and the ring-generation problem. We will describe the branch-and-price algorithm in detail in Section C.5.

When the metro-rings have been designed, the federal-ring is designed as the shortest ring, which includes exactly one node from each metro-ring and takes into account node rewards as described in the previous section. This is a Generalized Travelling Salesman Problem which can be solved using a branch-and-cut algorithm as done in [3]. This problem seems to be easier than the ring-generation problem which is solved many times, and thus the design of the federal-ring is not the bottleneck of the algorithm. We will not consider the design of the federal-ring any further in this paper.

C.4.1 The Ring-Partitioning Problem

Given a set of metro-rings $R \subset R^{met}$, the ring-partitioning problem is the problem of choosing the lowest cost subset of metro-rings in R, such that all nodes are covered exactly once. Define $p_{ir} = 1$ if node i is part of ring r, 0 otherwise. The variables u_r is 1 if ring r is selected, 0 otherwise. The ring-partitioning problem is then:

$$\min \quad \sum_{r \in R} c_r \cdot u_r \tag{6}$$

$$\text{s.t.} \quad \sum_{r \in R} p_{ir} \cdot u_r = 1 \qquad \forall i \in V \qquad (\pi_i) \tag{7}$$

$$u_r \in \{0,1\} \tag{8}$$

The objective (6) is the total cost of selecting metro-rings, where c_r is defined in equation (5). Constraints (7) ensure that each node is in exactly one metro-ring and constraints (8) are the integer domain constraints. Finally π_i are the dual variables for constraints (7). The problem obtained by relaxing constraint (8) is denoted the relaxed ring-partitioning problem. If branching is necessary, additional constraints are added, see Section C.5.1. Rings are iteratively generated and added to R. The ring-generation problem is described in the following section.

C.4.2 The Ring-Generation Problem

The objective of the ring-generation problem is based on the cost in equation (5). However this cost does not include any information on which other rings are in R, and thus it is possible that a node will never be included in any ring. The idea is to add a reward to the objective, which reflects how difficult a node is to cover in the ring-partitioning problem given the *current* set of rings R. A node is difficult to cover if e.g. a single ring $r \in R$ contains the node and thus r need to be selected regardless of the cost. If a node i is difficult to cover a high reward is put on including i in a ring. The reward used is the value of the dual variables in the optimal solution to the ring-partitioning problem, π_i.

Let $d_{ij} = B \cdot d'_{ij}$, let $n(r) \subseteq V$ be the nodes in r and let $D_r \subset D$ be the set of demands which start *and* end in r. Formally, we generate the ring with most negative reduced cost, where the reduced cost is given by the following equation.

$$c_r - \sum_{i \in n(r)} \pi_i = \sum_{e \in r} c_e - \sum_{ij \in D_r} d_{ij} - \sum_{i \in n(r)} \pi_i \tag{9}$$

We assume an upper limit, m is given on the number of nodes in the ring. Define the following variables, $y_i = 1$ if node i is in the ring, 0 otherwise, $x_e = 1$ if link e is in the ring, 0 otherwise and $z_{ij} = 1$ if demand ij can be handled by the ring, otherwise 0. (Equivalently, $z_{ij} = 1$ if $y_i = 1$ and $y_j = 1$, otherwise 0.)

For $S \subset V$, let $\delta(S) \subset E$ denote the set of edges with an endpoint in S and an endpoint not in S. Then the ring-generation problem can be stated as follows.

$$\min \quad \sum_{e \in E} c_e \cdot x_e - \sum_{ij \in D} d_{ij} \cdot z_{ij} - \sum_{i \in V} \pi_i \cdot y_i \tag{10}$$

$$\text{s.t.} \quad \sum_{e \in \delta(\{i\})} x_e = 2y_i \qquad \forall i \in V \tag{11}$$

$$z_{ij} \leq y_i \qquad \forall ij \in D \tag{12}$$

$$z_{ij} \leq y_j \qquad \forall ij \in D \tag{13}$$

$$z_{ij} \geq y_i + y_j - 1 \qquad \forall ij \in D \tag{14}$$

$$\sum_{i \in V} y_i \leq m \tag{15}$$

$$\sum_{e \in \delta(S)} x_e \geq 2(y_k + y_l - 1)$$

$$\forall S \subset V, 3 \leq |S| \leq n - 3, k \in S, l \notin S \tag{16}$$

$$x_e \in \{0,1\}, y_i \in \{0,1\}, z_{ij} \in \{0,1\} \tag{17}$$

The objective (10) corresponds exactly to the reduced cost given in equation (9). If a node is selected ($y_i = 1$), two links should be incident to node i, which is ensured by constraint (11). If both nodes i and j are selected the variable $z_{ij} = 1$, which is ensured by the constraints (12), (13) and (14). The number of nodes in the rings is bounded by the hop constraint (15). Subtour elimination constraints (16) ensure that a single ring is generated and finally integer solutions are ensured by the domain constraints (17).

We solve the ring-generation problem by branch-and-cut as described in [19], where the subtour elimination constraints are generated as needed. Also [10] describes cuts which may improve the performance of the branch-and-cut algorithm. The ring-generation problem is a generalization of the (Selective) Travelling Salesman Problem and of the Quadratic Knapsack problem and thus we denote it the Quadratic Selective Travelling Salesman Problem.

If branching is necessary, additional terms are added to the objective function and additional constraints are added. These additions are described in Section C.5.1.

C.5 The Branch-and-Price Algorithm

The branch-and-price algorithm is described in pseudo code in Figure C.2. The main idea in a branch-and-price algorithm is to perform the bounding in a branch-and-bound algorithm using column generation. The algorithm maintains an incumbent, i.e. the lowest cost feasible solution known, and a set of branch-nodes, i.e. a set of relaxed ring-partitioning problems. Initially the set of branch-nodes contains the ring-partitioning problem without any branching decisions. A branch-node corresponding to a relaxed ring-partitioning problem is solved using column generation in the inner while loop. It is resolved in each iteration of the inner while loop and a ring is generated by the ring-generation problem. If no ring exists with negative reduced cost the value of the ring-partitioning problem is a lower bound. This lower bound is used in the outer loop which is the branch-and-bound part of the algorithm.

In the outer loop it is checked whether the optimal solution to the relaxed ring-partitioning problem solution is feasible, i.e. integer, or if it is a lower bound only. If the solution is integer and better than the current incumbent, the incumbent is updated and that branch is fathomed. If the solution is fractional, the lower bound is compared with the current incumbent and if it is worse, the branch is fathomed. If neither is the case, branching is performed.

```
INCUMBENT = Infinity.
BRANCH-NODES = {Initial Relaxed Ring-Partitioning problem}
while BRANCH-NODES ≠ ∅ do
   Select branch B ∈ BRANCH-NODES
   do
      Solve relaxed ring-partitioning problem B
      Solve ring-generation problem, based on dual variables of B
      if Reduced cost of optimal ring < 0 then
         Add optimal ring to B
   while Reduced cost of optimal ring < 0
   Let OBJ_VAL = Optimum of B
   if the solution to the relaxed ring-partitioning problem is
         feasible (integer) and OBJ_VAL ¡ INCUMBENT then
      Update incumbent: INCUMBENT = OBJ_VAL
      Fathom branch
   else if OBJ_VAL ≥ INCUMBENT then
      Fathom branch
   else
      Branch: Add two branches to BRANCH-NODES
end while
```

Figure C.2: *The Branch-and-Price algorithm*

C.5.1 Ryan-Foster Branching

Branching in a branch-and-price algorithm is more complicated than in a standard branch-and-bound algorithm. We use Ryan-Foster branching [13] to obtain integer solutions. This is possible since all coefficients of all constraints in the ring-partitioning problem are 0 or 1 and all right hand sides are 1, see constraint (7).

Consider constraint i. Since the right hand side is 1 and variables have to be integer, exactly one ring with $p_{ir} = 1$ has to be selected ($u_r = 1$). For all other selected rings, $p_{ir} = 0$. We say that "node i is covered by ring r". The idea is now to identify a set of rings $S \subset R^{met}$ and create two branches, 1) node i has to be covered by a ring in S and 2) node i has to be covered by a ring not in S. The question is now, how do we select i and S.

Assume node i is partially covered by more than one ring, and assume ring r is one of these rings (i.e. $0 < u_r < 1$). Usual variable branching corresponds to letting $S = \{r\}$, thus the branches will be $u_r = 1$ and $u_r = 0$. This sort of branching is not suitable in a column generation algorithm for several reasons all related to the vast amount of variables that exists (but are not explicitly known). First of all since we set $u_r = 0$ in the ring-partitioning problem, r usually has a negative reduced cost and hence when solving the ring-generation problem, r will be generated *again*. This can be handled by modifying the ring-generation problem to specifically exclude r. However, usually rings similar to r exists and thus these rings will be generated instead. This means that the bound of the $u_r = 0$ branch will not improve much when branching and we have an unbalanced branch-tree where the depth is considerable.

The idea is to let S contain several rings and in particular include rings which *have not yet been generated* (i.e. not in R). Thus in general $S \setminus R \neq \emptyset$. Identify a fractional ring ($0 < u_r < 1$) and two nodes i and j with $p_{ir} = 1$ and $p_{jr} = 1$. If the solution is fractional, such two nodes always exists. Let $S = \{r \in R^{met} | p_{ir} = 1 \wedge p_{jr} = 1\}$, that is the rings that cover both i and j. The two branches are thus, 1) i and j are covered by the same ring and 2) i and j are covered by different rings.

A branch decision is identified by a node-pair $\{i, j\}$ and whether i and j should be covered by the same ring or not. For a ring-partitioning problem, we have several such branch decisions of both types. Denote by $B^{SAME} \subset V^2$ the set of branching decisions where node-pairs should be covered by the same ring and correspondingly denote by $B^{DIFF} \subset V^2$ the set of branching decisions where node-pairs should be covered by different rings. Then we add the following constraints to the ring-partitioning problem which implement the actual branching.

$$\sum_{\{r \in R | p_{ir}=1 \wedge p_{jr}=1\}} u_r = 1 \qquad \forall \{i, j\} \in B^{SAME} \qquad (\gamma_{ij}) \tag{18}$$

$$\sum_{\{r \in R | p_{ir}=1 \wedge p_{jr}=1\}} u_r = 0 \qquad \forall \{i, j\} \in B^{DIFF} \qquad (\delta_{ij}) \tag{19}$$

We denote the dual variables of the constraints by $\gamma_{\{i,j\}}$ and $\delta_{\{i,j\}}$ as indicated. The constraints added to the ring-partitioning problem affect the calculation of the reduced costs of rings, thus the objective of the ring-generation problem is changed. Note that $p_{ir} = 1 \wedge p_{jr} = 1$ exactly if $z_{ij} = 1$ in the ring-generation problem. Let $\gamma_{\{i,j\}} = 0$ if $\{i, j\} \notin B^{SAME}$ and $\delta_{\{i,j\}} = 0$ if $\{i, j\} \notin B^{DIFF}$, then the objective of the ring-generation problem (see equation (10)) becomes:

$$\sum_{e \in E} c_e \cdot x_e - \sum_{ij \in D} (d_{ij} + \gamma_{\{i,j\}} + \delta_{\{i,j\}}) \cdot z_{ij} - \sum_{i \in V} \pi_i \cdot y_i \tag{20}$$

When solving the ring-generation problem, it is furthermore necessary to ensure that only rings which fulfill the branching decisions are generated. This is ensured by the following constraints.

$$y_i - y_j = 0 \qquad \forall \{ij\} \in B^{SAME} \tag{21}$$

$$y_i + y_j \quad \leq \quad 1 \qquad \forall \, \{ij\} \in B^{DIFF} \tag{22}$$

Both constraints allows rings where both $y_i = 0$ and $y_j = 0$, but constraints (21) ensure that if node i is selected, then so is j and vice versa. On the other hand, constraints (22) ensure that rings generated include at most one of i and j.

C.6 Computational Results

To test the branch-and-price algorithm, problem instances with between 10 and 20 nodes are generated. The problem instances are generated similarly to what is done in [19]. The nodes are placed in a plane with the coordinates uniformly distributed between 0 and 100. The fixed costs (c_e) are determined as the Euclidean distance. Rather than generating both capacity costs (b_e) and the demands (d'_{ij}) and compute an average cost per ring per unit of demand to obtain d_{ij} (as discussed in Section C.3), we generate d_{ij} only. The d_{ij} values are generated as uniformly distributed between 0 and an upper bound u.

Selecting a proper value of u is critical. If u is selected too small, then the optimal solution is a single federal-ring including all nodes and no metro-rings. Using the same value of u as in [19] proved sufficient. The upper bound u used is given in the following equation.

$$u \approx \frac{5}{\sqrt{|V|^3}} \tag{23}$$

The value of u arise by considering the tradeoff between total average demand and the shortest tour measured in fixed link costs for rings with $|V|/2$ nodes. We refer to [19] for an in-depth explanation. The important observation is, that a tradeoff exists between the fixed link cost and the savings obtained from demands. As we shall see, the hop constraint (15) is in most, but not all cases binding; thus a tradeoff exists. The tests were run on a 1200 Mhz SUN Fire 3800. We use CPLEX 9.0 to solve linear programming models.

For each of 10, 12, 14, 16, 18 and 20 nodes, 10 different random instances are generated. We report results as averages over 10 instances. We vary the maximal number of nodes in the metro-rings, m between 4 and $\min\{10, |V|-3\}$. In addition to this, we investigate networks with 25 and 36 nodes with m equal to 5 and 6 respectively. It turns out, that since $|V|/m$ is integer for these networks, they are easier to solve than networks for which this is not the case. The results are given in Table C.1. The table shows the number of nodes, the maximum number of nodes in metro-rings, the number of branch-nodes, the total time spent in seconds and the percentage spent on the ring-partitioning

|V|	m	#Branch Nodes	Total Time (sec.)	Time Part.	Time Gene.	#Rings Gene.	#Metro Rings
10	4	8.0	4.0	6.7%	93.3%	40.2	3.0
10	5	1.0	2.5	4.6%	95.4%	18.8	2.0
10	6	11.2	7.8	4.7%	95.3%	64.4	2.0
10	7	7.2	4.9	5.7%	94.3%	42.1	2.0
12	4	4.0	3.7	4.9%	95.1%	23.0	3.0
12	5	13.0	16.4	3.9%	96.1%	83.2	3.0
12	6	1.0	7.3	3.1%	96.9%	30.1	2.0
12	7	15.0	24.5	3.5%	96.5%	105.5	2.0
12	8	19.2	27.6	3.7%	96.3%	123.4	2.0
12	9	9.8	14.1	3.6%	96.4%	65.4	2.0
14	4	3.2	5.7	4.7%	95.3%	27.2	4.0
14	5	6.4	18.5	2.6%	97.4%	51.1	3.0
14	6	44.4	105.1	2.7%	97.3%	277.3	3.0
14	7	1.0	29.8	1.4%	98.6%	46.8	2.0
14	8	32.6	110.2	2.9%	97.1%	275.8	2.0
14	9	50.6	118.5	3.2%	96.8%	327.5	2.0
14	10	36.4	85.6	3.4%	96.6%	251.5	2.0
16	4	5.8	11.9	4.2%	95.8%	39.6	4.4
16	5	29.4	87.9	2.6%	97.4%	176.3	4.0
16	6	34.4	158.4	2.1%	97.9%	251.1	3.0
16	7	68.8	359.3	2.3%	97.7%	539.8	3.0
16	8	1.8	84.2	0.9%	99.1%	68.7	2.0
16	9	44.2	324.9	1.9%	98.1%	405.2	2.0
16	10	60.6	383.0	2.5%	97.5%	570.7	2.0
18	4	16.4	32.3	3.6%	96.4%	74.2	5.0
18	5	2.6	32.4	1.8%	98.2%	40.3	4.0
18	6	6.0	89.5	1.2%	98.8%	75.5	3.2
18	7	33.4	446.5	1.3%	98.7%	325.6	3.0
18	8	124.0	1183.7	2.1%	97.9%	1116.0	3.0
18	9	1.0	217.3	0.6%	99.4%	89.9	2.0
18	10	30.2	737.2	1.3%	98.7%	440.2	2.0
20	4	6.8	27.0	3.9%	96.1%	50.6	5.7
20	5	8.6	91.8	1.5%	98.5%	71.7	4.7
20	6	24.2	356.4	1.1%	98.9%	190.3	4.0
20	7	12.8	407.0	0.7%	99.3%	143.4	3.2
20	8	53.4	1854.6	0.9%	99.1%	659.6	3.0
20	9	179.8	4344.2	1.4%	98.6%	2026.2	3.0
20	10	1.0	688.0	0.3%	99.7%	117.8	2.0
25	5	11.2	302.0	1.0%	99.0%	110.8	5.9
36	6	21.0	5457.8	0.4%	99.6%	245.9	7.0

Table C.1: Computational Results. Averages over 10 instances.

problem and the ring-generation problem respectively. Finally the number of times that metro-rings are generated (this includes cases where no metro-rings are actually found) and the number of metro-rings in the optimal solution are listed.

For all problem instances with up to 20 nodes, the branch-and-price algorithm terminates in at most 3 hours (average worst case is 73 minutes). Since the design of HRNs are considered strategic problems, the computational time is acceptable. As it can be seen, the bottleneck in the algorithm is the generation of rings which consistently takes more than 90% of the running time. The gradually increasing running time for increasing $|V|$ may both be attributed to increased running time for each ring-generation problem solved and to the increasing number of metro-rings which are generated (second last column). The number of branch-nodes is limited, making memory issues negligible. However, each branch requires generation of a substantial number of metro-rings, causing substantially higher running time.

Instances where $|V|/m$ is integer are easier than instances where this is not the case. This is due to the increased number of branch-nodes which is caused by an increased amount of fractional variables. Especially when $|V|/m = 2$, the possibility of obtaining an integer solution without branching is high. The special case when all metro-rings and the federal-ring have the same number of nodes, i.e. $|V|/m = m$ and $|V|/m$ integer, is considered in [14, 15, 16]. Since $|V|/m$ is integer, as discussed above, such instances are easier to solve to optimality than instances where this is not the case. The last two rows in Table C.1 gives results for instances with $|V| = 25$, $m = 5$ and $|V| = 36$, $m = 6$. The most difficult instances with 36 nodes are solved in less than 6 hours and on average over 10 instances in just above $1\frac{1}{2}$ hour.

For networks with up to 20 nodes, in most cases, the optimal solution contains exactly the minimum number of metro-rings needed, given m. Only in 22 cases out of 380 test runs in total, one more than the minimum number of metro-rings needed is in the optimal solution. In Table C.1, this is the reason why the last column contains fractions. This indicates that the demand values are sufficiently high to make the metro-rings profitable and the hop constraint (15) thus binding. On the other hand, since some instances exists for which this is not the case, the demand values are not too high.

C.6.1 Future Research

The approach described in this paper can handle instances of the modified HRN problem with up to 20 nodes. While we would like to solve larger problems, the discussion in Section C.1 should make clear that the modified HRN problem constitutes a hard optimization problem. Furthermore, previous HRN approaches have either considered heuristics, with no performance guarantees, or enumerative schemes which can handle less than 12 nodes. In order to be able to handle larger instances in reasonable time, it is paramount to reduce the time spent on ring-generation. Note that for each branch-node in the branch-and-bound algorithm, at least one ring-generation problem has to be solved to optimality (the one giving no rings) to ensure that the value obtained when solving the ring-partitioning problem is indeed a bound. Thus it is inevitable that the ring-generation problem has to be solved to optimality at least as many times as there are branch-nodes. The remaining number of times that rings are generated heuristics could be used, rings could be pre-generated and several rings could be generated each time. These techniques could probably increase the size of the modified HRN problems which can be handled. In this paper we have further considered the most abstract and general formulation of the problem. An obvious practical improvement could be to limit the links allowed in the network to the e.g. k nearest neighbours for each node. Another approach to reduce the problem hardness is the use of a so-called compatibility graph which disallow certain pairs of nodes to be in the same metro-network is suggested in [11].

As mentioned in Section C.3, the optimal solution of the modified HRN problem may not be optimal in the original HRN problem. This is mainly for two reasons: The metro-rings and the federal-ring is designed in separate (thus non-optimal) stages and secondly, the modification assumes the average cost per ring per unit of demand of rings are the same. It seems possible but nontrivial to include the federal-ring design in the branch-and-price algorithm, but it seems more difficult to solve the problem with the cost per ring per unit of demand. However, one initial approach to take is to investigate how much the optimal solution for the modified HRN problem deviates from the optimal solution to the original HRN problem. This could be done either by investigating very small instances for which optimal solutions can be found or by finding a lower bound on the original problem cost.

Also it would be interesting to allow bidirectional instead of unidirectional self-healing rings. One possibility is to use the same problem modification, and thus approximate the bidirectional rings with unidirectional rings. However, in that case the traffic approximation becomes even more unreliable.

Note that a capacity constraint can be added to the ring-generation problem, thus dealing with ring capacity for the metro-rings. If capacities are available in modular sizes for varying capacity costs, an idea is to generate rings for each of the available capacities and corresponding capacity cost. This will probably not work very well, essentially because the capacity cost per ring per unit of demand is different for rings and thus B is not a good estimate for at least some rings. It may be possible to get reasonable results, however, by using different estimates of B for the various capacities. Assume a low capacity ring has a high per unit capacity cost. In that case, intuitively a high B should be used for low capacity rings, since this will correspond to a low overall saving accounting for the more expensive per unit capacity, see equation (4).

C.7 Conclusion

In this paper we have considered the problem of designing HRNs. A problem modification has been presented which has previously been used to build heuristics for designing HRNs. A branch-and-price algorithm is described, implemented and tested. For the modified problem this algorithm finds provably optimal solutions to networks with up to 20 nodes in less than 3 hours. For problems with special structure, the algorithm finds provably optimal solutions with up to 36 nodes in less than 6 hours. The computational time depends heavily on the instances considered, and in particular it is possible to design considerably larger networks if the maximum number of nodes in metro-rings are small and/or if the number of nodes in the network is divisible with the maximum number of nodes in metro-rings. Algorithmic improvements which could speed up the algorithm have been suggested and we also suggest an investigation of how much the optimal solution to the modified problem deviates from the solution to the original problem. In particular this investigation is important if bidirectional self healing rings are considered.

Bibliography

[1] K. Altinkemer. (1994) Topological design of ring networks, *Computers & Operations Research* 21(4), 421-431.

[2] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh and P.H. Vance. (1998) Branch-and-price: column generation for solving huge integer programs, *Operations Research* 46(3), 316–29.

[3] M. Fischetti, J.J. Salazar Gonzalez, P. Toth. (1997) A branch-and-cut algorithm for the symmetric generalized traveling salesman problem, *Operations Research* 45(3), 378-394.

[4] T. Flanagan. (1990) Fiber network survivability, *IEEE Communications Magazine* 28(6), 46-53.

[5] B. Fortz, P. Soriano, C. Wynants. (2003) A tabu search algorithm for self-healing ring network design, *European Journal of Operational Research* 151(2), 280-295.

[6] M. Gawande, J.G. Klincewicz and H. Luss. (2000) Design of SONET/SDH ring assignment with capacity constraints, *Advances in performance analysis* 2, 159-217.

[7] O. Goldschmidt, A. Laugier and E.V. Olinick. (2003) SONET/SDH ring assignment with capacity constraints, *Discrete Applied Mathematics* 129(1), 99-128.

[8] D. Kang, K. Lee, S. Park, K. Park and S.-B. Kim. (2000) Design of local networks using USHRs, *Telecommunication Systems* 14(4), 197-217.

[9] J. G. Klincewicz. (1998) Hub location in backbone/tributary network design: a review, *Location Science* 6, 307-33.

[10] V. Mak and T. Thomadsen. (2004) Facets for the Cardinality Constrained Quadratic Knapsack Problem and the Quadratic Selective Travelling Salesman Problem, *IMM-Technical Report-2004-19.*

[11] K. Park, K. Lee, S. Park and H. Lee. (2000) Telecommunication node clustering with node compatibility and network survivability requirements, *Management Science* 46(3), 363-374.

[12] A. Proestaki and M.C. Sinclair. (2000) Design and dimensioning of dual-homing hierarchical multi-ring networks, *IEE Proceedings-Communications* 147(2), 96-104.

[13] D.M. Ryan and B. Foster. (1981) An integer programming approach to scheduling, *Computer Scheduling of Public Transport. Urban Passenger Vehicle and Crew Scheduling. Proceedings of an International Workshop*, 269-280.

[14] J. Shi and J.P. Fonseka. (1993) Dimensioning of self-healing rings and their interconnections, *Global Telecommunications Conference, 1993, including a Communications Theory Mini-Conference. Technical Program Conference Record, IEEE in Houston. GLOBECOM '93., IEEE.* 3, 1579-1583.

[15] J. Shi and J.P. Fonseka. (1994) Design of hierarchical self-healing ring networks, *Communications, 1994. ICC '94, SUPERCOMM/ICC '94, Conference Record, 'Serving Humanity Through Communications.' IEEE International Conference on.* 1, 478-482.

[16] J. Shi and J.P. Fonseka. (1995) Hierarchical self-healing rings, *IEEE/ACM Transactions on Networking*, 690-697.

[17] J. Shi and J.P. Fonseka. (1996) Interconnection of self-healing rings, *1995 IEEE International Conference on Communications. Converging Technologies for Tomorrow's Applications. ICC '96.* 3, 1563-1567.

[18] J. Shi and J.P. Fonseka. (1997) Analysis and design of survivable telecommunications networks, *IEE Proceedings-Communications* 144(5), 322-330.

[19] T. Thomadsen and T. Stidsen. (2003) The Quadratic Selective Travelling Salesman Problem, *IMM-Technical Report-2003-17.*

[20] F. Vanderbeck and L.A. Wolsey. (1996) An exact algorithm for IP column generation, *Operations Research Letters* 19, 151-159.

Appendix D

Joint Routing and Protection Using p-cycles

Submitted for *European Journal of Operational Research*

Joint Routing and Protection Using p-cycles

Thomas Stidsen[1] and Tommy Thomadsen [2]

Abstract

Today people rely heavily on electronic communication systems like the Internet, telephone systems, etc. Hence, it is important to ensure reliable electronic communication. The bulk of the electronic communication today is carried by circuit switched networks, thus protection against failures in these networks is paramount. Protection is possible by rerouting the electronic communication, bypassing the failed network component. In order to be able to reroute, extra capacity is, nevertheless, needed.

This article considers the recently suggested fast protection method, p-cycles. We develop a method for minimizing the capacity needed for protection using p-cycles. The routing of traffic influence the amount of extra capacity needed, thus we consider joint optimization of routing and protection.

An integer linear programming model is presented and a column generation algorithm is developed. The algorithm is faster and obtains better bounds and solutions than existing methods. The algorithm enables an experimental study of the capacity efficiency of p-cycles. The results show that p-cycles are comparable to any other protection method, with respect to the capacity usage. The results also show that substantial capacity savings can be achieved when routing and protection is performed jointly.

Based on the integer linear programming model, we discuss how protection costs can be taken into account in routing methods. We also discuss an alternative efficiency measure of the p-cycles, which takes into account the interaction with existing p-cycles.

Keywords: networks, p-cycles, routing, protection, column-generation.

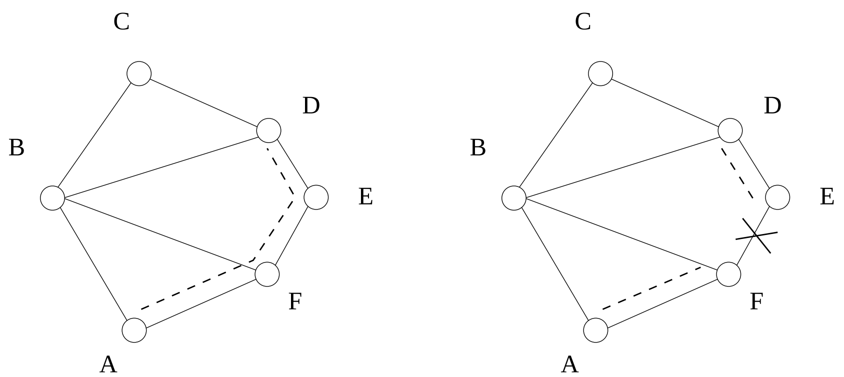
[1] Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark. Email: tks@imm.dtu.dk

[2] Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark. Email: tt@imm.dtu.dk

D.1 Introduction

Reliable communication networks are important in society today because of the increasing dependency on electronic communication. However, most communication networks are vulnerable to equipment failures, cable cuts, electric outages, etc. Furthermore, is difficult, if possible at all, to avoid such failures. Alternatively, the traffic may be reestablished by rerouting the traffic around the failed network components.

Most of the high capacity communication networks today are circuit switched, i.e. a connection is established prior to sending actual data. In this article we consider bidirectional connections which enables two way communication via bidirectional links. In Figure D.1(a), a bidirectional connection is established between node A and node D. Given a network of nodes and bidirectional links and a communication demand defined by a set of bidirectional connections, routing is the optimization task of deciding the paths which should be used by the bidirectional connections. The path of the connection in Figure D.1(a) use the links AF, FE, and ED. Furthermore, the connection occupies a certain bandwidth on the links AF, FE, and ED of its path. The required capacity of a link is the sum of the bandwidths of all connections which pass the link. The *working capacity* of the network is the summed capacity of all the links.



(a) A bidirectional connection

(b) A broken bidirectional connection

Figure D.1: A circuit switched network

In Figure D.1(b), the effect of a link failure of link FE is shown: The link between E and F fails and the connection between A and D is lost. When a link failure occurs, the connections may be reestablished by rerouting the connections passing the failed link. Rerouting is applied in protection methods to recover failures. A number of different protection methods has been suggested: Span protection, path protection, ring protection, global rerouting [16, 19], and p-cycle protection. For a comprehensible review of the different protection techniques we refer to [6]. Generally these protection methods protect against any single link failure. We consider cycle protection [13] as an abstract model of ring protection.

In order to be able to reroute in link failure situations, capacity is needed in addition to the working capacity. Determining what additional capacity should be installed in order to be able to handle any single link failure is an optimization task. The additional capacity necessary is denoted the *protection capacity*.

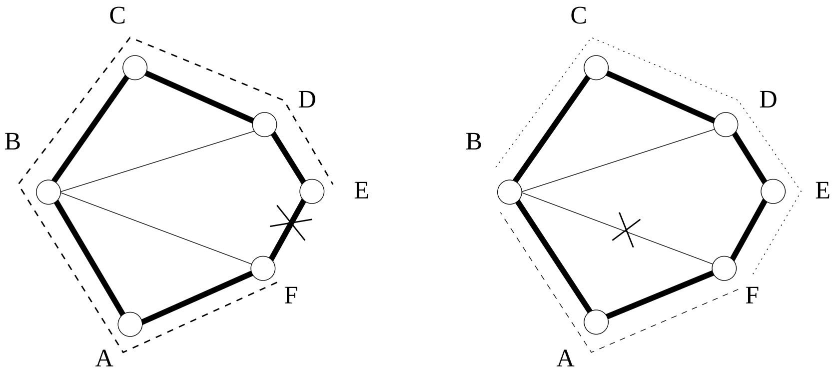
Recently the so-called p-cycle protection method (Pre-configured Protection Cycle), has been suggested [8]. The authors claim that the p-cycle protection method is both capacity efficient and offer fast protection, leading to the claim that p-cycles provide "ring-like speed with mesh-like capacity".

Routing is usually done prior to protecting the network. For p-cycles, it is, however, beneficial to optimize routing and protection jointly. This article considers the joint routing and protection problem, where protection is performed using p-cycles. The main contribution is the development of a column generation algorithm which determines close to optimal solutions for the joint routing and protection problem.

The remainder of the article is organized as follows. In Section D.2, the general problem of routing and p-cycle protection is described. Previous work on optimization of p-cycle protection is briefly reviewed in Section D.3. In Section D.4, the column generation algorithm for joint routing and p-cycle allocation is described. This algorithm is tested on six networks and in Section D.5, the results are presented and discussed. Finally concluding remarks are given in Section D.6.

D.2 The p-cycle Protection Method

The p-cycle protection method uses additional capacity allocated in cycles to protect the links. The allocated cycles are denoted p-cycles. The same amount of capacity is required on all links of the p-cycle. The capacity is pre-configured such that in case of a link failure, the only nodes that need to do rerouting are the end nodes of the failed link. Thus no signaling is required. The p-cycle protects two types of links, *on-cycle* links, see Figure D.2(a) and *straddling* (chord) links, see Figure D.2(b). In the figures, the thick solid lines indicate the pre-configured capacity of the p-cycle. The failed links, in Figure D.2(a) link EF and in Figure D.2(b) link BF, are marked with a cross.



(a) On-cycle link protection (b) Straddling link protection

Figure D.2: p-cycle protection

On-cycle protection uses the fact that there is always one other way around the cycle, in case of a link failure. In case a link on the cycle fails, the connections on that link are rerouted over the remaining links of the cycle. The rerouted connections are illustrated with the dashed line from node E through D C B A to node F. Naturally the maximal number of connections which can be protected for the on-cycle links corresponds to the pre-configured capacity of the p-cycle.

Figure D.2(b) shows how a straddling link can be protected. Because the end-nodes are on the cycle but the *link* is not on the cycle, the cycle has two routes between the end-nodes (B and F) of the failed link, illustrated with the dashed line and the dotted line in Figure D.2(b). The p-cycle can hence protect twice the pre-configured capacity of the p-cycle. For a more comprehensive description of p-cycles we refer to chapter 10 in [6].

A link may be protected by several p-cycles, i.e. if a link fails, the connections using that link may be protected by rerouting them along several different p-cycles protecting that link.

This article studies the problem of jointly planning routing and p-cycle protection. Given a network and a connection demand, the sum of the working capacity and the protection capacity of the network is minimized.

D.3 Previous Work on p-cycle Planning

p-cycles were first suggested in [8] and it was claimed that p-cycles provide "ring-like speed with mesh-like capacity". Since then, a number of articles have been published regarding different aspects of p-cycles. In this section we briefly review those which are most relevant in connection with the joint routing and p-cycle protection problem considered here.

In [15] theoretical arguments are given for the efficiency of p-cycles. Bounding-type arguments are given for the claim that the p-cycle method is the most capacity efficient pre-configured protection method. They do, however, base the argumentation on fully connected networks. This seems far from the rather sparsely connected telecommunication networks.

A Mixed Integer Program (MIP) model for planning p-cycle protection is presented in [8]. A prerouted network is assumed, i.e. the protection capacity requirement is minimized given the working capacity. One problem with the MIP model is that it requires enumeration of all possible p-cycles. Furthermore, it may be capacity inefficient since routing and protection are performed separately. Because the number of p-cycles grows exponentially, only networks of moderate size may be solved to optimality. By pre-selecting "promising" p-cycles, the size of the networks which can be handled may be increased albeit sacrificing the optimality guarantee. This is considered in [4, 9]. In [4] two measures for evaluating p-cycles are suggested. In Section D.4.4.2 we study these measures in more detail. In [9], the effect of preselecting p-cycles of different size is investigated.

The problem of *joint* routing and p-cycle protection is studied in [7, 12]. In [7] a number of paths and p-cycles are pre-selected, making optimization of networks of medium size possible, again sacrificing the optimality guarantee. In [12], column generation is applied to implicitly represent all paths and p-cycles. The column generation subproblem is, however, not solved to optimality and thus no bounds can be derived, see Section D.4.4.1. Still, it is in [12] demonstrated that low capacity requirements can be achieved using p-cycle protection. In [13], the related problem of joint routing and protection using cycles is studied.

A different approach is taken in [14]. Here a complex MIP model, which does *not* require enumeration of all possible p-cycles, is formulated. The number of binary variables of the formulation is $O(|N| \cdot |L| \cdot |C|)$, where $|N|$ is the number of nodes, $|L|$ is the number of nodes and $|C|$ is the number of different p-cycles which are actually used. While this is certainly an improvement compared to an exponential number of variables in the MIP formulation from [8], the size of the formulation still grows significantly making optimal solution methods intractable for networks of medium size. Instead, an elaborate method for stepwise optimization of gradually refined models is suggested. The approach is verified by application to full meshed networks with up to 25 nodes.

D.4 Solution Methodology

As discussed in Section D.3, the MIP model suggested in [8] requires enumeration of all possible p-cycles to achieve an optimal solution. In this section, we describe how the use of a column generation algorithm allows us to solve a relaxation of the MIP model through *implicit* enumeration of the p-cycles. This enables solution of the LP-relaxed MIP model to optimality generating only a fraction of the possible p-cycles.

Section D.4.1 describes the MIP model for joint routing and p-cycle protection. In Section D.4.2 the column generation algorithm which is needed to solve the relaxed MIP model is described. The column generation algorithm requires the solution of two sub-problems: The path generation problem described in Section D.4.3, and the p-cycle generation problem described in Section D.4.4. Finally, Section D.4.5 describes how to use the generated paths and p-cycles to find near optimal solutions to the original MIP model.

D.4.1 The Joint Routing and Protection Planning Problem

Consider a network consisting of a set of nodes N and a set of links L. Furthermore, a set of connection demands D, indexed by unordered node pairs, $k, l \in V$ are defined. The constant $d_{kl} \in N_0$ is the number of connections demanded between nodes k and l. A set of paths P_{kl}, exist for each demand kl and a set of p-cycles R are given. Let c_{ij} be the capacity cost for allocating one unit of capacity on link $ij \in L$. The capacity cost of a path $p \in P_{kl}$ is $c_p^{kl} = \sum_{ij \in p} c_{ij}$. The capacity cost of a p-cycle $r \in R$ is $c_r = \sum_{ij \in r} c_{ij}$, i.e. the sum of the capacity cost of the *on-cycle* links of the p-cycle. We assume that the capacity unit of the required connections d_{kl} is equal to the capacity units of the links.

The constants $PATH_{p,ij}^{kl}$ have value 1 if path $p \in P_{kl}$ use link $ij \in L$ and 0 otherwise. The constants $PCYC_{r,ij}$ have value 1 if link ij of p-cycle r is on-cycle, 2 if link ij is straddling, and 0 otherwise. The $PCYC_{r,ij}$ constants define the protection offered by the p-cycle.

The variables $v_p^{kl} \in Z^+$ are the number of connections of demand $kl \in D$ that use path $p \in P_{kl}$ and the variables $u_r \in Z^+$, are the pre-configured capacity of p-cycle $r \in R$. Then a MIP model for the Joint routing and p-cycle protection problem, henceforth called the JP model can be formulated:

minimize:

$$\overbrace{\sum_{r \in R} c_r \cdot u_r}^{protection\ cost} + \overbrace{\sum_{kl \in D} \sum_{p \in P_{kl}} c_p^{kl} \cdot v_p^{kl}}^{routing\ cost} \tag{1}$$

subject to:

$$(\xi_{kl}) \qquad \sum_{p \in P_{kl}} v_p^{kl} = d_{kl} \quad \forall\, kl \in D \tag{2}$$

$$(\pi_{ij}) \qquad \sum_{r \in R} PCYC_{r,ij} \cdot u_r - \sum_{kl \in D} \sum_{p \in P_{kl}} PATH_{p,ij}^{kl} \cdot v_p^{kl} \geq 0 \quad \forall\, ij \in L \tag{3}$$

$$v_p^{kl} \in Z^+ \qquad \forall\, p \in P_{kl} \tag{4}$$

$$u_r \in Z^+ \qquad \forall\, r \in R \tag{5}$$

The objective function (1) calculates the combined routing and protection cost. The constraints (2) ensure that all demands are satisfied by routing exactly the required connections along one or more of the available paths. The constraints (3) ensure that each link is protected against failure by allocation of enough protection capacity along p-cycles which offers protection to the link. Notice the difference between on-cycle link protection and straddling link protection is included in the $PCYC_{r,ij}$ constant. The dual variables of constraints (2) are ξ_{kl} and the dual variables of constraints (3) are π_{ij}.

The JP model is a generalization of the MIP model suggested in [8], which arises when P_{kl} contain exactly one path, the shortest, for each demand kl. The same model as the above is used in [7, 12]. The main problem with the JP model is that the number of paths and p-cycles grows exponentially with the number of nodes (and links) in the network. In order to ensure optimality, all paths and p-cycles must be considered explicitly or implicitly. To avoid explicit representation of paths and p-cycles, column generation is applied. The Relaxed JP model, R-JP is created by relaxing the integer domain constraints (4) and (5) of the variables v_p^{kl} and u_r, i.e. $v_p^{kl}, u_r \in R^+$. The R-JP model is an LP model, which can then be solved using column generation where the paths and p-cycles are taken into account implicitly. The column generation algorithm solves a R-JP model of reduced size where only a small subset of paths P and p-cycles R are included. We denote this the R-JP(P,R) model. Paths and p-cycles are then generated when needed.

D.4.2 Column Generation Algorithm

The idea of a column generation algorithm is to only generate the variables when needed, i.e. when the *reduced cost* of a variable is negative. For each iteration of the column generation algorithm the paths (one for each demand) with the minimal reduced cost is found and the p-cycle with the minimal reduced cost is found. If the reduced cost of a path or a p-cycle is negative, they are called *improving*. If no improving paths or p-cycles are found, the algorithm terminates and the R-JP model has been solved to optimality using only a subset of possible paths and p-cycles. The column generation algorithm is given in pseudo-code in Figure D.3.

P = Shortest path for each demand node pair kl
R = one dummy p-cycle for each link ij
do
    Solve the R-JP(P,R) problem
    Solve routing subproblems searching for improving paths
    if improving paths found **then**
        Add improving paths to P
    Solve p-cycle subproblem searching for an improving p-cycle
    if improving p-cycle found **then**
        Add improving p-cycle to R
while improving path or improving p-cycle is found

Figure D.3: *The joint routing and p-cycle protection column generation algorithm*

Initially the column generation algorithm is started with a set of shortest paths, one for each demand, and a set of *dummy* p-cycles one for each link ij. A dummy p-cycle is a (non-existent) p-cycle which has the ability of protecting just one link and which is so expensive that it will never show up in the optimal solution. Then the R-JP(P,R) model is solved based on the current set of paths P and the current set of p-cycles R. Based on the dual variables from equation (2) (ξ_{kl}) and equation (3) (π_{ij}), improving paths and p-cycles are found. This process continues until no improving paths or p-cycles are found.

D.4.3 Subproblem I: Path Generation

The path generation problem is the problem of generating paths with negative reduced cost. The reduced cost of a variable $\hat{c}_p^{kl}$ can be calculated based on the dual variables ξ_{kl} and π_{ij} from equation (2) and equation (3) in the R-JP(P,R) model and the link cost c_{ij} as follows.

$$\hat{c}_p^{kl} = \sum_{ij \in p} c_{ij} - \xi_{kl} + \sum_{ij \in p} \pi_{ij} \tag{6}$$

Each reduced cost contains three terms, the sum of the link costs c_{ij}, a reward term ξ_{kl} for providing an additional path to route the demand kl and a sum of the link protection costs π_{ij}. The term ξ_{kl} appear in the reduced cost for all kl-paths. Therefore the path with the lowest reduced cost for a demand kl can be found as the shortest path in a network with link costs defined as follows.

$$\overline{c}_{ij} = c_{ij} + \pi_{ij} \tag{7}$$

By duality $\pi_{ij} \geq 0$ and by assumption $c_{ij} \geq 0$, this means that $\overline{c}_{ij} \geq 0$. Thus we can apply the Floyd-Warshall algorithm [3] and the shortest paths for all demands kl can be calculated in $O(|N|^3)$. The running time may be improved using iterated Dijkstra, but since running time for generating paths is insignificant, this has not been implemented.

For all node pairs kl, if a path exists with $\hat{c}_p^{kl} < 0$ it is an improving path and it is included into the set of paths P in the R-JP(P,R) model.

Column generation is a standard approach used for the multi commodity flow problem, hence the pricing problem of paths has been extensively studied. The problem is studied in connection with network restoration in [11] and in connection with p-cycles in [12].

When the column generation algorithm terminates, the price $\overline{c}_{ij}$ is the price for using that link, including both routing costs *and* protection costs. Often joint routing and p-cycle protection is unrealistic because, as is argued in [14], introduction of new p-cycles in a network is a strategic decision, whereas routing is an operational decision. We suggest that after the strategic choice of p-cycles, based on a *forecast* of the demand, routing is performed as shortest path routing based on $\overline{c}_{ij}$ prices. This is not optimal because new p-cycles may be needed, but an estimation of the protection costs is utilized in the routing.

D.4.4 Subproblem II: p-cycle Generation

The second subproblem is the p-cycle generation problem, i.e. the problem of generating p-cycles with negative reduced costs. The reduced costs of the p-cycles depend only on the π_{ij} dual values. The reduced costs may be calculated as given below.

$$\begin{aligned}\widehat{c}_r = \sum_{ij\in r} c_{ij} - \sum_{ij\ straddling\ r} 2\cdot\pi_{ij} - \sum_{ij\in r}\pi_{ij} \\ = \sum_{ij\in r} c_{ij} - \sum_{ij\ straddling\ r\ or\ ij\in r} 2\cdot\pi_{ij} + \sum_{ij\in r}\pi_{ij} \qquad (8)\end{aligned}$$

The last equality sign follows immediately by including both straddling and on-cycle links into the second sum and afterwards correcting by adding the third sum.

The p-cycle generation problem is an NP-hard optimization problem [12, 17] which we have previously termed the Quadratic Selective Travelling Salesman problem. This problem is described in detail in [17] and we refer to this article for an in-depth treatment. In [10] a polyhedral study of the problem is carried out.

The following MIP model of the P-Cycle Generation problem, henceforth called the PCG model, uses three types of variables: The variables $y_i \in \{0,1\}$ represent the nodes which are part of the p-cycle, 1 for being included 0 otherwise. The variables $x_{ij} \in \{0,1\}$ for $ij \in L$ represents the links where the protection capacity is pre-configured, 1 for being included 0 otherwise. Finally the variables $z_{ij} \in R^+$ represents all node pairs $ij \in L$ which are included in the p-cycle, 1 for the node pair being included 0 otherwise. The PCG is then expressed as follows.

minimize:

$$\sum_{ij\in L}(c_{ij}+\pi_{ij})x_{ij} - \sum_{ij\in L} 2\pi_{ij}z_{ij} \qquad (9)$$

subject to:

$$\sum_{j \in V} x_{ij} = 2y_i \qquad \forall\ i \in N \quad (10)$$

$$z_{ij} \leq y_i \qquad \forall\ ij \in L \quad (11)$$

$$z_{ij} \leq y_j \qquad \forall\ ij \in L \quad (12)$$

$$z_{ij} \geq y_i + y_j - 1 \qquad \forall\ ij \in L \quad (13)$$

$$\sum_{i \in S, j \notin S, ij \in L} x_{ij} \geq 2(y_k + y_l - 1)$$

$$\forall\ S \subset N, 3 \leq |S| \leq |N| - 3\ , k \in S, l \notin S \quad (14)$$

$$x_{ij}, y_i \in \{0,1\} \quad z_{ij} \in R^+ \quad (15)$$

The objective equation (9) calculates the reduced cost as described above in equation (8). All nodes which are in the p-cycle are required to have to two incident links, which is ensured by equation (10). For each $ij \in L$ where i and j are included in a p-cycle, i.e. $y_i = 1$ and $y_j = 1$, the variable $z_{ij} = 1$. This is ensured by equation (11), (12) and (13). Since $\pi_{ij} \geq 0$, constraints (13) are implied and are thus not necessary in the formulation. Sub-tour elimination constraints are added in order to ensure that connected cycles are constructed (14). Finally the domain constraints (15) ensure integer values for the x and y variables which in turn force integrality of the z variables.

The PCG model is solved using the branch-and-cut algorithm described in [17]. Solving the PCG model is *the* bottleneck of the column generation algorithm. This is validated by computational tests, see Table D.2 in Section D.5.1. However, the total computation time is acceptable, thus we have deemed improvements unnecessary. If a speed up of the column generation algorithm is needed, heuristic generation of improving p-cycles could be applied. Inspiration for this could be sought in algorithms for the TSP problem and pricing problems for cycles [12]. However, to ensure optimality of the column generation algorithm, guarantee of non-negative reduced costs are required, thus, ultimately optimal solution of the PCG model is required.

D.4.4.1 Reduced cost of cycles

For comparison we modify the algorithm to deal with the Joint routing and Cycle protection (JC) model. We use the same column generation algorithm as for the JP model. The only difference is the removal of straddling protection from the $PCYC_{r,ij}$ constant, i.e. $PCYC_{r,ij} = 1$ if link ij of p-cycle $r \in R$ is on-cycle and 0 otherwise. The path generation problem, see Section D.4.3, remains the same, but the MIP model for the cycle generation problem is slightly different from the PCG model. The objective is to find the cycle with the most negative reduced cost, hence equation (9) is changed to equation (16) below, where the reward for the straddling links have been removed.

$$\text{minimize} \quad \sum_{ij \in L} (c_{ij} - \pi_{ij}) x_{ij} \tag{16}$$

Only the objective function is changed to find improving cycle instead of improving p-cycles. However, at the same time the z_{ij} variables and the constraints in equation (11), (12) and (13) become obsolete. This indicates that cycle generation is easier than p-cycle generation, and in fact cycles can be generated in polynomial time. Applying the Bellman-Ford algorithm [3], cycles with negative reduced costs, negative cycles, can be found in $O(|L| \cdot |N|^2)$ time.

D.4.4.2 p-cycle Efficiency

As mentioned in Section D.3, one way to reduce the problem of the large number of possible p-cycles is to pre-select a fraction of promising p-cycles. In [4] two different measures of the p-cycles efficiency for p-cycle pre-selection is suggested: “A Priori p-cycle Efficiency” $AE(r)$, see equation (17); and “Demand-weighted p-cycle Efficiency” $EW(r)$, see equation (18).

$$AE(r) = \frac{\sum_{ij} PCYC_{r,ij}}{\sum_{ij \in r} c_{ij}} \tag{17}$$

$$EW(r) = \frac{\sum_{ij} CAP_{ij} \cdot PCYC_{r,ij}}{\sum_{ij \in r} c_{ij}} \tag{18}$$

The efficiency measure $AE(r)$ counts the number of protected links, divided by the cost of the p-cycle. In $EW(r)$ the offered protection capacity is weighted with the working capacity which needs to be protected for each link, CAP_{ij}. Hence this measure assumes that the demands are already routed.

To compare $AE(r)$ and $EW(r)$ measures with the reduced cost ($\widehat{c}_r$) from equation (8), the reduced costs of the p-cycles is divided by the cost $\sum_{ij \in r} c_{ij}$ (assuming $\sum_{ij \in r} c_{ij} \neq 0$) of the p-cycle and equation (19) is obtained.

$$\widehat{c}'_r = 1 - \frac{\sum_{ij \in r} \pi_{ij} \cdot PCYC_{r,ij}}{\sum_{ij \in r} c_{ij}} \tag{19}$$

Given a p-cycle r, the sign of $\widehat{c}'_r$ is the same as $\widehat{c}_r$, because we assume $c_{ij} \geq 0$, i.e. $\widehat{c}_r < 0 \Rightarrow \widehat{c}'_r < 0$. However, the division may have changed the order of the p-cycles with negative reduced costs, hence the best p-cycle according to equation (8) is not necessarily the best p-cycle according to equation (19). The division effectively makes the shorter p-cycles more attractive. If we ignore the constant term and change the sign of the fraction, we obtain a new measure which should be maximized.

$$\widehat{c}''_r = \frac{\sum_{ij} \pi_{ij} \cdot PCYC_{r,ij}}{\sum_{ij \in r} c_{ij}} \tag{20}$$

It is interesting to compare the optimal p-cycles according to the three different measures: $AE(r)$ (equation (17)), $EW(r)$ (equation (18)) and $\widehat{c}''_r$ (equation (20)). The optimal p-cycle according to the $AE(r)$ measure, is the p-cycle with the lowest average cost for link protection. The main problem is that it does not take into account the actual need for protection, i.e. the working capacity which needs to be protected. The $EW(r)$ measure weighs the importance the protection of the links according to the working capacity CAP_{ij} of each link. The main problem with the $EW(r)$ measure is that it does not take the interplay of the different p-cycles into account, i.e. a link may not be very interesting to protect, even though CAP_{ij} is high, because the link might already be cheaply covered by other efficient p-cycles. Given a network, a set of existing p-cycles and a demand, we conjecture that the measures defined in equation (8) and equation (20) are the best measures of future p-cycles to include into the network. Furthermore, these measures seems most appropriate when choosing p-cycles to add in response to increased demand.

D.4.5 Getting Integer Solutions

The column generation algorithm obtains an optimal solution to the R-JP model, but it is *not* guaranteed to return an integer solution, i.e. the optimal solution to the JP model. In this article we have chosen the simple solution of solving the JP model using a standard MIP solver with the paths P and p-cycles R collected during the column generation algorithm. Hence it is really the JP(P,R) model which is solved and it is important to acknowledge that the MIP solver only returns the optimal solution given the available paths and p-cycles and *not* the optimal integer solution to the full JP model. But because we have an optimal lower bound from the column generation algorithm, the solution to the R-JP model, we can quantify the worst case optimality gap. As the results clearly illustrates in Section D.5.3 this approach is fully sufficient to achieve close to optimal performance for all the networks tested. In order to get the real optimal solution a branch-and-price algorithm is needed [1, 18].

D.5 Results and Discussion

Our column generation algorithm and the integer heuristic is tested on six networks, see Table D.1. The objective of the tests and discussions in this section is twofold: To examine the efficiency of the column generation algorithm and to compare the capabilities of p-cycles with cycle protection and the global rerouting lower bound [16, 19]. The global rerouting lower bound is achieved by allowing rerouting of all connections in case of any single link failure. Note that global rerouting is a lower bound for any protection method. In [19] a heuristic for global rerouting is suggested but here we report results obtained by a column generation algorithm which guarantees the lower bound [16].

	Nodes	Links	Avg. Node Degree	Working Capacity	Global Rerouting Abs.	Rel.
Cost239 [2]	11	26	4.73	86	11.6	13 %
Europe	13	21	3.23	158	90.0	57 %
USA [4]	28	45	3.21	1273	641.2	50 %
Italy [5]	33	68	4.12	1718	581.4	34 %
France [4]	43	71	3.3	3473	1604.0	46 %
France 2 [4]	43	71	3.3	4043	3156.3	78 %

Table D.1: The tested networks

The columns in Table D.1 contain (in order): The number of nodes, the number of links, the average node degree, the working capacity i.e. capacity after shortest path routing of all demands, and the global rerouting lower bound both in absolute extra capacity and the percentage extra compared to working capacity.

For all networks, one connection is requested for all node pairs, except for the network France 2 where the same (sparse) demand pattern as in [4] was used. Otherwise, the networks France and France 2 are identical. In the tests we assume unit costs for the links, i.e. $c_{ij} = 1$, as have been done previously in [4].

Based on the test networks, Section D.5.1 presents results regarding computational efficiency of the algorithm. Section D.5.2 compares the protection capacity of p-cycles with the protection capacity of cycles, using shortest path routing and joint routing, and with the global rerouting lower bound. In Section D.5.3, the integer solutions are compared with the bound. Finally, in Section D.5.4, the importance of straddling link protection offered by the p-cycle method is studied.

D.5.1 Computational Efficiency

Table D.2 presents data regarding the running time of the column generation algorithm. For each network, the total running time, the percentage of the time spent on solving the R-JP problem (including time spent on initialization and path generation), the percentage of the time spent on generating p-cycles, the percentage of the time spent on obtaining integer solutions, the number of p-cycles generated (Gen.), the number of p-cycles used in the integer solutions (Used), and the time spent on generating one p-cycle on average. The CPLEX 9.0 solver is used both to solve the R-JP model in the column generation algorithm, to solve the linear programs in the branch-and-cut algorithm for the PCG model, and to obtain the integer solutions of the JP(P,R) model. The MIP solver generates the integer solutions as described in Section D.4.5, but if a provably optimal solution is not found after 30 seconds, the MIP solver is terminated and the best feasible solution found is returned. Preliminary tests show that 30 seconds was sufficient to obtain good heuristic solutions. The computer used was a 1200 MHz SUN Fire 3800 machine.

	Total Time (sec.)	JP Time (%)	PCG Time (%)	Integer Time (%)	#p-cycles Gen.	Used	Avg. PCG Time (sec.)
Cost239	0.4	25.0 %	75.0 %	0.0 %	8	3	0.04
Europe	1.0	10.0 %	90.0 %	0.0 %	10	5	0.09
USA	44.3	2.0 %	30.2 %	67.7 %	17	11	0.79
Italy	160.6	3.1 %	78.1 %	18.7 %	44	14	2.85
France	360.1	2.1 %	96.3 %	1.6 %	43	22	8.07
France 2	239.2	0.7 %	99.1 %	0.2 %	41	19	5.78

Table D.2: Computational efficiency

The column generation algorithm terminates in less than 361 seconds for all the test networks. The main part of the running time is spent on generating p-cycles, and to some extend finding an integer solution. The number of generated p-cycles is low, always less than 50, even though the number of possible p-cycles for example in the France network is at least 500000 [4]. Furthermore only about half of these are used in the integer solutions. The running time may be improved by pre-generating a number of p-cycles e.g. by using the pre-selection methods suggested in [4, 9].

D.5.2 Protection Capacity Efficiency

In this section, we compare the (integer) solutions for the JP model, the Shortest path routing p-cycle protection (SP) model, the JC model and the Shortest path routing Cycle protection (SC) model. The SP protection method and the SC protection method only allows the demands to be satisfied using one path: The shortest. Hence, the JP model is reduced to contain only the shortest path in the set of paths P_{kl} for each demand. As mentioned earlier in Section D.3, other articles have considered p-cycle protection assuming a prerouted demand. Thus the comparison between JP and SP is interesting. The cycle protection models are solved using the same column generation algorithm described in Section D.4.2, with the modifications described in Section D.4.4.1.

For each network in Table D.3, the working capacity is given in the first column. The protection capacity in absolute number and relative to the working capacity is given for the integer solutions for the four different models.

		p-cycle Protection				Cycle Protection			
	Working	JP		SP		JC		SC	
	Abs.	Abs.	Rel.	Abs.	Rel.	Abs.	Rel.	Abs.	Rel.
Cost239	86	30	35%	37	43%	90	105%	94	109%
Europe	158	112	71%	147	93%	162	103%	182	115%
USA	1273	861	68%	1064	84%	1328	104%	1472	116%
Italy	1718	868	51%	1206	70%	1774	103%	1892	110%
France	3473	2255	65%	2904	84%	3732	107%	3954	114%
France 2	4043	3345	83%	3470	86%	4774	118%	4848	120%

Table D.3: p-cycle and cycle protection efficiency

From Table D.3 it is clear that the joint routing and p-cycle protection method is the most efficient. This method requires between 35% and 83% protection capacity to protect the network. The corresponding cycle protection method requires between 103% and 118% protection capacity. p-cycles are most capacity efficient for the networks with the highest node degree: Cost239 and Italy. The higher density of the networks enables better use of straddling links, which is shown in Section D.5.4.

In [14] a number of good arguments against joint routing and protection are given. While we acknowledge these, we find it interesting that savings of 3% –

22% of the required protection capacity is possible for p-cycles. A possible explanation of the improved efficiency of joint routing and protection is offered in Section D.5.4. For cycle protection, the total capacity savings are, however, only 2% – 12%.

In [4], it is suggested to solve the SP model using p-cycles generated prior to optimization. Results are presented for USA and France 2. For USA, all p-cycles can be enumerated, thus the optimal solution of 1064 is obtained, which coincide with the solution we have obtained. For France 2, all p-cycles cannot be generated, but by generating 15000, a heuristic solution of 3675 is obtained. For comparison, we obtain a heuristic solution of 3470 using the SP model and a solution of 3345 if joint optimization is applied.

It could be argued that the comparison in Table D.3 is not fair, since the percentages are given compared to no protection at all. In Table D.4, the global rerouting lower bound [16, 19] is compared to p-cycle protection with and without joint routing. The first column contains the working capacity and the second column the additional capacity needed for global rerouting protection. Then follows the protection capacity, the extra capacity compared to the global rerouting lower bound, and the extra capacity in percent of the working capacity for JP and SP.

	Working	Global	JP			SP		
		Rerouting	Abs.	Extra	Extra %	Abs.	Extra	Extra %
Cost239	86	11.6	30	18.4	21 %	37	25.4	30%
Europe	158	90.0	112	22.0	14 %	147	57.0	36%
USA	1273	641.2	861	219.8	17 %	1064	422.8	33%
Italy	1718	581.4	868	286.6	17 %	1206	624.6	36%
France	3473	1604	2255	651	19 %	2904	1300	37%
France 2	4043	3156.3	3345	188.7	5 %	3470	313.7	7.8%

Table D.4: Global rerouting vs. p-cycle protection

It is interesting to observe, that at most 21% extra capacity is needed to ensure protection using p-cycles as compared to the global rerouting lower bound. Furthermore, France 2 is only 5% from the global rerouting lower bound. This is due to the sparse demand pattern.

D.5.3 Integer Solution Quality

Table D.5 shows the gap between the lower bound found by the column generation algorithm and the integer solution found by the MIP solver.

	p-cycle Protection		Cycle Protection	
	JP	SP	JC	SC
Cost239	2.33 %	4.65 %	5.45 %	2.50 %
Europe	0.00 %	0.59 %	0.75 %	0.99 %
USA	0.35 %	0.15 %	0.09 %	0.00 %
Italy	0.81 %	0.11 %	0.08 %	0.14 %
France	0.15 %	0.01 %	0.00 %	0.00 %
France 2	0.08 %	0.08 %	0.04 %	0.03 %

Table D.5: Integer gap (%) to Column generation lower bound

As can be seen from Table D.5, the solutions obtained are within 1% from optimum for all variants of the algorithms for all networks, except Cost239 where the integer solutions are up to 5.45% from the lower bound. Thus in general the algorithm produces close to optimal solutions.

The Cost239 network is small, thus all p-cycles can be generated. Furthermore, the optimal integer solution can be obtained, however the gap is still substantial. Thus, the heuristic solution obtained is good, but the lower bound for Cost239 is significantly worse than for the other networks. This may be caused by the high density of the Cost239 network compared to the other networks.

D.5.4 Straddling Link Protection and Surplus Capacity

The difference between p-cycles and cycles is in essence the possibility of protecting straddling links. In this section we investigate how much of the protection capacity is straddling protection compared to on-cycle protection. The p-cycles may be able to protect more working capacity in the link than is actually present. This is denoted *surplus capacity*. The surplus capacity for a link ij corresponds to the value of the left hand side of inequality (3) and the total surplus capacity is the sum of surplus capacity for all links.

Table D.6 compare the on-cycle protection capacity, the straddling protection capacity and the surplus capacity for JP and SP. All capacities are given in percent of the total protection capacity, i.e. on-cycle plus straddling protection capacity. It can be seen in Table D.6 that for the networks Cost239 and Italy, more than 50% of the protection capacity is straddling protection. For the rest of the networks 40% to 50% is straddling protection. The higher amount of straddling capacity in the networks Cost239 and Italy is due to the higher density of these networks. This also explains the higher capacity efficiency of

	JP			SP		
	On-cycle	Straddling	Surplus	On-cycle	Straddling	Surplus
Cost239	29 %	71 %	17 %	31 %	69 %	29 %
Europe	52 %	48 %	25 %	52 %	48 %	44 %
USA	52 %	48 %	20 %	56 %	44 %	33 %
Italy	39 %	61 %	18 %	43 %	57 %	39 %
France	50 %	50 %	17 %	58 %	42 %	30 %
France 2	56 %	44 %	30 %	55 %	45 %	36 %

Table D.6: Pre-configured capacity: On-cycle, straddling and surplus

the p-cycle protection method for these networks in Table D.3.

The effect of joint routing and protection only slightly increases the amount of straddling capacity. On the other hand, joint routing and protection significantly decreases the amount of surplus capacity and this seems to be the main reason for the improved capacity efficiency of joint routing and protection.

D.6 Conclusion

In this article we have described an integer linear programming model for the problem of jointly routing and protecting a network using p-cycles. A column generation algorithm is implemented to obtain lower bounds. Based on the columns generated, heuristic solutions are found. The gap between the lower bound and the heuristic solution is insignificant.

An experimental study shows that the algorithm obtains better bounds and solutions faster than previously used algorithms. Lower bounds and solutions are found for networks with up to 43 nodes and 71 links in at most six minutes.

The experiments further show that straddling link protection is a valuable addition to cycle protection. Also, joint routing and protection reduce the total capacity usage compared to when routing is predetermined. The gap between the joint routing and protection for p-cycles and the global rerouting lower bound is only $5\% - 21\%$, which is quite remarkable since the p-cycle protection method is fast.

Based on the integer linear programming model, it is discussed how the protection cost can be taken into account in routing methods. Finally, a new measure of p-cycle efficiency is discussed, which takes the interplay of existing p-cycles into account.

Bibliography

[1] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh and P.H. Vance. Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46(3):316–29, 1998.

[2] P. Batchelor, B. Daino, P. Heinzmann, D.E. Hjelme, P. Leuthold, R. Inkret, G.D. Marchis, H.A. Jager, F. Matera, M. Joindot, B. Mikac, A. Kuchar, H.-P. Nolting, E.L. Coquil, J. Spath, F. Tillerot, B.V. Caenegem, N. Wauters, and C. Weinert. Study on the implementation of optical transparent transport networks in the european environment-results of the Research Project COST 239. *Photonic Network Communication* 2(1):15–32, 2000.

[3] T.H. Cormen, C.E. Leiserson and R.L. Rivest. *Introduction to Algorithms.* McGraw-Hill Book Company, 2th edition, 2001.

[4] J. Doucette, D. He, W. Grover, and O. Yang. Algorithmic approaches for efficient enumeration of candidate p-cycles and capacitated p-cycle network design. *Fourth International Workshop on the Design of Reliable Communication Networks*, 212–220, 2003.

[5] W. Grover, J. Doucette, M. Clouqueur, D. Leung, and D. Stamatelakis. New options and insights for survivable transport networks. *IEEE Communications Magazine*, 40(1):34–41, 2002.

[6] W.D. Grover. *Mesh-Based Survivable Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking*, Prentice Hall, 2003.

[7] W.D. Grover and J. Doucette. Advances in optical network design with p-cycles: joint optimization and pre-selection of candidate p-cycles. *IEEE/LEOS Summer Topical Meeting*, WA2–49, 2002.

[8] W.D. Grover and D. Stamatelakis. Cycle-oriented distributed preconfiguration: ring-like speed with mesh-like capacity for self-planning network restoration. *IEEE International Conference on Communications*, 1:537–543, 1998.

[9] J. Kang and M. Reed. Bandwidth protection in MPLS networks using p-cycle structure. Proceedings of the Fourth International Workshop on the Design of Reliable Communication Networks, 356–362, 2003.

[10] V. Mak and T. Thomadsen. Facets for the Cardinality Constrained Quadratic Knapsack Problem and the Quadratic Selective Travelling Salesman Problem. *IMM-Technical Report-2004-19*, 2004.

[11] K. Murakami, and H.S. Kim. Joint optimization of capacity and flow assignment for self-healing ATM networks. *IEEE International Conference on Communications*, 1:216–220, 1995.

[12] D. Rajan, and A. Atamtürk. Survivable network design: routing of flows and slacks. In Anandalingam, G and Raghavan, S., editors, *Telecommunications Network Design and Management*, Kluwer Academic Publishers, 65–81, 2003.

[13] D. Rajan, and A. Atamtürk. A directed cycle-based column-and-cut generation method for capacitated survivable network design. *Networks*, 43(4):201–211, 2004.

[14] D.A. Schupke. An ILP for optimal p-cycle selection without cycle enumeration. *Eighth Working Conference on Optical Network Design and Modelling (ONDM)*, 2004.

[15] D. Stamatelakis and W.D. Grover. Theoretical underpinnings for the efficiency of restorable networks using preconfigured cycles (p-cycles). *Communications, IEEE Transactions on* 48(8):1262–1265, 2000.

[16] T. Stidsen. P. Kjærulff. Protection lower bounding through global rerouting. *Work In Progress*.

[17] T. Thomadsen and T. Stidsen. (2003) A Branch-and-Cut Algorithm for the Quadratic Selective Travelling Salesman Problem. *Submitted for Telecommunication Systems*.

[18] F. Vanderbeck and L.A. Wolsey. An exact algorithm for IP column generation. *Operations Research Letters* 19(4):151–159, 1996.

[19] J. Yamada. A spare capacity design method for restorable networks. *IEEE Global Telecommunications Conference* 2:931–935, 1995.

Appendix E

The Generalized Fixed-Charge Network Design Problem

Accepted for publication in *Computers and Operations Research*

The Generalized Fixed-Charge Network Design Problem

Tommy Thomadsen[1] and Thomas Stidsen [2]

Abstract

In this paper we present the Generalized Fixed-Charge Network Design (GFCND) problem. The GFCND problem is an instance of the so-called Generalized Network Design problems. In such problems, clusters instead of nodes have to be interconnected by a network. The network interconnecting the clusters is a Fixed-Charge network, and thus the GFCND problem generalizes the Fixed-Charge Network Design problem. The GFCND problem is related to the more general problem of designing hierarchical telecommunication networks.
A mixed integer programming model is described and a branch-cut-and-price algorithm is implemented. Violated constraints and variables with negative reduced costs are found using enumeration. The algorithm is capable of obtaining optimal solutions for problems with up to 30 *clusters and up to* 300 *nodes. This is possible, since the linear programming relaxation bound is very tight and there are few non-zero variables and few binding constraints.*

Keywords: Fixed-Charge Network Design, Generalized Network Design, Branch-Cut-and-Price, Hierarchical Networks

E.1 Introduction

Hierarchical telecommunication networks consist of two or more layers of networks. Hierarchical telecommunication networks exist for historical reasons and enable economy of scale in the central high speed networks, the backbone-networks. This paper examines the problem of designing a backbone mesh network interconnecting given clusters. The problem is an important subproblem when constructing hierarchical telecommunication networks. This problem is denoted the *Generalized Fixed-Charge Network Design* (GFCND) problem and is illustrated in Figure E.1.

[1]Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark. Email: tt@imm.dtu.dk

[2]Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark. Email: tks@imm.dtu.dk

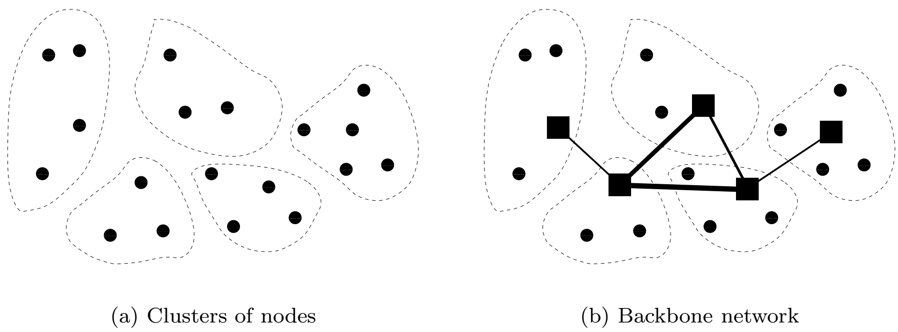


Figure E.1: The Generalized Fixed-Charge Network Design

Figure E.1(a) shows 5 clusters of nodes. We assume that nodes in each cluster are internally connected by a cluster-network. The clusters on the other hand are not connected but have to be connected by a backbone-network, in order to be able to communicate. Figure E.1(b) shows a possible backbone-network. A solution to the GFCND problem consists of a set of hubs, one for each cluster, illustrated as filled squares. The hubs are connected by edges to form the backbone-network. The required capacity of the edges is illustrated in Figure E.1(b) as different thicknesses of the edges. The required capacity of the edges depend on the communication demand and how the demands are routed. The cost of the backbone-network is the sum of the edge establishment costs and the edge capacity costs. Given a set of clusters of nodes, the GFCND problem consists of locating hubs and determining the cheapest backbone-network.

The outline of this paper is as follows. Section E.2 discusses related problems. Section E.3 presents a Mixed Integer Programming (MIP) model for the GFCND problem. Section E.4 describes a branch-cut-and-price algorithm which solves the GFCND problem. Section E.5 describes how test instances are generated and Section E.6 presents computational results for the test instances. Finally, Section E.7 gives some concluding remarks.

E.2 Related Problems

The GFCND problem is related to two different types of well known optimization problems "Generalized Network Design" [4] and "Fixed-Charge Network Design" [10, 2]. If the hubs are known in advance, the GFCND problem becomes the Fixed-Charge Network Design, hence the name.

E.2.1 The Fixed-Charge Network Design Problem

The problem of designing cost efficient networks, taking into account both edge establishment costs and edge capacity costs, is denoted the Fixed-Charge Network Design problem. The Fixed-Charge Network Design problem is NP-hard [7]. Its application to many different types of problem areas is described in [10]. A number of different optimization method has been applied to the Fixed-Charge Network Design problem: Benders Decomposition e.g. [3, 9], Dual-Ascent [2], Cutting-Plane [1] and Lagrangian relaxation [6, 7].

E.2.2 Hierarchical Network Design

Hierarchical network design problems combine a series of interconnected optimization problems:

Clustering: Decide which nodes should belong to the same cluster.

Hub-Selection: For each cluster select a hub from the cluster to include in the backbone-network.

Cluster-Network Design: Determine the network which connects the nodes inside each cluster.

Backbone-Network Design: Determine the backbone-network which connects the hub nodes.

Cluster-Network Routing: Route the communication demands in the cluster-networks.

Backbone-Network Routing: Route the communication demands in the backbone-network.

Hierarchical network design problems combine these interdependent subproblems and solutions are hierarchical networks able to handle the communication demands. The papers reviewed in [8] consider such hierarchical network design problems. In some cases the subproblems are solved separately step by step and thus optimal solution of the combined problem is not guaranteed.

The GFCND problem is a combination of some of the hierarchical network design subproblems. Given a set of clusters, the Hub-Selection, the Backbone-Network Design, and the Backbone-Network Routing problems constitute the GFCND problem.

The costs incurred by the Cluster-Network Design and Cluster-Network Routing problems can be included in GFCND as follows. Given a set of clusters, solve the Cluster-Network Design and Cluster-Network Routing problems for each cluster and for each node chosen as hub. Associate the computed cost with selecting the node as hub in GFCND. The difficulty of solving the Cluster-Network Design and Cluster-Network Routing problems depend on the actual cluster networks. E.g. the cluster networks may be star, tree, ring or even Fixed-Charge Networks. In [11] a GFCND like problem is considered where clusters are Fixed-Charge Networks and the clusters are not given in advance.

E.2.3 The Generalized Network Design Problem

A number of well known network optimization problems are generalized by introducing clusters of nodes in [4]. Many standard network optimization problems assume that all nodes have to be connected, but in [4] this requirement is altered by requiring that one node from each cluster has to be connected. Three of the problems discussed in [4] are related to the list of optimization problems in Section E.2.2: The Generalized Minimum Spanning Tree problem, the Generalized Traveling Salesman Problem, and the Generalized Minimum 2-edge-connected Spanning Network problem. The problems combine the Hub-Selection problem with the Backbone-Network Design with special constraints on the architecture of the backbone-network. A particularly interesting paper is [5], where a branch-and-cut algorithm for the Generalized Traveling Salesman Problem is considered. This approach is tested on instances with up to 89 clusters and 442 cities. Some of the techniques which we employ are inspired by this paper.

E.3 A MIP Model for the GFCND Problem

In this Section we describe a MIP model for the GFCND problem. This model defines the previous described GFCND problem.

Let V be the set of all nodes, the edges E be a set of unordered node-pairs and the arcs A be a set of ordered node-pairs. For each edge there exists exactly two arcs of opposite direction. Let C be a set of clusters where each cluster is a subset of V such that $\bigcup_{c \in C} c = V$ and $c \bigcap c' = \emptyset$ for distinct clusters c and c'. The demands D is the set of unordered cluster-pairs. We use the index $e \in E$ for the edges, the index $a \in A$ for the arcs, the index $d \in D$ for the demands and index $i \in V$ for the nodes. Furthermore we use s_a and t_a to denote the start and terminating node of arc a. Correspondingly we use s_d and t_d to denote the start and terminating cluster of demand d.

The binary decision variable h_i correspond to whether or not node i should be selected to be a hub-node. The binary decision variables y_e correspond to whether or not edge e between two hub-nodes should be established. Finally, the fractional decision variable x_a^d correspond to the fraction of demand d along arc a.

The communication demand volume for demand d is given by b_d. Furthermore, the capacity cost of arc a is given by c_a, which is the cost per unit of demand using that arc. Finally, the fixed cost of edge e is f_e. Given these definitions, the MIP model of the GFCND problem is defined as follows.

$$\min \quad \sum_{e \in E} f_e y_e \quad + \sum_{d \in D, a \in A} c_a b_d x_a^d \tag{1}$$

$$\text{s.t.} \quad \sum_{a | s_a = i} x_a^d - \sum_{a | t_a = i} x_a^d = \begin{cases} h_i & \text{if } i \in s_d \\ -h_i & \text{if } i \in t_d \\ 0 & \text{otherwise} \end{cases} \quad d \in D, i \in V \tag{2}$$

$$x_a^d + x_b^d \leq y_e \quad a, b \in A, s_a = t_b, t_a = s_b, \{s_a, t_a\} = e \in E, d \in D \tag{3}$$

$$y_e \leq h_i \quad e \in E, i \in e \tag{4}$$

$$\sum_{i \in c} h_i = 1 \quad c \in C \tag{5}$$

$$h_i, y_e \in \{0, 1\} \tag{6}$$

$$x_a^d \in [0, 1] \tag{7}$$

In the above MIP model, the objective, equation (1) calculates the combined edge establishment costs and edge capacity requirement costs. The flow constraints (2) ensure that the flow requirement for each demand is fulfilled. The left hand side is the outflow minus the inflow of demand d in node i. The right hand side is h_i if demand d orginates in cluster s_d, $-h_i$ if d terminates in cluster t_d, and 0 otherwise. By using h_i and $-h_i$ instead of 1 and -1, as is customary, flows originate and terminate in established hubs, only. The constraints (3) ensure that flow is only allowed along established edges. The constraints (4) ensure that only edges between selected hubs may be established. The constraints (5) ensure that exactly one hub node is selected in each cluster. Finally the domains of the h_i, y_e and x_a^d variables are defined by (6) and (7). By relaxing constraint (6) the linear programming relaxation of the MIP model is obtained. We denote this the Linear Programming (LP) model.

By including the following constraints, the value of the linear programming relaxation may improve.

$$\sum_{e=\{k,i\}|k\in C} y_e \leq h_i \qquad \forall\, c \in C, i \in V, i \notin c \tag{8}$$

These constraints (8) are denoted the fan constraints [5]. For a cluster c and a node $i \notin c$, the constraints express that the sum of the edges connecting the nodes in cluster c to node i can at most be one and only if node i is a hub. The fan constraints are valid, since the edges can only be selected if node i is a hub. Also, since only one hub is allowed for each cluster, only *one of* the edges can be selected.

The fan constraints (8) replace constraints (4) and are stronger, since constraints (8) can be obtained by lifting constraints (4).

E.4 Solving the GFCND problem

The MIP model of the GFCND problem described in Section E.3 can be used to find optimal solutions using a MIP solver. However, given the substantial number of variables and constraints, only problems with few nodes and clusters may be solved this way. As an alternative to using a MIP solver directly, we present a branch-cut-and-price algorithm. The branch-cut-and-price algorithm takes advantage of the fact that only few non-zero variables and few binding constraints exist. The branch-cut-and-price algorithm is specified in Figure E.2.

Initially the branch-cut-and-price algorithm initialize the three sets Vars, Cons and Branches. The set Vars contains a subset of the variables in the LP model. It is initialized to contain all of the h_i variables and all of the y_e variables, but only a subset of the x_a^d variables. The x_a^d variables are determined such that a feasible solution exists. To that end, a node is selected randomly from each cluster and a Minimum Spanning Tree (MST) is determined. For each demand d the variables x_a^d along the shortest path in the MST are included in Vars. Correspondingly the set Cons constains a subset of the constraints. It is initialized to contain all of the constrains (5). Furthermore, if a previously selected x_a^d variable is a term in a constraint (2) or (3), this constraint is included in Cons. Branches contains a set of branches, where each branch is a set of variable fixations. Each fixation either fix a h_i variable or a y_e variable to 0 or 1. Initially Branches is set to contain one branch, the branch with no variable fixations. Finally the Incumbent value is intialized to infinity.

In the main loop a Branch is selected from Branches and the inner loop is

```
VARS = initial set of variables
CONS = initial set of constraints
BRANCHES = set consisting of an initial branch
INCUMBENT = +∞
do
  Select and remove BRANCH ∈ BRANCHES
  do
    LP-LOWERBOUND = Solve LP(CONS, VARS, BRANCH)
    Add variables with negative reduced cost to VARS
    Add violated constraints to CONS
  while variables or constraints added
  if LP solution is integer and LP-LOWERBOUND < INCUMBENT:
    Update incumbent: INCUMBENT = LP-LOWERBOUND
    Fathom
  else if LP-LOWERBOUND ≥ INCUMBENT:
    Fathom
  else
    Select branching variable v
    Add the two branches v = 1 and v = 0 to BRANCHES
while BRANCHES ≠ ∅
```

Figure E.2: *The branch-cut-and-price algorithm*

entered. Given BRANCH, VARS and CONS, the LP model is solved. The first time the LP model is solved, it is solved from scratch. In following iterations, the LP model is *resolved*, i.e. the previous solution is used as a starting point. After the LP model has been solved, the variables x_a^d with negative reduced cost are added and the constraints (2), (3), and (8) which are violated are added. Since there are only polynomially many variables and constraints, these are found through enumeration. When the inner loop is terminated, the LP model with the variable fixations in BRANCH has been solved. The first time, the inner loop is terminated, the initial branch with no variable fixations have been solved. This solution is the solution to the LP model and is denoted the *Root-LP* solution.

Given the solution to the LP model, it is tested if the solution is integer, i.e whether the h_i and y_e variables have integer values. If this is the case, and the LP-LOWERBOUND is better than the INCUMBENT, the INCUMBENT is updated and the BRANCH is fathomed, i.e. it is not considered anymore. If otherwise the LP-LOWERBOUND value is worse than the current value of the INCUMBENT, the BRANCH is fathomed. If the BRANCH has not been fathomed, branching is necessary. A branching variable is determined as the highest valued fractional

h_i variable, and if no fractional h_i variable exists, the highest valued fractional y_e is selected. Two new branches are created based on BRANCH. In addition to the variable fixations in BRANCH, the branching variable is fixed to 0 in one branch and 1 in the other branch.

Branch-cut-and-price algorithms may require more advanced branching schemes than simple variable branching, see e.g. [12]. This is, however, not necessary for the branch-cut-and-price algorithm in this paper, since the binary variables are all included in VARS initially. The reason this is possible is, that there are only $|V|(|V|+1)/2$ binary variables.

E.5 Test Instance Generation

Each test instance consists of a number of nodes randomly located in the unit square. The clusters are created in that same way as in [5] and as described in the following. Let there be given a number of nodes $|V|$ and a desired number of clusters $|C|$. Center nodes are selected iteratively for each of the $|C|$ clusters. The first center node is selected randomly. The remaining center nodes are selected such that the current center node is the node with largest minimum distance to the previously selected center nodes. The clusters are then formed by associating each of the remaining $|V| - |C|$ nodes to the closest center node.

An important characteristic of the test instances is the density of the optimal solution, which is usually expressed by the average node degree. However, since a solution interconnects clusters only, the density of a solution is expressed as the *average cluster degree*. The average cluster degree is the number of edges times 2 divided by the number of clusters. The average cluster degree of the optimal solution is influenced by modifying the ratio between the establishment costs and the capacity costs and by modifying the demands. For each pair of clusters a demand d is defined. The value of the demand, b_d is set equal to a uniformly distributed random variable between 0 and 20 times the product of the number of nodes in each of the two clusters.

Given a scaling factor R it turns out, that the following equation leads to reasonable constant average cluster degrees regardless of the value of $|V|$ and $|C|$.

$$f_e = R \cdot c_e \cdot |V| \tag{9}$$

The edge capacity costs c_e are set equal to the Euclidean distance. Given R and $|V|$ the edge establishment costs f_e are then obtained by equation (9).

After some experimentation, the scaling factor is set to $R = 0.5$. This leads to average cluster degrees between 2.4 and 4.2. For smaller scaling factors, the average cluster degrees increase and the test instances become more time consuming for the branch-cut-and-price algorithm. For higher scaling factors, the average cluster degrees decrease and the test instances become less time consuming for the branch-cut-and-price algorithm.

E.6 Computational Results

To test the branch-cut-and-price algorithm we generate test instances with 50 to 300 nodes in steps of 25 nodes and with 10 to 30 clusters in steps of 10 clusters. The tests are performed on a 1000 Mhz SUN Fire V440, running Solaris 9. We use CPLEX as LP-solver. A time limit of 10 hours is set and the program size is limited to 4 GB of ram. The results regarding the branch-cut-and-price algorithm are given in Table E.1.

Problem		A. Cluster	Time (sec.)			Branch	Gap %	
$\|C\|$	$\|V\|$	Degree	Total	Root-LP	Branch	Nodes	Fan	No Fan
10	50	2.4	42	40	2	1	0.04 %	0.04 %
	75	2.8	73	73	0	0	0.00 %	0.00 %
	100	3.4	234	234	0	0	0.00 %	0.00 %
	125	2.6	605	604	0	0	0.00 %	0.15 %
	150	3.8	654	653	0	0	0.00 %	0.00 %
	175	3.8	767	767	0	0	0.00 %	0.00 %
	200	4.2	2785	2784	0	0	0.00 %	0.19 %
	225	3.6	3318	3281	37	1	0.02 %	0.23 %
	250	3.2	5499	5497	0	0	0.00 %	0.00 %
	275	4.0	4658	4656	0	0	0.00 %	0.00 %
	300	3.4	7510	7508	0	0	0.00 %	0.00 %
20	50	2.9	1499	1182	317	6	0.14 %	0.14 %
	75	2.8	4904	4903	0	0	0.00 %	0.00 %
	100	3.2	11978	11976	0	0	0.00 %	0.00 %
	125	3.5	34385	29799	4586	7	0.03 %	0.03 %
	150	3.5	34361	34358	0	0	0.00 %	0.00 %
30	50	2.6	23511	16108	7404	12	0.14 %	0.14 %

Table E.1: *Branch-cut-and-price algorithm results*

The first two columns contain the number of clusters and nodes for the given test instances. The third column contains the average cluster degree for the optimal network generated by the branch-cut-and-price algorithm. The fourth, fifth and sixth columns contains, respectively, the total computation time of the branch-cut-and-price algorithm, the time required for solving the Root-LP problem and the time required by the branch-cut-and-price algorithm after solving the Root-LP. The seventh column contains the number of branches required to obtain the optimal integer solution. The eighth and ninth columns contain the integrality gap between the optimal integer solution and the optimal Root-LP solution with and without the fan constraints (8), respectively. The "No Fan" integrality gap is found by solving the Root-LP without the fan constraints (8), but with (4), and comparing this optimal solution with the optimal integer solution.

The results show that the branch-cut-and-price algorithm can find optimal solutions for test instances with up to 10 clusters and 300 nodes, 20 clusters and 150 nodes, or 30 clusters and up to 50 nodes. Furthermore it is clear that the majority of the computation time is spent on solving the Root-LP problem. The reason is that the integrality gap is small and thus few branches are necessary. In fact, for 12 out of 17 instances, branching is not necessary at all, since the Root-LP solution is integer. Hence only a small fraction of the total computation time is spent on branching. When the fan constraints are not used, 7 out of 17 instances have an integrality gap. In 3 of these 7 instances, the integrality gap is reduced by adding fan constraints and in 2 instances the integrality gap is even closed.

Since the solution of the Root-LP problem is the most time consuming part, we will look more closely at the Root-LP solution time, see Table E.2.

Problem		Root-LP	Branch-cut-and-price, time (sec.)				Directly LP
$\|C\|$	$\|V\|$	Iterations	Pricing	Separation	Resolving	Total	Time (sec.)
10	50	38	4	15	21	40	179
	75	41	9	32	32	73	447
	100	56	24	78	131	234	1740
	125	87	41	120	440	604	5273
	150	82	57	170	422	653	6582
	175	111	89	237	432	767	9722
	200	75	125	462	2189	2784	-
	225	117	170	581	2515	3281	-
	250	111	218	761	4500	5497	-
	275	105	260	971	3402	4656	-
	300	131	347	1164	5962	7508	-
20	50	64	69	263	848	1182	5115
	75	220	225	738	3926	4903	18589
	100	56	293	1319	10358	11976	-
	125	253	721	2803	26236	29799	-
	150	88	876	3973	29488	34358	-
30	50	148	537	1628	13930	16108	30098

Table E.2: *Root-LP solving time statistics*

For each instance, the third column contains the number of iterations required when solving the Root-LP. Column four, five, six, and seven contain, respectively, the time required for pricing, the time required for separation, the time required for resolving, and the total time required for solving the Root-LP. Finally the eighth column contains the computation time required by CPLEX to solve the Root-LP problem directly, i.e. with all constraints and variables included. The entries containing a dash could not be solved directly by CPLEX because of memory limitations. The instances that CPLEX can solve directly require significantly more computation time than the computation time required for solving the Root-LP using the branch-cut-and-price algorithm.

From Table E.2 it is clear that the resolve operation when solving the Root-LP problem dominates the computation time. For all but one test instance it takes up more than half of the total computation time, and this fraction increases with the number of nodes and clusters. Hence, to reduce the overall computation time significantly, the resolve time should be reduced. The resolve time depends on the number of constraints and number of variables generated. The number of generated constraints and the number of binding constraints after solving the Root-LP are given in Table E.3.

Problem		Total	Generated		Binding	
$\|C\|$	$\|V\|$	#	#	%	#	%
10	50	54555	4208	7.7 %	2751	5.0 %
	75	118769	5450	4.6 %	3442	2.9 %
	100	213721	7581	3.5 %	4539	2.1 %
	125	327144	10298	3.1 %	6530	2.0 %
	150	478590	10455	2.2 %	5976	1.2 %
	175	646580	10566	1.6 %	5903	0.9 %
	200	848457	14531	1.7 %	8261	1.0 %
	225	1073270	16247	1.5 %	9208	0.9 %
	250	1310585	18539	1.4 %	10534	0.8 %
	275	1575959	18587	1.2 %	10313	0.7 %
	300	1868922	21686	1.2 %	12053	0.6 %
20	50	236313	19785	8.4 %	12694	5.4 %
	75	517659	29467	5.7 %	18206	3.5 %
	100	924765	36063	3.9 %	22720	2.5 %
	125	1439391	48534	3.4 %	29548	2.1 %
	150	2052129	52614	2.6 %	32218	1.6 %
30	50	544622	54215	10.0 %	32619	6.0 %

Table E.3: *Constraints generated and binding*

For each instance, the third column contains the total number of constraints in the MIP model. Columns four and five contain the number of generated constrains and columns six and seven contain the number of binding constraints in the optimal Root-LP solution. Only a small fraction of the constraints need to be generated, but the number of constraints generated increases with the number of nodes and the number of clusters. However, the *fraction* of the total number of constraints decrease with the number of nodes. Furthermore, at least half of the generated constraints are binding in the optimal solution, hence at most half of the generated constraints are redundant.

The number of generated variables and the number of non-zero variables after solving the Root-LP are given in Table E.4.

Problem		Total	Generated		Non-zero	
$\|C\|$	$\|V\|$	#	#	%	#	%
10	50	100332	16548	16.5 %	187	0.1864 %
	75	222024	27843	12.5 %	115	0.0518 %
	100	403230	49416	12.3 %	110	0.0273 %
	125	620199	55588	9.0 %	120	0.0193 %
	150	910787	78556	8.6 %	106	0.0116 %
	175	1233407	108638	8.8 %	108	0.0088 %
	200	1621638	151830	9.4 %	105	0.0065 %
	225	2054277	165116	8.0 %	337	0.0164 %
	250	2511122	192565	7.7 %	112	0.0045 %
	275	3022294	242587	8.0 %	105	0.0035 %
	300	3586883	248903	6.9 %	109	0.0030 %
20	50	448106	61591	13.7 %	1359	0.3033 %
	75	996009	112420	11.3 %	606	0.0608 %
	100	1793467	165789	9.2 %	560	0.0312 %
	125	2804285	253558	9.0 %	807	0.0288 %
	150	4009794	335084	8.4 %	561	0.0140 %
30	50	1039153	143615	13.8 %	4577	0.4405 %

Table E.4: *Variables generated and non-zero*

For each instance, the third column contains the total number of variables in the MIP model. Columns four and five contain the number of generated variables and columns six and seven contain the number of non-zero variables in the optimal Root-LP solution.

The number of non-zero variables in the optimal solutions remain constant regardless of the number of nodes in the test instances. Unfortunately the number of generated variables increase with the number of nodes in the test instances. This indicates that a significant reduction of the computation time of the algorithm may be achieved by selecting better variables. This should both reduce the resolve time per iteration and reduce the number of needed iterations, for an overall reduction in the resolve time.

The instances ($|C| = 10, |V| = 50$), ($|C| = 10, |V| = 225$), ($|C| = 20, |V| = 50$), ($|C| = 20, |V| = 125$) and ($|C| = 30, |V| = 50$) have significantly more non-zero variables than the other instances with similar characteristics. This is due to the fact that these instances have a non-zero integrality gap.

The number of generated variables and constraints are significantly below the total number of variables and constraints, respectively. This is the reason that the branch-cut-and-price algorithm can solve instances with many more nodes and clusters than if the MIP model is solved directly as seen in Table E.2.

E.7 Conclusion

This paper has presented the GFCND problem. The GFCND problem involves interconnecting clusters by a Fixed-Charge network. A MIP model for the GFCND problem has been formulated and enhanced with additional cuts to achieve a smaller integrality gap. A branch-cut-and-price algorithm for the GFCND problem has been presented and tested. Violated constraints and variables with negative reduced costs are found using enumeration

The algorithm has been tested on instances with up to 10 clusters and 300 nodes or up to 30 clusters and 50 nodes. It has been demonstrated that the integrality gap is zero in 12 out of 17 data instances and less than 0.14% for the remaining data instances. Because of the small integrality gap, the computational bottleneck is the solution of the Root-LP problem. The branch-cut-price algorithm is efficient compared with solving the MIP model directly, since only a fraction of the constraints and variables are generated.

Acknowledgments

The authors would like to thank the anonymous reviewer for helpful comments.

Bibliography

[1] Balakrishnan, A. (1987) LP extreme points and cuts for the fixed-charge network design problem, *Mathematical Programming* 39(3), 263-284.

[2] Balakrishnan, A. and Magnanti, T.L. and Wong, R.T. (1989) A Dual-Ascent Procedure for Large-Scale Uncapacitated Network Design, *Operations Research* 37(5), 716-740.

[3] Costa, A.M. (2005) A survey on benders decomposition applied to fixed-charge network design problems, *Computers and Operations Research* 32(6), 1429-1450.

[4] Feremans, C. and Labbe, M. and Laporte, G. (2003) Generalized network design problems, *European Journal of Operational Research*, 148(1), 1-13.

[5] Fischetti, M. and Salazar Gonzalez, J.J and Toth, P. (1997) A branch-and-cut algorithm for the symmetric generalized traveling salesman problem, *Operations Research*, 45(3), 378-394.

[6] Gendron, B. and Crainic, T. G. and Frangioni, A. (1999) Multicommodity Capacitated Network Design, *Telecommunications Network Planning*, Kluwer, 1-19.

[7] Holmberg, K. and Yuan, D. (1998) A Lagrangean approach to network design problems, *International Transactions in Operational Research*, 5(6), 529-539.

[8] Klincewicz, J. G. (1998) Hub location in backbone/tributary network design: a review, *Location Science*, 6(1-4), 307-335.

[9] Magnanti, T.L. and Mireault, P. and Wong, R.T. (1986) Tailoring Benders Decomposition for Uncapacitated Network Design, *Mathematical Programming Studies*, 26, 112-154.

[10] Magnanti, T.L. and Wong, R.T. (1984) Network design and transportation planning: models and algorithms, *Transportation Science*, 18(1), 1-55.

[11] Thomadsen, T. (2005) Joint Hub Location, Node Clustering and Network Design of Two-Tiered Meshed Networks, *To appear in Proceedings of INOC 2005.*

[12] Vanderbeck, F. and Wolsey, L.A. (1996) An exact algorithm for IP column generation, *Operations Research Letters*, 19(4), 151-159.

Appendix F

The Two-Layered Fully Interconnected Network Design Problem – Models and an Exact Approach

Submitted for *Computers and Operations Research*

The Two-Layered Fully Interconnected Network Design Problem – Models and an Exact Approach

Tommy Thomadsen[1] and Jesper Larsen[2]

Abstract

This paper considers the design of two-layered fully interconnected networks. A two-layered network consists of clusters of nodes, each defining an access network and a backbone network. We consider the integrated problem of determining the access networks and the backbone network simultaneously. A mathematical model is presented, but as the linear programming relaxation of the mathematical model is weak, a model based on the set partitioning model and column generation approach is also developed. The column generation subproblems are solved by solving a series of quadratic knapsack problems. We obtain superior bounds using the column generation approach than with the linear programming relaxation. The column generation method is therefore developed into an exact approach using the Branch-and-Price framework. With this approach we are able to solve problems consisting of up to 25 nodes in reasonable time. Given the difficulty of the problem, the results are encouraging.

Keywords: Hierarchical networks, Fully interconnected networks, Hub location, Branch-and-Price.

F.1 Introduction

Wired communication networks are usually organized in a hierarchal structure based on two or more layers. This structure has proven robust to changing demands and upgrades and is seen as the right compromise between cost and redundancy. We consider networks with two layers, but the analysis is generalizable to more layers.

The two layers in the network are denoted the backbone network and the access networks. The backbone network connects disjoint clusters of nodes, each

[1]Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark. Email: tt@imm.dtu.dk

[2]Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark. Email: jla@imm.dtu.dk

comprising an access network. The node connecting an access network to the backbone network is called a hub. An example is shown in Figure F.1. Here hubs are shown as squares, thin lines are connections in an access network, while thick lines represent connections in the backbone network. The dashed lines mark the clusters, each representing an access network.

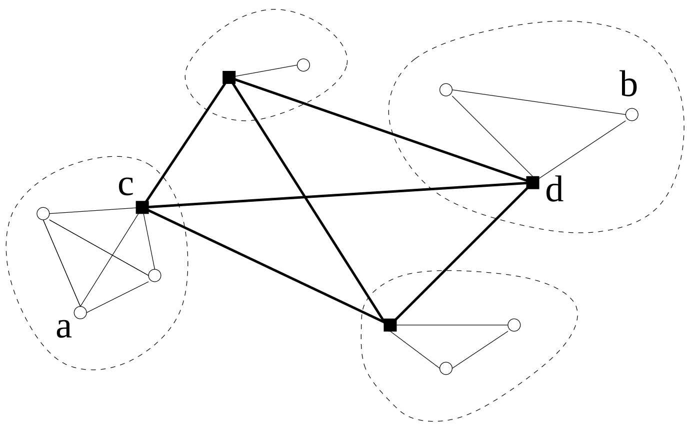


Figure F.1: Example of a two-layered network. All access networks and the backbone network are fully interconnected.

When designing such hierarchal or layered networks, a number of interrelated questions have to be resolved: Which nodes should be hubs, how should we define the clusters, and which interconnections should we allow. Since theses problems are interrelated, they should be addressed by an integrated approach in order to ensure an optimal solution.

This paper considers the joint selection of hubs and clustering of nodes of two-layered networks. In each of the layers, we assume the networks to be fully interconnected. This corresponds to Figure F.1, where the access networks and the backbone network are fully interconnected, that is, for two nodes in the backbone or the same cluster, there is a link. The objective is to minimize link costs consisting of a cost for each link that needs to be established. This problem is denoted the Fully Interconnected Network Design Problem (FINDP).

Note that communication between two nodes in the same access network (eg. a and c in Figure F.1) is handled within the access network. On the other hand, communication between two nodes not in the same access network has to be routed via the backbone network. So communication from a to b is routed via c and d in the backbone network in Figure F.1.

Klincewizc [6] surveys work on two-layered network design. A significant part of the work reviewed focus on fully interconnected backbone networks. In particular [3] and [8] consider the design of fully interconnected backbone network and a star network in the access networks. Using the fully interconnected network on both layers is an open problem according to the survey paper of Klincewizc.

The FINDP could be used in setting up a computation cluster of computers. Instead of affording a direct connection between all computers, they are divided into two layers, with a backbone network ensuring fast access beyond the access network of a computer.

The paper is organized as follows. Section F.2 presents a mathematical model for the FINDP problem. As the linear relaxation of the mathematical model is weak, we develop a column generation approach in Section F.3, and present the Branch-and-Price framework to obtain integral solutions in Section F.4. Experimental results are presented in Section F.5, and finally the conclusions are given in Section F.6.

F.2 Network Design

First we formulate a mathematical model for the FINDP. Consider the graph $G = (V, E)$, where V is the set of nodes and E is the set of undirected links. Let c_{ij} denote the cost of link ij in E.

We consider a problem where there are lower and upper bounds on the number of clusters and the number of nodes in the clusters. Let b_{min} and b_{max} be a lower bound and an upper bound on the number of clusters, respectively. Furthermore v_{min} is the lower bound on the number of nodes in any cluster and v_{max} the corresponding upper bound.

We initially define three sets of variables x_{ij}, y_{ij} for i, j in V, $i < j$, and h_i for i in V. The variable h_i is 1 if node i is hub for a cluster, and 0 otherwise. The variable x_{ij} is 1 if ij is a link in the access and 0 otherwise, and correspondingly y_{ij} is 1 if ij is a link in the backbone network and 0 otherwise. Initially we get:

$$\begin{aligned}
\min \quad & \sum_{ij \in E} c_{ij} x_{ij} + \sum_{ij \in E} c_{ij} y_{ij} && && (1)\\
\text{s.t.} \quad & y_{ij} + x_{ij} \leq 1 && \forall ij \in E && (2)\\
& h_i + h_j + x_{ij} \leq 2 && \forall ij \in E && (3)
\end{aligned}$$

$$
\begin{aligned}
& y_{ij} \le h_k && \forall ij \in E, k \in \{i,j\} && (4)\\
& x_{ik} + x_{jk} \le x_{ij} + 1 && \forall i,j,k \in V, i < j, k \neq i, k \neq j && (5)\\
& y_{ik} + y_{jk} \le y_{ij} + 1 && \forall i,j,k \in V, i < j, k \neq i, k \neq j && (6)\\
& x_{ij}, y_{ij}, h_i \text{ binary} && && (7)
\end{aligned}
$$

The objective function (1) is the sum of costs in the access networks and the backbone network. Inequality (2) ensures that a link cannot be used both in the backbone network and the access network at the same time. Next inequality (3) states that if both nodes i and j are hubs then there can be no access network link between these nodes. Now inequality (4) says that if a node k is not a hub, then a backbone network link cannot be incident to k. Finally (5) and (6) ensure that the access network, respectively the backbone network are fully interconnected.

The model can be strengthened by adding

$$h_i + h_j \le y_{ij} + 1 \qquad \forall ij \in E \tag{8}$$

where (2) and (8) dominates (3), and (4) together with (8) dominates (6).

This initial model does, however, not ensure that each cluster contains a hub. Therefore, we introduce the variable w_{ij} for each ordered pair (i,j) where $i,j \in V, i \neq j$. If w_{ij} is 1 node i is hub and node j is connected to i and 0 otherwise, i.e.

$$w_{ij} = h_i x_{ij}.$$

Described by linear constraints we get:

$$
\begin{aligned}
& w_{ij} \le h_i && \forall i,j \in V, i \neq j && (9)\\
& w_{ij} \le x_{ij} && \forall i,j \in V, i \neq j && (10)\\
& h_i + x_{ij} \le 1 + w_{ij} && \forall i,j \in V, i \neq j && (11)\\
& h_j + \sum_{i, i \neq j} w_{ij} = 1 && \forall ij \in E && (12)\\
& w_{ij} \text{ binary} && && (13)
\end{aligned}
$$

The constraints (9) and (10) force w_{ij} to 0 if node i is not a hub or if there is no link in the access network between i and j. Inequality (11) set w_{ij} to 1 if node i is a hub *and* there is a link between i and j in the network. Then (12) either forces node i to be a hub *or* to be connected to exactly one hub j, and as a consequence, each cluster contains a hub.

Finally we describe bounds on the number of clusters (14) and nodes in each cluster (15):

$$b_{\min} \leq \sum_i h_i \leq b_{\max} \tag{14}$$

$$v_{\min} - 1 \leq \sum_j x_{ij} \leq v_{\max} - 1 \qquad \forall i \in V \tag{15}$$

When we refer to the model FINDP for the two-layered fully interconnected network problem we refer to the model defined by (1)-(2), (4)-(5) and (7)-(15). The linear programming relaxation is obtained by replacing (7) and (13) with a non-negativity constraint on all variables. This model is denoted LP-FINDP.

F.3 Decomposition and Column Generation

LP-FINDP generally provides a poor bound on the optimal value of the FINDP. This is largely due to the weak LP relaxation and the inherent symmetry in the formulation.

In order to obtain a better formulation of the FINDP problem let $\mathcal{C}$ be the set of all clusters with a hub selected, and let $\mathcal{B}$ be the set of all backbone networks, where each backbone network is a set of hubs. Let a_i^c be 1 if node i is in cluster c and 0 otherwise. Furthermore let s_i^c be 1 if node i is hub in cluster c and 0 otherwise. For the backbone, let s_i^b be 1 if node i is in backbone, and 0 otherwise. For a cluster c in $\mathcal{C}$ let c_c be the cost of the cluster, i.e. the sum of all cost on the links in c, that is,

$$c_c = \sum_{i,j,i<j} a_i^c a_j^c c_{ij}$$

and correspondingly let c_b be the cost of the backbone b in $\mathcal{B}$, so:

$$c_b = \sum_{i,j,i<j} s_i^b s_j^b c_{ij}$$

Now we can formulate an alternative model of the FINDP problem. Variables are u_c which are 1 if cluster c in $\mathcal{C}$ is selected, 0 otherwise and v_b which are 1 if backbone b in $\mathcal{B}$ is selected, 0 otherwise.

$$
\begin{aligned}
\min \quad & \sum_{c\in\mathcal{C}} c_c u_c + \sum_{b\in\mathcal{B}} c_b v_b & & (16)\\
\text{s.t.} \quad & \sum_{c\in\mathcal{C}} a_i^c u_c \qquad\qquad\quad = 1 \quad i \in V & & (17)\\
& -\sum_{c\in\mathcal{C}} s_i^c u_c + \sum_{b\in\mathcal{B}} s_i^b v_b = 0 \quad i \in V & & (18)\\
& u_c, v_b \text{ binary} & & (19)
\end{aligned}
$$

Here, (16) is the objective function minimizing the accumulated cost of the access networks and the backbone network. The equalities (17) ensure that all nodes belong to exactly one access network, while the constraints (18) ensure that hub nodes are in the backbone network.

Consider a small example of 4 nodes that should be divided into exactly 2 clusters of precisely 2 nodes each. For this small instance, it is possible to enumerate all possibilities. Let us denote the nodes a, b, c and d. Figure F.2 shows the coefficient matrix and right-hand sides for this small problem. Each possible cluster consists of 2 nodes, so for each possible cluster there are two hub candidates. Therefore each feasible cluster results in two different columns representing the cluster but with different hubs (represented by the -1 coefficient in the lower half). Then follows a column for each possible backbone solution. If e.g. we choose the two clusters (a, b) and (c, d), then depending on the choice of hub, the matching column in the rightmost part will enforce the right cost of the backbone network.

$$
\begin{array}{l|cccccccccccc|cccccc|l}
a: & 1 & 1 & 1 & 1 & 1 & 1 & & & & & & & & & & & & & =1\\
b: & 1 & 1 & & & & & 1 & 1 & 1 & 1 & & & & & & & & & =1\\
c: & & & 1 & 1 & & & 1 & 1 & & & 1 & 1 & & & & & & & =1\\
d: & & & & & 1 & 1 & & & 1 & 1 & 1 & 1 & & & & & & & =1\\
\hline
a: & -1 & & -1 & & -1 & & & & & & & & 1 & 1 & 1 & & & & =0\\
b: & & -1 & & & & & -1 & & -1 & & & & 1 & & & 1 & 1 & & =0\\
c: & & & & -1 & & & & -1 & & & -1 & & & 1 & & 1 & & 1 & =0\\
d: & & & & & & -1 & & & & -1 & & -1 & & & 1 & & 1 & 1 & =0
\end{array}
$$

Figure F.2: Coefficient matrix for a small example consisting of 4 nodes. Blanks in the matrix represent entries of value 0.

It is only possible to enumerate all columns for very small instances. E.g. for a problem with $K = 20, v_{\min} = b_{\min} = 6$ and $v_{\max} = b_{\max} = 8$ we get approximately 2.1 million columns. In order to avoid generating all clusters and backbones a priori, we use the iterative method of column generation. The clusters and backbone networks are generated as they are needed. For each constraint (17) we associate a dual variable α_i and for each constraint (18) we associate the dual variable β_i.

Preliminary results show significantly better bounds by adding the following strengthening constraint to the model:

$$\sum_{b \in \mathcal{B}} v_b = 1 \tag{20}$$

The constraint (20) has an associated dual variable γ.

Most often column generation is seen as a master problem and a subproblem. The master problem solves an LP relaxation and delivers dual variables to the subproblem where new variables are computed in case a better one exists. The linear programming relaxation of the FINDP (defined by (16) to (20)) problem is denoted CG-FINDP and is obtained by replacing (19) with non-integrality constraints. For the CG-FINDP the master problem is associated with *two* subproblems; one for generating clusters and one for generating backbone configurations, see Figure F.3.

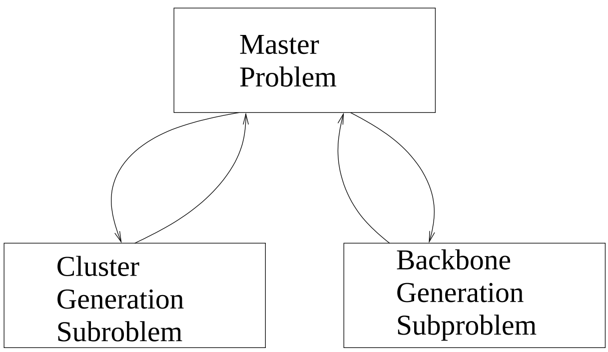


Figure F.3: Overview of the column generation algorithm.

F.3.1 The Backbone generation subproblem

For an optimal solution the reduced cost of a backbone column is:

$$\sum_{ij \in E} c_{ij} y_{ij} - \sum_{i \in V} \beta_i s_i - \gamma \tag{21}$$

where y_{ij} and s_i are binary variables representing whether the link ij respectively the node i is part of the backbone network.

In order to find a new column to enter the basis, we seek the column with the most negative reduced cost. By multiplying the reduced cost with -1 the objective function is:

$$\max \sum_{i \in V} \beta_i s_i + \gamma - \sum_{ij \in E} c_{ij} y_{ij} \tag{22}$$

The constraints for obtaining a feasible backbone network are:

$$y_{ij} \leq s_i \qquad ij \in E \tag{23}$$

$$y_{ij} \leq s_j \qquad ij \in E \tag{24}$$

$$s_i + s_j \leq y_{ij} + 1 \qquad ij \in E \tag{25}$$

$$b_{min} \leq \sum_{i \in V} s_i \leq b_{max} \tag{26}$$

$$s_i, y_{ij} \text{ binary} \tag{27}$$

The link ij can only be selected for the backbone network if both node i and j are part of it. This is ensured by the constraints (23) and (24). Furthermore, inequalities (25) enforces link ij to be part of the backbone network if nodes i and j are selected. Finally, constraint (26) ensures that the bounds on the number of clusters (access networks) are enforced. The pricing problem for the backbone generation subproblem is defined by (22)-(27). A solution approach to this model is described in Section F.3.3

F.3.2 The Cluster generation subproblem

The definition of the cluster generation subproblem is parallel to the approach for the backbone network. Let a_i and x_{ij} be 1 if the node i respectively the link ij is in the cluster, and 0 otherwise. Furthermore the variable s_i is 1 if node i is hub, and 0 otherwise. Now the reduced cost of a cluster is:

$$\sum_{ij \in E} c_{ij} x_{ij} - \sum_{i \in V} \alpha_i a_i + \sum_{i \in V} \beta_i s_i \tag{28}$$

The cluster generation problem seeks the column with the most negative reduced cost, or equivalent, has the objective:

$$\max \sum_{i \in V} \alpha_i a_i - \sum_{i \in V} \beta_i s_i - \sum_{ij \in E} c_{ij} x_{ij} \tag{29}$$

A feasible column (defining an access network) must fulfill:

$$\sum_{i \in V} s_i = 1 \tag{30}$$

$$s_i \leq a_i \qquad i \in V \tag{31}$$

$$x_{ij} \leq a_i \qquad ij \in E \tag{32}$$

$$x_{ij} \leq a_j \qquad ij \in E \tag{33}$$

$$a_i + a_j \leq x_{ij} + 1 \qquad ij \in E \tag{34}$$

$$v_{min} \leq \sum_{i \in V} a_i \leq v_{max} \tag{35}$$

$$a_i, s_i, x_{ij} \text{ binary} \tag{36}$$

Each access network has precisely one hub, which is ensured by equation (30), and the hub has to be part of the access network, which is ensured by (31). Parallel to (23)-(25) for the backbone generation problem, (32)-(34) ensure that a link ij is selected if and only if both nodes i and j are selected. A feasible access network can only have between $v_{\min}$ and $v_{\max}$ nodes i.e. it has to fulfill (35). Thus the pricing problem for the cluster generation is defined by (29)-(36). A solution approach is discussed in the following section.

F.3.3 Solving the subproblems

Instead of solving the problems for backbone and cluster generation directly, it can be observed that both of the subproblems can in fact be solved as a series of quadratic knapsack problems (QKP). For at general description of QKP see [5]. The quadratic knapsack problem seeks to maximize a quadratic objective function subject to a single capacity constraint. If we let the binary variable q_i be equal to 1 if item i is selected and 0 otherwise, and let q_{ij} be 1 if both i and j are selected and 0 otherwise. Finally let p_i be the profit of selecting item i, p_{ij} be the profit of selecting both item i **and** j. Then the QKP can be formulated as:

$$\max \quad \sum_i p_i q_i + \sum_i \sum_{j:i<j} p_{ij} q_{ij} \tag{37}$$

$$\text{s.t.} \quad q_{ij} \leq q_i \qquad i, j \tag{38}$$

$$q_{ij} \leq q_j \qquad i, j \tag{39}$$

$$q_i + q_j \leq q_{ij} + 1 \qquad i, j \tag{40}$$

$$\sum_j w_j q_j \leq C \qquad (41)$$

$$q_{ij}, q_j \text{ binary} \qquad (42)$$

where w_j is the weight of the j'th item and C is the capacity of the knapsack. Constraints (38), (39), and (40) ensure consistency of variables and (41) is the knapsack constraint. A solution approach to the QKP is described in [2]. The approach is based on Lagrangian relaxation and seems to be the current state-of-the-art for exact solution of the QKP. Furthermore the source code is available at the homepage of David Pisinger, see `www.diku.dk/~pisinger`. This approach and the available code is used to solve both subproblems.

Let us first consider our subproblem for the backbone network, (22)- (27). The subproblem bears some resemblance with the QKP. The nodes can be considered items with a weight of 1 and the profit equals the cost of the links in the backbone. The deviation is that both a lower and upper bound exist on the contents of the knapsack. To address this, we take the approach of adding a constant to all coefficients in the objective, such that all coefficients are non-negative. With non-negative coefficients in the objective and weights equal to 1 in the knapsack, the optimal solution to the corresponding QKP, will always fill the knapsack to capacity. Hence, the subproblem can be solved by solving a series of QKP's, one for each of the capacities $C = b_{\min}, b_{\min} + 1, \ldots, b_{\max}$. The algorithm is shown in Figure F.4.

for $b_k = b_{\min}$ to $b_{\max}$ **do**
  Solve QKP for $C = b_k$
  if Solution has a positive value **then**
    add the cluster column to the master problem
  end if
end for

Figure F.4: The backbone generation algorithm.

Similarly to the backbone generation problem, we add a constant to all coefficients in the objective, such that all coefficients are non-negative and solve a problem for each of $C = v_{\min}, v_{\min} + 1, \ldots, v_{\max}$. However, the cluster generation problem poses one additional complication compared to the backbone generation problem. In addition to choosing nodes, one of the nodes has to be designated as the hub node. We handle this by enforcing each of the nodes to be hub, one at a time. This corresponds to fixing each of the s_i variables to 1 in turn. However, fixing a variable to 1 cannot be done directly using the code used to solve the QKP's. Instead, a sufficiently high value is added to the objective coefficient of the node, thus ensuring that the node is selected.

As a consequence, $|V|(v_{\max} - v_{\min} + 1)$ QKP problems have to be solved. The algorithm is shown in Figure F.5.

for $i \in V$ **do**
  for $v_k = v_{\min}$ to $v_{\max}$ **do**
    Solve QKP for $C = v_k$ and s_i forced to 1
    if Solution has a positive value **then**
      add the cluster column to the master problem
    end if
  end for
end for

Figure F.5: The cluster generation algorithm.

The approach described in [2] also contains a greedy heuristic for the QKP. In our generation of subproblems, we run this heuristic on the full series of subproblems before running the exact approach. The exact approach is only used if no columns can be obtained by the heuristic. The QKP is also used as a subproblem in a column generation approach in [4]. Furthermore, [9] uses a similar approach to solve a related two layered network design problem.

F.3.4 Initialization

To initialize the column generation, a number of "dummy" columns for the clustering and the backbone part are generated. The dummy columns for the clustering part consists of one column per node. Each column contains one node with no designated hub, that is, the column contains a single 1 for the i'th row ($a_i^c = 1$).

For the backbone part we also generate one column per node i. These columns have a 1 corresponding to the i'th node being a hub ($s_i^b = 1$), and all remaining coefficients are 0. In order to satisfy (20), an additional "dummy" column is added. It does not contain any nodes but has a coefficient 1 for the constraint (20). All these dummy columns are added to ensure a feasible LP upon branching and they are assigned a value sufficiently high in order to force them out of the basis in the optimal solution.

F.4 Branch-and-Price

As the column generation method described above cannot guarantee integral solutions, it has to be embedded in the Branch-and-Bound framework. The combination of column generation and Branch-and-Bound is often denoted Branch-and-Price or IP column generation [1, 10].

In case the solution of a branch node in the Branch-and-Bound tree is not integer and cannot be fathomed, we branch. Here, we implement the Ryan-Foster branching [7]. We branch on whether node i and j are in the same cluster or not. A constraint enforcing this requirement is added to the master problem. This does, however, not guarantee integer optimality. Two clusters with the same nodes but with different hubs may be selected, each with u_c equal to $\frac{1}{2}$. Now branching on whether node i and j are in the same cluster cannot be applied, yet the solution is not integer. This is handled by branching on whether a node is hub or not, that is, in one branch, node i is forced to be hub, and in the other branch node, i cannot be hub.

In the master problem, the choices taken by the branching strategy results in the addition of constraints. Let $B_1 \subseteq E$ be the set of "nodes are/are not in the same cluster" branches and $B_2 \subseteq V$ be the set of "node i is/is not hub" branches. We define p_b^c for $\{i, j\} = b \in B_1$ to be equal to one if i and j are in the same cluster, otherwise 0. Furthermore, recall that s_i^c is 1 if i is hub in cluster c. So we get:

$$\sum_{c \in C} p_b^c u_c = 0/1 \quad b \in B_1 \tag{43}$$

$$\sum_{c \in C} s_i^c u_c = 0/1 \quad i \in B_2 \tag{44}$$

where (43) with the right-hand side 0 corresponds to forbid clusters containing i and j, and a right-hand side of 1 forces a cluster to contain both i and j. Correspondingly a right-hand side of 0 in (44) means that node i is not hub, and 1 forces i to be hub.

Branching is implemented by first determining the u_c column with fractional values closest to $\frac{1}{2}$. Then, the first row covered (i.e. it has a coefficient of 1) by this column is found. Now we search for another column that has a fractional value and covers the same row. Such a column must exist due to the partitioning constraints for the clusters (17). So either:

1. The columns cover exactly the same rows. This implies that the hub are not identical and we branch on whether the node is hub or not.

2. The columns cover different rows. Now we determine the first row where they differ and branch on whether nodes are in the same cluster or not.

Referring back to the example in Figure F.2, consider the situation where the first two columns are picked with a value of $\frac{1}{2}$ and the two remaining nodes where covered by a single column with value 1. Then the branching approach will detect that both columns define the same cluster (they cover the exact same rows) and therefore branching will force the node represented by the first row (node a) to either be or not be a hub. If instead we had u_c equal to $\frac{1}{2}$ for the first and the third column, the test reveals that the two columns define different clusters and branching will force a and b to be in the same cluster or not.

In the branching tree, a depth first strategy is applied. This enables use of "warm start" in the LP relaxation of the master problem with the previous solution. Having two candidates for branching on the same depth, we choose the one that fixes the right hand side values to 1 in (43) and (44).

The branch constraints (43) and (44) leads to dual variables which needs to be incorporated into the subproblems. Taking these dual variables into account in the subproblem is sufficient, i.e. it is not necessary to force the subproblem to generate columns feasible with respect to a given set of branches. The dual variables δ_b and ϵ_i corresponding to (43) and (44) are only added to the calculation of the reduced cost for a cluster column. No modification of the backbone columns are necessary.

We now modify (28) to reflect that the reduced cost of the columns should include the dual variables of the branch constraints:

$$\sum_{ij \in E} c_{ij} x_{ij} - \sum_{i \in V} \alpha_i a_i + \sum_{i \in V} \beta_i s_i - \sum_{b \in B_1} \delta_b p_b - \sum_{i \in B_2} \epsilon_i s_i \tag{45}$$

where p_b is 1 for $b = (i, j)$, if i and j are in the same cluster.

Therefore, for the cluster generation problem, the objective of the pricing problem (29) is modified to:

$$\max \sum_{i \in V} \alpha_i a_i - \sum_{i \in V} \beta_i s_i - \sum_{ij \in E} c_{ij} x_{ij} + \sum_{b \in B_1} p_b \delta_b + \sum_{i \in B_2} s_i \epsilon_i \tag{46}$$

By noting that $p_b = p_{\{i,j\}} = a_i a_j = x_{ij}$ and rewriting, we obtain:

$$\max \sum_{i \in V} \alpha_i a_i + \sum_{i \in V} (\epsilon_i - \beta_i) s_i + \sum_{ij \in E} (\delta_{ij} - c_{ij}) x_{ij} \tag{47}$$

Thus including the additional dual variables is only a matter of modifing the constants of the objective, and hence can easily be included.

F.5 Experimental Results

We have tested the two bounds and the Branch-and-Price approach on generated instances with n nodes for $n = 10, 15, 20$, and 25. All the graphs are fully connected and two types of instances have been generated. Euclidean instances where the link costs are proportional to the Euclidean distances between the endpoints which have been randomly located in the unit square, and random instances, where the link costs are randomly selected using a uniform distribution. Furthermore $b_{\min}$ and $v_{\min}$ are set to $|\sqrt{n}| - B_d$, and $b_{\max}$ and $v_{\max}$ are set to $|\sqrt{n}| + B_d$. Here we have tested each instance with B_d equal to $1, 2$, and 3.

First we have tested the column generation scheme (CG-FINDP) against the LP relaxation of the FINDP (LP-FINDP) to see which approach produces the tightest bounds. The results are shown in Table F.1 and Table F.2. In the tables, "Gap" is the gap to the known optimal solution, "Iter" denotes the number of iterations, i.e. the number of calls to the subproblems in the column generation algorithm, and "Cols" identifies the number of columns generated.

Problem		LP-FINDP		CG-FINDP			
n	B_d	Seconds	Gap (%)	Seconds	Gap (%)	Iter	Cols
10	1	0.15	52.6	0.47	14.9	8	159
10	2	0.12	61.1	0.88	3.9	9	219
10	3	0.15	69.9	1.05	13.1	9	237
15	1	1.15	31.3	1.84	8.8	17	388
15	2	1.31	60.0	2.32	1.9	14	408
15	3	0.38	73.3	3.25	17.2	13	450
20	1	1.69	57.0	3.62	27.7	21	545
20	2	1.16	65.6	6.14	11.1	17	718
20	3	1.36	76.0	8.24	13.8	14	747
25	1	6.39	27.1	13.30	14.9	24	846
25	2	4.73	56.5	14.58	20.2	21	1272
25	3	5.07	70.8	19.06	15.6	19	1224

Table F.1: The LP relaxation of the FINDP model vs. the column generation approach for a lower bound on the Euclidean instances.

For both types of graphs, it is evident that the column generation approach

Problem		LP-FINDP		CG-FINDP			
n	B_d	Seconds	Gap (%)	Seconds	Gap (%)	Iter	Cols
10	1	0.16	66.5	0.41	5.7	9	163
10	2	0.14	74.4	1.26	4.5	9	194
10	3	0.13	78.5	0.82	4.7	8	222
15	1	1.74	49.6	1.15	5.3	12	294
15	2	1.29	76.0	3.64	1.8	13	397
15	3	0.47	80.4	5.34	5.9	14	487
20	1	1.81	56.8	5.49	1.5	21	599
20	2	1.09	80.2	12.15	9.7	16	636
20	3	1.12	88.9	14.46	7.1	19	697
25	1	6.61	48.5	14.97	6.1	25	763
25	2	4.97	76.2	33.24	4.0	27	1220
25	3	4.64	83.7	36.97	1.8	27	1367

Table F.2: The LP relaxation of the FINDP model vs. the column generation approach for a lower bound on the randomly generated instances.

produces bounds superior to the LP relaxation. On the Euclidean instances the gaps are between 27% and 76% for the LP relaxation, which will make it difficult to obtain an efficient exact approach based on an LP relaxation. In contrast the bound for CG-FINDP are between 1.88% and 28%, and for the random instances, the bounds are even better. Here the largest deviation from the optimal solution is below 10% for the CG-FINDP. For the LP relaxation, the gaps have increased and are in the interval between 48% and 89%. The cost of better bounds is a modest increase in running times. Note that both the number of columns and iterations needed is low. We never need to generate more than 1400 columns and run 27 iterations.

Based on the results above we have only tested an exact approach based on the column generation bound. The results of the tests are presented in Table F.3 and Table F.4. Here the column "BB" displays the number of Branch-and-Bound nodes needed to find the optimal solution, "Cols" and "Iter" denote the number of columns respectively the number of iterations in the column generation process that is needed. Finally, the remaining 4 columns presents the total running time, and then a breakdown into the Master Problem ("MP"), the exact pricing algorithm ("SP opt") and the heuristic pricing algorithm ("SP heu").

The results clearly show that the randomly generated instances are easier to solve than the Euclidean instances. Obviously, the tighter gap plays an important role. Except for the three Euclidean instances with 25 nodes, all running

Problem		BB	Cols	Iter	Seconds			
n	B_d				Total	MP	SP opt	SP heu
10	1	54	441	213	25	0	17	6
10	2	34	540	152	27	0	18	8
10	3	6	346	41	8	0	4	3
15	1	245	1859	1172	391	9	326	52
15	2	78	1201	431	201	2	166	31
15	3	66	1310	398	256	2	207	42
20	1	1621	7226	6228	4796	302	4091	369
20	2	120	2481	679	696	9	610	68
20	3	1006	6971	4324	7551	221	6655	636
25	1	4568	19137	19375	42619	5626	35279	1463
25	2	45839	55692	115849	671346	248690	402661	14474
25	3	4922	18658	16978	71056	4948	62838	2984

Table F.3: Results for the Branch-and-Price for the FINDP on the Euclidean instances

Problem		BB	Cols	Iter	Seconds			
n	B_d				Total	MP	SP opt	SP heu
10	1	4	179	14	1	0	1	0
10	2	2	219	8	2	0	1	0
10	3	2	247	12	3	0	2	1
15	1	15	551	123	40	0	34	5
15	2	54	917	269	177	1	152	19
15	3	120	1547	634	575	4	495	69
20	1	34	1168	215	179	2	169	13
20	2	150	2061	787	1197	10	1093	79
20	3	262	3248	1565	3151	31	2867	231
25	1	45	1697	475	944	6	885	36
25	2	77	2453	510	1720	9	1609	65
25	3	42	2281	367	1565	6	1455	64

Table F.4: Results for the Branch-and-Price for the FINDP on the randomly generated instances

times can be seen as reasonable. It is worth noting that a large fraction of the time spent by our algorithm is spent in the exact SP. In the breakdown of the time usage, the time spent in the exact SP algorithm always accounts for the vast majority of the running time. Most of the problems, including all randomly generated instances, are solved generating only a few thousand columns.

The exact SP algorithm can be replaced by a single call to a MIP solver, as a consequence producing at most one column. Since the heuristic SP produces most columns, the call to the exact SP procedure is often just to check that all columns have been generated. Thus in interplay with the heuristic, this could be faster. Initial computational experiences support this.

F.6 Conclusion

The contribution of this paper is the development of two different models (a mathematical model and one based on column generation) and an exact solution approach for a two-layered network design problem. The problem is defined by using a fully interconnected topology both for the access networks and the backbone network.

Our computational experiments are based on two sets of instances, one randomly generated and one using Euclidean distances. The results show that the bound based on column generation is superior to the LP relaxation of the mathematical model. The gaps are often more than a factor 10 worse on the LP relaxation. It seems impossible to base an efficient exact approach on the LP relaxation. The bounds on the column generation approach are tight enough – especially on the random instances – to develop an optimal approach, even though this bound is more time consuming to compute than the LP relaxation.

The optimal method is able to solve all randomly generated instances within one hour. The bounds on the Euclidean instances are worse than for the randomly generated instances, which is also reflected in the running times. For the Euclidean instances 5 out of the 12 instances cannot be solved within one hour – one instance takes almost 8 days to solve. It is noteworthy that most of our problems are solved generating only a few thousand columns.

We believe that further improvements can be obtained by proving optimality of the subproblems solving the pricing problems directly in a MIP solver instead of solving a series of QKP's. Furthermore the running times on especially the Euclidean instances suggests research in heuristics based on the optimal method. Feasible solutions obtained by such heuristics can be used to speed up the Branch-and-Price algorithm.

Bibliography

[1] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance. Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46(3):316–29, 1998.

[2] A. Caprara, D. Pisinger, and P.Toth. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11(2):125–137, 1999.

[3] A.T. Ernst and M. Krishnamoorthy. Efficient algorithms for the uncapacitated single allocation p-hub median problem. *Location Science*, 4(3):139–154, 1996.

[4] E.L. Johnson, A. Mehrotra, and G.L. Nemhauser. Min-cut clustering. *Mathematical Programming*, 62(1):133–151, 1993.

[5] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.

[6] J.G. Klincewicz. Hub location in backbone/tributary network design: a review. *Location Science*, 6:307–335, 1998.

[7] D.M. Ryan and B.A. Foster. An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport Urban Passenger Vehcile and Crew Scheduling*, 269–280, North Holland, Amsterdam, 1981.

[8] D. Skorin-Kapov, J. Skorin-Kapov and M. O'Kelly. Tight linear programming relaxations of uncapacitated p-hub median problems. *European Journal of Operational Research*, 94(3):582–593, 1996.

[9] T. Thomadsen. Design of Two-Layered Fixed Charge Networks. *Work in progress*, 2005.

[10] F. Vanderbeck, L.A. Wolsey. An exact algorithm for IP column generation. *Operations Research Letters* 19(4):151–159, 1996.

Appendix G

Design of Two-Layered Fixed Charge Networks

Submitted for *Networks*

Preliminary version appears in Proceedings of the International Network Optimization Conference(INOC), 2005, Lisbon.

Design of Two-Layered Fixed Charge Networks

Tommy Thomadsen[1]

Abstract

This paper considers design of two-layered meshed networks. A two-layered meshed network consists of clusters of nodes comprising the access network layer and a backbone layer interconnecting the clusters. Designing a two-layered meshed network involves a number of interrelated problems: Hub location, clustering of nodes, interconnection of nodes, and routing. In this paper these problems are all solved jointly. A mathematical model is presented and two lower bounds are derived. One is the straightforward linear programming relaxation and the other is obtained by reformulating the problem into a model suitable for column generation. The column generation subproblems are solved by solving a series of quadratic knapsack problems. Furthermore, a heuristic is implemented to obtain feasible solutions. Results are presented which computationally evaluates the quality of the lower bounds versus the quality of the heuristic solutions. The column generation lower bound is always the best of the two bounds, though differences are small. The gap between the heuristic solution and the bound is up to 9% and the majority of this gap seems to be due to the quality of the bounds. On the other hand, the heuristic performs quite well. Especially in light of the difficulty of the problem, the results are encouraging.

Keywords: Hub location, Node Clustering, Backbone/Access Network Design, Hierarchical Networks, Meshed Networks.

G.1 Introduction

Communication networks usually have a hierarchical structure composed of two or more layers. A hierarchical structure has proved beneficial to cope with changing traffic demands and regular upgrades. When designing such hierarchical or layered networks, interrelated problems have to be solved. For instance, the location of hubs, clustering of nodes, interconnection of nodes, and routing

[1]Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark. Email: tt@imm.dtu.dk

are problems which should be solved. These problems have often been considered independently or only a few of them have been considered simultaneously. Since the problems are interrelated, this may lead to suboptimal designs.

This paper considers *joint* hub location, clustering of nodes and network design (establishing the links and route the demands) of two-layered meshed networks. The two layers are denoted the backbone network and the access network. The backbone network connects disjoint clusters of nodes comprising the access network. An example is given in Figure G.1.

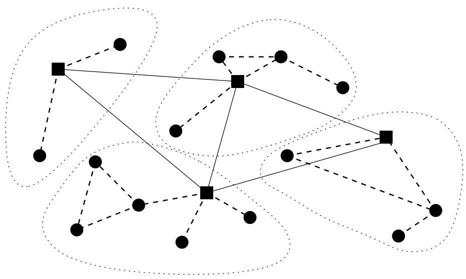


Figure G.1: *An example of a two-layered meshed network.*

In the figure, the squares are the hub nodes or the backbone nodes and the solid lines are the links of the backbone network. The circles and the dashed lines are the access network nodes and links. The clusters are indicated, and each cluster consists of a hub node, some access network nodes and some access network links. The backbone consists of all hubs and backbone links interconnecting the hubs. The objective is to minimize link costs consisting of a fixed cost and a capacity cost and at the same time satisfy the communication demand. This is denoted the hub location, clustering and network design (HLCND) problem.

Problems that combine hub location, clustering and network design are reviewed in [9]. Common to all papers are that they study networks where either the backbone or the access networks (usually both) have special structure; either a path, star, tree, ring, or fully interconnected. A recent paper on the topic is [6] which considers the design of two-layered networks where both the backbone and the access networks are trees. Also [2] studies a problem where the backbone network is either a tree or a ring and the access network consist of stars. In the paper, node costs depending on the equipment installed are taken into account. The problem when the backbone is a ring and the access network consists of stars is considered in [10]. The authors present an exact algorithm for the problem. Finally [12] considers a problem, where a fixed charge backbone network has to be determined, given the clusters.

Several papers have addressed fixed charge network design. An early paper is [11]. More recent papers are e.g. [3] and [7], which both consider Lagrangian methods for the network design problem. A review of work on capacitated network design can be found in [5]. The HLCND generalizes the fixed charge network design problem, since the backbone and each of the clusters of the access network consists of fixed charge networks.

This paper considers design of two-layered networks, i.e. *joint* hub location, clustering of nodes and network design, where the clusters and the backbone are fixed charge networks. This is in contrast to other papers considering combined hub location, clustering and network design, since all these papers enforce a special structure on either the backbone, the clusters, or both. Furthermore, routing is most often not included in previously considered problems, whereas this is included in this paper.

The rest of the paper is organized as follows. Section G.1 presents a mathematical model for the HLCND problem. The linear programming relaxation of the model is computed by means of cutting plane, which is presented in Section G.3. An alternative model is investigated in Section G.4. Column generation is used for solving the linear programming relaxation of this model. In order to obtain feasible solutions a heuristic is developed, which is described in Section G.5. Section G.6 presents computational results and finally Section G.7 gives some concluding remarks.

G.2 A Mathematical Model for the HLCND Problem

Consider the graph $G = (V, E)$, where V is the set of nodes and E is the set of undirected edges representing the links. Let A be the set of directed arcs corresponding to E, such that for each edge $(ij) = e$ in E, two arcs exist, one in each direction. For edge e, the arcs are denoted $e_1 \in A$ and $e_2 \in A$. Furthermore D is a set of communication demands. The terms $s(a)$ and $t(a)$ are used to denote the start and terminal node of arc a in A. For d in D, $O(d)$ and $D(d)$ denote the origin and the destination node, respectively.

Let f_e denote the fixed cost for an edge e in E and c_a denote the capacity cost per unit of communication demand for an arc a in A. Let b_d denote the units of communication demanded corresponding to demand d in D. Furthermore let c_{min} and c_{max} be a lower and an upper bound on the number of clusters and similarly let v_{min} and v_{max} be a lower and an upper bound on the number of nodes in each cluster.

The model has four types of variables, y_e, x_a^d, h_i, and z_{ij}. The variable y_e, e in E is 1 if edge e is used, 0 otherwise. The variable x_a^d, e in E, a in A denotes the fraction of the demand d routed on arc a. Also, h_i, i in V is 1 if node i is a hub, 0 otherwise and z_{ij}, i and j in V, $i < j$ is 1 if nodes i and j are in the same cluster, 0 otherwise. For notational convenience z_{ji} is sometimes used instead of z_{ij}. The model for the HLCND problem is then:

$$\min \sum_{e \in E} f_e y_e + \sum_{d \in D, a \in A} b_d c_a x_a^d \tag{1}$$

$$\text{s.t.} \quad \sum_{a \in A, s(a)=i} x_a^d - \sum_{a \in A, t(a)=i} x_a^d = \begin{cases} 1 & \text{if } i = O(d) \\ -1 & \text{if } i = D(d) \\ 0 & \text{otherwise} \end{cases} \quad \forall d \in D, i \in V \tag{2}$$

$$x_{e_1}^d + x_{e_2}^d \leq y_e \qquad \forall e \in E, d \in D \tag{3}$$

$$y_e \leq z_{ij} + h_k \qquad \forall e = (i,j) \in E, k \in \{i,j\} \tag{4}$$

$$h_i + h_j + z_{ij} \leq 2 \qquad \forall i \in V, j \in V, i < j \tag{5}$$

$$z_{ik} + z_{jk} \leq z_{ij} + 1 \qquad \forall i,j,k \in V, i < j, k \neq i, k \neq j \tag{6}$$

$$v_{min} - 1 \leq \sum_j z_{ij} \leq v_{max} - 1 \qquad \forall i \in V \tag{7}$$

$$c_{min} \leq \sum_i h_i \leq c_{max} \tag{8}$$

$$x_a^d \in [0,1] \quad y_e \in \{0,1\} \tag{9}$$

$$z_{ij} \in \{0,1\} \quad h_i \in \{0,1\} \tag{10}$$

The objective (1) is the sum of the fixed cost and of the capacity cost. Constraints (2) are the flow conservation constraints. They ensure, that for each demand the inflow equals the outflow at intermediate nodes, the net outflow is 1 at the origin node and the net inflow is 1 at the destination node. Constraints (3) ensure that if any demand are routed on a link in either direction, then the link is in use and the fixed cost for the link is accounted for in the objective. Together with (9), constraints (1), (2) and (3) describe a fixed charge network design problem [11].

The remaining constraints enforce the selection of hubs and clustering of nodes. If a link between two nodes i and j are used, then the nodes are either in the same cluster or both nodes are hub nodes, which is ensured by constraints (4).

Each cluster can only contain one node, which is ensured by constraints (5) by prohibiting solutions for which two nodes are both hubs and in the same cluster. Constraints (6) ensure that if nodes i and k are in the same cluster, and nodes j and k are in the same cluster, then nodes i and j are in the same cluster as well. Together with constraints (10), constraints (4), (5), and (6) ensure that the nodes are in disjoint clusters containing one hub node and links either connect nodes within clusters or hub nodes. Constraints (7) enforce bounds on the number of nodes in each cluster and similarly constraint (8) enforce bounds on the number of clusters.

HLCND can be solved by applying a standard MIP solver. It is, however, only possible to find optimal solutions for up to 12 nodes in less than 1 hour of computation time. Even finding the value of the Linear Programming Relaxation (LPR) takes close to 1 hour for networks with 30 nodes and requires gigabytes of memory.

Instead of solving HLCND using a MIP solver, a Greedy Randomized Adaptive Search Procedure (GRASP) [4] is used for obtaining feasible solutions. However, the value of the LPR is useful for evaluating the quality of the heuristic solutions. In order to obtain the value of the LPR for larger instances, we develop a cutting plane algorithm. Also with the purpose of improving the value of the LPR, we investigate a reformulation of the model.

G.3 The Cutting Plane Algorithm

Given the problem structure, a substantial number of x variables are zero in an optimal solution. To reduce the computation time spent on obtaining a lower bound, the cutting plane algorithm does not include all x variables initially but price in variables when needed.

The initial problem solved contains all y, t and z variables and constraints (7) and (8). In addition a subset of the x variables and a subset of the constraints (2) and (3) are included. These subsets of variables and constraints are selected such that a feasible solution always exists. This is done by constructing a feasible solution, and including all nonzero x variables. In addition all constraints of type (2) and (3) that contain a nonzero variable are included.

During each iteration, the reduced cost of the x variables are evaluated. If a variable has a negative reduced cost, it is added. Similarly, the constraints (2), (3), (4), (5), and (6) are evaluated during each iteration. All constraints that are violated are added. Following this, the problem is resolved. This continues until no variables or constraints are added.

In addition some constraints of the following type are added. Consider a subset S of V and let $T \subset E$ be a spanning tree on S. Then the following constraints are valid.

$$\sum_{i \in S} h_i + \sum_{(i,j)=e \in T} z_{ij} \leq |S| \tag{11}$$

This is a generalization of (5). For $|S|$ equal to 3, these constraints are used in the cutting plane algorithm. During each iteration, the constraints are evaluated and, if violated, added.

G.4 A Column Generation Model

In this section an alternative model of the HLCND problem is considered. A variable exists for each possible cluster and each possible backbone. Since there are an exponential number of these variables, the model is solved using column generation. To explain the model, a substantial number of constants and variables need to be defined and thus a short example is given to ease understanding.

The paper [8] solves a clustering problem by using column generation in much the same way as we suggest for the HLCND. The subproblem in the column generation approach suggested in [8] is the quadratic knapsack problem. In case of the HLCND, it turns out that the subproblems in the column generation approach can be solved by solving a series of quadratic knapsack problems. Thus, methods developed for the quadratic knapsack problem are used.

Let C be the set of all feasible clusters with a hub selected and B be the set of all backbone networks, i.e. each backbone is a set of hubs. For the clusters, three types of constants are defined, a_i^c, a_e^c, and h_i^c. The constant a_i^c is 1 if node i is in cluster c, otherwise 0 and a_e^c is 1 if both endpoint nodes of edge e are in cluster c, otherwise 0. The constant h_i^c is 1 if node i is hub in cluster c, otherwise 0.

Two types of constants are defined for the backbone, a_i^b and a_e^b. Similarly to the cluster variables, a_i^b is 1 if node i is in backbone b, otherwise 0 and a_e^b is 1 if both endpoint nodes of edge e are in backbone b, otherwise 0.

Figure G.2 gives an example of a network with clusters and hubs indicated. The two clusters give rise to two columns, denoted c_1 and c_2, and similarly the two hubs correspond to a backbone network column denoted b_1.

For cluster c_1, the constants $a_1^{c_1}$, $a_2^{c_1}$ and $a_3^{c_1}$ are 1, since nodes 1, 2, and 3 are

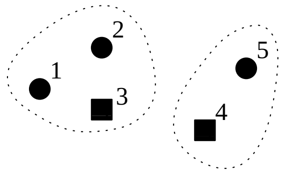


Figure G.2: *An example with clusters and hubs indicated.*

in the cluster. The constants $a_{12}^{c_1}$, $a_{13}^{c_1}$ and $a_{23}^{c_1}$ are 1 since these links may be in the cluster. Finally $h_3^{c_1}$ is 1, since node 3 is hub. Remaining constants for cluster c_1 are 0.

For cluster c_2, the constants $a_4^{c_2}$, $a_5^{c_2}$ are 1, since nodes 4 and 5 are in the cluster. The constant $a_{45}^{c_2}$ is 1, since the link between 4 and 5 may be in the cluster. Finally $h_4^{c_2}$ is 1, since node 4 is hub. Remaining constants for cluster c_2 are 0.

For the backbone b_1, $a_3^{b_1}$ and $a_4^{b_1}$ are 1 since nodes 4 and 5 are in the backbone. Also $a_{34}^{b_1}$ is 1 since node 3 and 4 are in the backbone, and remaining constants for the backbone are 0.

Note that in defining the link constants, there is no indication as to whether a link is actually selected. The definition only considers whether the endpoint nodes of a link is in the cluster. In the example, the three links in cluster c_1 are possible to select, thus the corresponding constants are one, but when the network is designed they may not be included in the optimal solution.

Two types of variables are used, one type for the clusters and one for the backbone. Let u_c, c in C be 1 if cluster c is selected, 0 otherwise and v_b, b in B be 1 if backbone b is selected, 0 otherwise. The model for the HLCND problem is then as given in the following.

$$\min \sum_{e \in E} f_e y_e + \sum_{d \in D, a \in A} b_d c_a^d x_a^d \tag{12}$$

s.t. (2), (3) and (9)

$$\sum_{c \in C} a_i^c u_c = 1 \qquad i \in V \qquad (\alpha_i) \tag{13}$$

$$-\sum_{c\in C} h_i^c u_c + \sum_{b\in B} a_i^b v_b = 0 \qquad i \in V \qquad (\beta_i) \tag{14}$$

$$\sum_{c\in C} a_e^c u_c + \sum_{b\in B} a_e^b v_b \geq y_e \qquad \forall e \in E \qquad (\gamma_e) \tag{15}$$

$$\sum_{b\in B} v_b = 1 \qquad (\delta) \tag{16}$$

$$u_c \in \{0,1\} \quad v_b \in \{0,1\} \tag{17}$$

The core of the formulation is the three constraints (13), (14), and (15). Constraints (13) ensure that each node is in exactly one cluster and constraints (14) ensure that nodes that are hub nodes of a selected cluster are in the backbone network. In addition to selecting clusters and the backbone, it has to be ensured, that links are only established between nodes within clusters or in the backbone. This is ensured by Constraints (15). Finally constraint (16) ensures that exactly one backbone network is selected. This constraint is not necessary but strengthens the linear programming relaxation. The dual variables corresponding to each type of constraint are indicated in brackets after the constraints.

Two column generation problems are now apparent, one for the cluster columns, u_c and one for the backbone columns, v_b. These are described in the following.

G.4.1 The Cluster Generation Problem

For an optimal solution and thus a set of values for the dual variables, the reduced cost of a u_c column is the following.

$$-\sum_{i\in V} \alpha_i a_i^c + \sum_{i\in V} \beta_i h_i^c - \sum_{ij\in E} \gamma_e a_e^c \tag{18}$$

The column generation subproblem seeks the column with the most negative reduced cost, or equivalently, has the following objective:

$$\max \sum_{i\in V} \alpha_i a_i - \sum_{i\in V} \beta_i h_i + \sum_{ij\in E} \gamma_e a_e \tag{19}$$

In this objective, a_i, a_e, and h_i are binary variables representing whether node i is in the cluster, edge e is in the cluster, and node i is hub, respectively. The following constraints ensure that feasible clusters are generated.

$$\sum_{i\in V} h_i = 1 \tag{20}$$

$$v_{min} \leq \sum_{i \in V} a_i \leq v_{max} \tag{21}$$

$$h_i \leq a_i \quad i \in V \tag{22}$$

$$a_e \leq a_i \quad e \in E, i \text{ endpoint of } e \tag{23}$$

$$a_i + a_j \leq a_e + 1 \quad e \in E, (i, j) = e \tag{24}$$

$$a_i \in \{0, 1\}, h_i \in \{0, 1\}, a_e \in \{0, 1\} \tag{25}$$

Exactly one hub has to be determined, which is ensured by constraint (20). Furthermore, constraint (21) ensure that bounds are enforced on the number of nodes in the cluster. Consistency between the variables is ensured by the remaining constraints. Constraints (22) ensure that a node is selected if it is the hub. The connection between a_i and a_e is established by constraints (23) and (24), and finally constraints (25) ensure that all variables are integer.

Instead of solving the problem directly, a series of problems are solved. This has the beneficial side-effect that more than one column is usually being generated per iteration. The h_i variables are fixed one at a time and the bounds on the cluster size is fixed to each of $[v_{min}, v_{min} + 1, \ldots, v_{max}]$ in turn. Thus in total $|V|(v_{max} - v_{min} + 1)$ problems are solved.

Fixing these values has the consequence that one node is mandatory (and the hub) and the number of nodes in the cluster is fixed. Assume the node fixed to be hub is denoted k and that the cluster size is fixed to v_k. Given this, the problem that has to be solved is the following:

$$\max \sum_{i \in V} \alpha_i a_i - \beta_k + \sum_{ij \in E} \gamma_e a_e \tag{26}$$

$$\text{s.t.} \quad \sum_{i \in V} a_i = v_k \tag{27}$$

$$a_k = 1 \quad i \in V \tag{28}$$

$$a_e \leq a_i \quad e \in E, i \text{ endpoint of } e \tag{29}$$

$$a_i + a_j \leq a_e + 1 \quad e \in E, (i, j) = e \tag{30}$$

$$a_i \in \{0, 1\}, a_e \in \{0, 1\} \tag{31}$$

Except for constraint (28) and the fact that constraint (27) is an equation rather than an inequality, this problem is the quadratic knapsack problem(QKP) [1]. However, by replacing a_k by a sufficiently high constant, by adding a sufficiently high value to all constants, such that all constants are non-negative, and by replacing constraint (27) by an inequality, the problem can be solved as a QKP. The code developed and described by Caprara, Pisinger, and Toth in [1] has been used to do the computations. The generated cluster columns are added if the reduced cost is negative, which corresponds to that the objective value is positive. The algorithm used to generate clusters is shown in figure G.3.

for $h \in V$
  for $v_k = v_{min}$ to v_{max}
    Solve the QKP with v_k as bound and h mandatory.
    If the solution has a positive value
      add the corresponding cluster column.

Figure G.3: *The cluster generation algorithm.*

G.4.2 The Backbone Generation Problem

Similarly to the cluster generation problem, the following model is obtained for generation of backbone columns. In the model, a_i and a_e are binary variables representing whether node i is in the cluster and edge e is in the cluster, respectively.

$$\max \sum_{i \in V} \beta_i a_i + \sum_{e \in E} \gamma_e a_e + \delta \tag{32}$$

$$\text{s.t.} \quad c_{min} \leq \sum_{i \in V} a_i \leq c_{max} \tag{33}$$

$$a_e \leq a_i \quad e \in E, i \text{ endpoint of } e \tag{34}$$

$$a_i + a_j \leq a_e + 1 \quad e \in E, (i,j) = e \tag{35}$$

$$a_i \in \{0,1\}, a_e \in \{0,1\} \tag{36}$$

Constraint (33) enforces bounds on the number of nodes. Consistency between variables is established by constraints (34) and (35). Finally constraints (36) ensure that the variables are integer.

The bounds on the number of nodes in the backbone is fixed to $[c_{min}, c_{min} + 1, \ldots, c_{max}]$ in turn. Each of the $c_{max} - c_{min} + 1$ problems are then solved as

QKPs. The generated backbone columns are added if the reduced cost is negative, corresponding to that the objective is positive. The backbone generation algorithm is shown in figure G.4.

```
for c_k = c_min to c_max
   Solve the QKP with c_k as bound.
   If the solution has a positive value
      add the corresponding backbone to the master.
```

Figure G.4: *The backbone generation algorithm.*

Solving the QKPs optimally is time consuming. However, since good heuristics exist for the QKP, these are used first, and when no heuristic solutions can be obtained, the optimal method is invoked. This is done for both the cluster and backbone generation algorithm. The heuristic developed in [1] is used.

In addition to generation of the cluster and backbone columns, the x variables are priced in the same manner as for the cutting plane algorithm described in Section G.3. Also, similarly to the cutting plane algorithm, all of the constraints (2) and (3) are not included initially, but added when needed.

G.5 The GRASP

The GRASP which provides feasible solutions is shown in figure G.5. For a general introduction to GRASP, see e.g. [4].

```
BEST ← infinity
for a fixed number of iterations do
   CURRENT ← Construct a feasible solution
   while a better neighbor solution exists.
      CURRENT ← any neighbor solution better than CURRENT
   if CURRENT better than BEST
      BEST ← CURRENT
```

Figure G.5: *The GRASP for HLCND.*

The GRASP consists of a construction phase and a local optimization phase that is run a fixed number of times and maintains the best known feasible solution.

In the computational results presented, 100 iterations are carried out. The construction phase consists of a randomized greedy algorithm described in the following. A feasible solution is a subdivision of the nodes in the network into clusters and a selection of one hub node in each cluster, such that the number of nodes in each cluster and the number of clusters are within bounds.

G.5.1 The Construction Phase

The construction phase is a randomized greedy algorithm, i.e. during construction, the best decision is not necessarily made, but for a fixed k, *one of* the k best decisions are made. The construction phase merges sets of nodes into feasible clusters. Merges that result in too large clusters are never considered. Initially each node is equivalent with one set. Sets are then merged based on an estimate of the probability that two nodes are in the same set. This is continued until feasible clusters are obtained. In the event that a situation occur, in which no feasible clusters can be obtained, the construction is restarted.

For node pairs, two different estimates of the probability that two nodes are in the same set are considered. This leads to two different algorithms denoted GRASP_dist and GRASP_LPR. The first algorithm is based on the distance between nodes whereas the second is based on the value of the z variables in the LPR solution. The estimate used for two *sets* of nodes A and B, where either or both of the sets have more than one node, is the average over all node pairs such that one node is in A and the other node is in B.

In all the computational results reported, k is 10, i.e. one of the 10 best merges, selected randomly, are carried out. To speed up construction, the estimate is maintained for current pairs of sets. Each time a merge is accomplished, the maintained estimates are updated, which can be done in linear time for *all* the maintained estimates. This has to be compared with the quadratic time necessary to compute *each* estimate, given estimates for pairs of *nodes* only.

Finally hubs have to be selected in each cluster, which is done randomly based on estimates for each node that this is hub. In GRASP_dist, the estimate is based on the distance to the center of the network. In GRASP_LPR, the estimate is based on the values of the h variables in the LPR solution.

G.5.2 The Local Optimization Phase

The local optimization phase investigates solutions which are neighbor solutions to the constructed solution. Two types of neighbor solutions are considered. The first consists of moving one node from one cluster to another. The second consists of selecting an alternative hub within a cluster.

All neighbor solutions of the first type are considered and accepted if better than the current solution. Then the neighbor solutions of the second type are considered. This is continued until no improving neighbor solutions exists.

G.5.3 Evaluation of a Solution

Whenever a feasible solution is constructed or obtained by local optimization, it has to be evaluated, i.e. the cost of the network has to be computed. For a feasible solution, the clusters and the hubs are determined, thus the evaluation consists of determining the links and route the demands. Note that, since each cluster contains exactly one hub, the demands from a node in a cluster to any other cluster has to go through the hub node. Thus routes and links in a cluster can be determined independently of the backbone and other clusters. Similarly, given the clusters and hubs, it is obvious which demands runs through the backbone and from and to which hub nodes. Thus the backbone network can be determined independent of the links and routes in clusters.

For each cluster (and similarly for the backbone), several demands run from the same node to the hub node in the cluster. To ease computation, it is beneficial to aggregate demands with the same start and end node. Evaluating a feasible solution, then amounts to solving a fixed charge network design problem for all clusters and the backbone independently. Since the number of nodes in each cluster and in the backbone is low compared with the overall network, it is possible to find the optimal solution using a MIP solver. It is, however, by far the most computational intensive part of the GRASP.

As mentioned, the optimal route within a cluster is independent of how the other clusters and the backbone are designed. Furthermore, a substantial number of solutions are constructed, and locally optimized. Thus, some clusters will inevitably reappear, and in that case previously computed values for the clusters can be reused. To reduce the time spent on solving fixed charge network design problems, the computed values for each cluster is recorded and reused.

Furthermore, since a number of solutions are constructed and locally optimized, entire solutions that have been evaluated previously reappear. Thus values of entire solutions identified by the clusters and hubs are recorded and reused whenever possible.

G.6 Computational Results

For evaluating the algorithms, test instances are generated by locating nodes in the unit square. A link exists between all pairs of nodes and the fixed costs and capacity costs are proportional to the Euclidean distance. Uniformly distributed demands are generated for each pair of nodes. The ratio between the fixed cost and the capacity cost is kept constant and for the results presented equal to the average demand per link multiplied by $0.2|V|$. This leads to feasible solutions with an average node degree of approximately 2-2.5, which corresponds to solutions with clusters consisting of mainly trees and with a backbone that is usually two-connected. This seems reasonable from a practical point of view.

In addition, values for c_{min}, c_{max}, v_{min} and v_{max} are generated. This is done by providing the value v_{free}, which is a nonzero integer. The value v_{free} can be interpreted as how "freely" the clusters and backbone can be determined, i.e. how many nodes they contain. Define v_{center} to be $\sqrt{|V|}$ rounded to nearest integer. Then $v_{min} = v_{center} - v_{free}$ and $v_{max} = v_{center} + v_{free}$. Given this, $c_{min} = \left\lceil \frac{|V|}{v_{max}} \right\rceil$ and $c_{max} = \left\lfloor \frac{|V|}{v_{min}} \right\rfloor$. As an example, when v_{free} takes the value 1, this corresponds to very tight bounds on the number of nodes in the clusters and in the backbone and correspondingly, large values of v_{free} gives loose bounds. Only instances with v_{free} fixed to 1 and 2 are considered.

Table G.1 presents the computational results for the lower bounds.

The first two columns identify the problem by the number of nodes and v_{free}. The third column shows the time required to compute the optimal solution. The remaining columns show the relative gap to the best known solution and the time required to compute the lower bounds. The best known solution is the optimal solution if it is known, and otherwise the best of the solutions produced by the two GRASPs. The optimal solution is computed by branch-and-bound using the column generation model for bounding. Note that in the column generation model, the cluster and backbone column variables are integer if the y variables are integer. Thus branching is on the y variables.

The gaps are substantial, but in all cases the column generation model bound is the same or lower than the LPR bound. However, the larger the instances are, the closer the bounds are to each other. The time required to compute

Problem		Optimal	LPR		Col. Gen. Model	
$\|V\|$	v_{free}	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)
10	1	232	3.13	1	1.94	1
12	1	2306	3.21	1	2.60	4
15	1	147523	7.07	4	6.55	10
	2	3266	2.67	3	2.23	10
20	1		5.46	27	5.30	68
	2		3.86	25	3.77	56
25	1		6.06	125	5.98	379
	2		4.07	113	4.05	356
30	1		6.53	503	6.50	988
	2		6.21	438	6.19	999
40	1		8.02	8322	8.02	10060
	2		6.64	5228	6.64	15351
50	1		7.44	107634	7.44	43957
	2		7.38	102995	7.38	50571

Table G.1: *Computational results for the lower bounds.*

the bounds are substantial, an in the worst case close to 30 hours. However, recall that the number of x variables in the formulation is $O(|V|^4)$. Thus, the instances seems deceptively small from the node count, but they are quite large in terms of variables and constraints. The high computation times are therefore not surprising.

Table G.2 presents the computational results for the GRASP.

The first two columns identify the problem. The remaining columns show the relative gap to the best of the two lower bounds and the optimal solution, if known, and the time required to compute the heuristic solutions. GRASP_LPR requires the solution of the LPR, thus in addition to the reported time, the time required to compute the LPR is used.

For the four smallest instances, for which the optimal solution is known, the gap is less than 1%. Note that for the same four instances, the gap for the lower bounds in Table G.1 is much worse. Thus, the majority of the gap is due to the quality of the lower bound, whereas the heuristics seems to produce quite good solutions. The gap is in the worst case 9%, but since this is presumably due to the lower bounds, it is more interesting to compare the two GRASPs.

Problem		GRASP_dist		GRASP_LPR	
$\|V\|$	v_{free}	Gap(%)	Time	Gap(%)	Time
10	1	0.00	4	0.00	5
12	1	0.00	12	0.64	12
15	1	0.00	31	0.00	31
	2	0.29	38	0.19	46
20	1	5.81	76	5.59	74
	2	3.92	293	4.91	410
25	1	6.36	124	6.36	125
	2	4.22	234	5.39	293
30	1	6.95	545	7.15	547
	2	6.69	1288	6.60	1844
40	1	8.72	1977	8.72	2315
	2	7.11	4113	7.92	6261
50	1	9.00	4873	8.03	4902
	2	7.97	7528	8.56	9115

Table G.2: *Computational results for the GRASP.*

GRASP_dist and GRASP_LPR obtain comparable results which are never more than 1% from each other. Neither seems better than the other. The computational times are comparable with the mentioned caveat that GRASP_LPR requires the evaluation of the LPR. Thus if no bound is required, GRASP_dist is preferable.

The value of v_{free} affects the quality of the lower bound versus the heuristic solution and the computational time of the GRASPs. For the two instances with 15 nodes, for which the optimal solution is known, the quality of the lower bound is best for $v_{free} = 2$. In general the gap between the heuristic solutions and the lower bound is smallest for $v_{free} = 2$. Also notable is, that instances with $v_{free} = 1$ require less computation time than instances with $v_{free} = 2$. This can be explained by the observation, that for $v_{free} = 2$, larger clusters and backbones are allowed compared with $v_{free} = 1$. When evaluating solutions, fixed charge network design problems are solved using a MIP solver, and since the computation time is highly dependent on the size of the fixed charge network design problems, the overall computational time increases.

G.7 Conclusion

The major contribution of this paper is to investigate the hub location, clustering and network design problem. The problem consists of determining a two-layered fixed charge network, which includes hub location and clustering of nodes. The model and the reformulated model suitable for column generation are both new and are used to obtain lower bounds. The GRASP developed produce acceptable solutions.

The problem generalizes fixed charge network design and furthermore is related to other problems which address the design of two-layered networks. The problem presented here is much more difficult than most related problems, since in these a special structure is enforced on either the backbone network, the cluster networks, or both, which ease computations.

Computational experiments show that the bound based on column generation is better than the straightforward linear programming relaxation for small instances. The gaps between the feasible solutions and the lower bounds are, however, up to 9%. The majority of the gap seems to be due to the bounds, whereas the heuristic seems to produce quite good feasible solutions. The heuristic solutions are less than 1% from the optimal solutions for the few small examples where an optimal solution has been obtained.

The GRASP produces solutions for the hub location, clustering and network design problem, and since bounds are presented, we can computationally evaluate how well the GRASP is performing. However, if an optimal method is to be developed, the bounds should be improved both in terms of the quality of the bound achieved and in terms of running time. Other authors have previously solved fixed charge network design problems using Lagrangian methods, so that seems as an obvious path to follow. The complexity of the problem, nevertheless, implies that the largest instances will be very difficult to solve to optimality. At the very least, using Lagrangian methods could decrease the computation time required to obtain the bound and hopefully improve the bound. The GRASP seems to produce rather good solutions, however, the computation time should be decreased. Thus, it may be interesting to consider alternative heuristics.

Bibliography

[1] Caprara, A., Pisinger, D., Toth, P. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, Vol. 11, No. 2, 125–137, 1999.

[2] Chamberland, S., Sansò, B., Marcotte, O. Topological design of two-level telecommunication networks with modular switches. *Operations Research*, Vol. 48, No. 5, 745–60, 2000.

[3] Crainic, T. G., Frangioni, A., Gendron B. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, Vol. 112, No. 1-3, 73–99, 2001.

[4] Feo, T. A., Resende M. G. C. Greedy Randomized Adaptive Search procedures. *Journal of Global Optimization*, Vol. 6, 109–133, 1995.

[5] Gendron, B., Crainic, T. G., Frangioni, A. Multicommodity Capacitated Network Design. *Telecommunications Network Planning*, Sansò, B., Soriano, P. ed. Kluwer, 1-19, 1999.

[6] Gouveia, L., Telhada, J. An Augmented Arborescence Formulation for the Two-Level Network Design Problem. *Annals of Operations Research*, Vol. 106, No. 1-4, 47–61, 2001.

[7] Holmberg, K., Yuan, D.A Lagrangean approach to network design problems *International Transactions in Operational Research*, Vol. 5, No. 6, 529–539, 1998.

[8] Johnson, E.L., Mehrotra, A., Nemhauser, G.L. Min-cut clustering *Mathematical Programming*, Vol. 62, No. 1, 133–151, 1993.

[9] Klincewicz, J.G. Hub location in backbone/tributary network design: a review. *Location Science*, Vol. 6, 307–335, 1998.

[10] Labbe, M., Laporte, G., Martin, I.R., Gonzalez, J J.S. The Ring Star Problem: Polyhedral analysis and exact algorithm. *Networks - Bognor Regis*, Vol. 43, No. 3, 177–18, 2004.

[11] Magnanti, T.L., Wong, R.T. Network design and transportation planning: models and algorithms. *Transportation Science*, Vol. 18, No. 1, 1–55, 1984.

[12] Thomadsen, T., Stidsen, T. The Generalized Fixed-Charge Network Design Problem. *Computers and Operations Research*, To appear, 2005.

Bibliography

[1] A. Balakrishnan. Lp extreme points and cuts for the fixed-charge network design problem. *Mathematical Programming*, 39(3):263–284, 1987.

[2] A. Balakrishnan, T.L. Magnanti, and R.T. Wong. A dual-ascent procedure for large-scale uncapacitated network design. *Operations Research*, 37(5):716–740, 1989.

[3] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–36, 1989.

[4] E. Balas. The prize collecting traveling salesman problem. ii. polyhedral results. *Networks*, 25(4):199–216, 1995.

[5] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46(3):316–29, 1998.

[6] P. Bauer, J.T. Linderoth, and M.W.P. Savelsbergh. A branch and cut approach to the cardinality constrained circuit problem. *Mathematical Programming*, 91(2):307–348, 2002.

[7] A. Billionnet and F. Calmels. Linear programming for the 0-1 quadratic knapsack problem. *European Journal of Operational Research*, 92(2):310–325, 1996.

[8] A. Billionnet, A. Faye, and E. Soutif. A new upper bound for the 0-1 quadratic knapsack problem. *European Journal of Operational Research*, 112(3):664–672, 1999.

[9] A. Caprara, D. Pisinger, and P. Toth. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11(2):125–37, 1999.

[10] COIN-OR. COmputational INfrastructure for Operations Research `www.coin-or.org`.

[11] A.M. Costa. A survey on benders decomposition applied to fixed-charge network design problems. *Computers and Operations Research*, 32(6):1429–1450, 2005.

[12] T.G. Crainic, A. Frangioni, and B. Gendron. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, 112(1-3):73–99, 2001.

[13] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1-3):229–237, 1999.

[14] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[15] C. Feremans, M. Labbe, and G. Laporte. Generalized network design problems. *European Journal of Operational Research*, 148(1):1–13, 2003.

[16] M. Fischetti, J.J. Salazar Gonzalez, and P. Toth. The symmetric generalized traveling salesman polytope. *Networks*, 26(2):113–23, 1995.

[17] M. Fischetti, J.J. Salazar Gonzalez, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–94, 1997.

[18] M. Fischetti, J.J. Salazar Gonzalez, and P. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10(2):133–48, 1998.

[19] M. Gendreau, M. Labbe, and G. Laporte. Efficient heuristics for the design of ring networks. *Telecommunication Systems - Modeling, Analysis, Design and Management*, 4(3-4):177–88, 1995.

[20] M. Gendreau, G. Laporte, and F. Semet. A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks*, 32(4):263–73, 1998.

[21] B. Gendron, T. G. Crainic, and A. Frangioni. Multicommodity capacitated network design. *Telecommunications Network Planning, Kluwer*, pages 1–19, 1999.

[22] L. Gouveia and J.M. Pires. Models for a steiner ring network design problem with revenues. *European Journal of Operational Research*, 133(1):21–31, 2001.

[23] K. Holmberg and D. Yuan. A lagrangean approach to network design problems. *International Transactions in Operational Research*, 5(6):529–539, 1998.

[24] K. Holmberg and D. Yuan. A lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research*, 48(3):461–81, 2000.

[25] D. Kang, K. Lee, S. Park, K. Park, and S.-B. Kim. Design of local networks using ushrs. *Telecommunication Systems*, 14(4):197–217, 2000.

[26] J.G. Klincewicz. Hub location in backbone/tributary network design: a review. *Location Science*, 6(1-4):307–335, 1998.

[27] M. Labbe, G. Laporte, I.R. Martin, and J.J. Salazar Gonzalez. The ring star problem: Polyhedral analysis and exact algorithm. *Networks - Bognor Regis*, 43(3):177–189, 2004.

[28] G. Laporte and S. Martello. The selective travelling salesman problem. *Discrete Applied Mathematics*, 26(2-3):193–207, 1990.

[29] G. Laporte and F. Semet. Computational evaluation of a transformation procedure for the symmetric generalized traveling salesman problem. *INFOR Journal*, 37(2):114–120, 1999.

[30] T.L. Magnanti, P. Mireault, and R.T. Wong. Tailoring benders decomposition for uncapacitated network design. *Mathematical Programming Studies*, 26:112–154, 1986.

[31] T.L. Magnanti and R.T. Wong. Network design and transportation planning: models and algorithms. *Transportation Science*, 1984.

[32] K. Park, K. Lee, S. Park, and H. Lee. Telecommunication node clustering with node compatibility and network survivability requirements. *Management Science*, 46(3):363–374, 2000.

[33] J. Petrek and V. Siedt. A large hierarchical network star-star topology design algorithm. *European Transactions on Telecommunications*, 12(6):511–22, 2001.

[34] A. Proestaki and M.C. Sinclair. Design and dimensioning of dual-homing hierarchical multi-ring networks. *IEE Proceedings-Communications*, 147(2):96 –104, 2000.

[35] D.M. Ryan and B. Foster. An integer programming approach to scheduling. *Computer Scheduling of Public Transport. Urban Passenger Vehicle and Crew Scheduling. Proceedings of an International Workshop*, pages 269–80, 1981.

[36] J. Shi and J.P. Fonseka. Hierarchical self-healing rings. *IEEE/ACM Transactions on Networking*, 3(6):690 –697, 1995.

[37] J.J. Shi and J.P. Fonseka. Analysis and design of survivable telecommunications networks. *IEE Proceedings-Communications*, 144(5):322–330, 1997.

[38] M.M. Sigurd. *Column Generation Methods and Applications*. PhD thesis, Dept. of Computer Science, University of Copenhagen, Denmark, 2004, `www.diku.dk/~sigurd`.

[39] F. Vanderbeck and L.A. Wolsey. An exact algorithm for ip column generation. *Operations Research Letters*, 19(4):151–159, 1996.

[40] T. Volgenant and R. Jonker. On some generalizations of the travelling-salesman problem. *Journal of the Operational Research Society*, 38(11):1073–9, 1987.