Degree-Based Spanning Tree Optimization

Summary of PhD Thesis

Gábor Salamon

Department of Computer Science and Information Theory
Budapest University of Technology and Economics

Supervisor: András Recski

2010

1 Introduction

Design process, operation and maintenance of telecommunication networks are dynamically developing research areas. Graphs are often used as mathematical models of these networks giving a special importance to the research of effective graph algorithms. In most cases, the obtained mathematical models are too complex to be solved optimally, thus the aim is to find a suboptimal approximate solution running in an acceptable time limit. In the thesis, we consider a design problem from the world of optical telecommunication networks. We build a model yielding to a spanning tree optimization problem. Our main goal is to present and build approximation algorithms for several related spanning tree optimization problems. We show that our algorithms can be efficiently used for network design applications as they perform well from both the theoretical and empirical point of view.

Our mathematical analysis has direct connection to the hamiltonicity theory, therefore, beside algorithmic aspects, our results have their own theoretical importance as well.

Connection routing in DWDM networks

Dense Wavelength Division Multiplexing (DWDM) [26] is a technology widely used in optical networks in order to increase the available bandwidth. The basic idea is to use different light wavelengths within a single optical fiber enabling multiple data connections at the same time. Switch devices in a DWDM network must be able to deal with wavelength multiplexing in order to correctly route data connections.

Let us have two applications which want to communicate to each other over our DWDM network. Their co-operation must be independent of the details of the underlying network protocols and technologies. Therefore, they are using an *application layer* to communicate. This application layer is built on the top of and is served by a *logical network layer* which is responsible for building up and managing the connection and for accessing the layers of physical transport. Logical network layer sends the data to be transferred to the *electronic (physical) layer* which converts it to an electronic signal. This signal now can be transported without dealing with its logical meaning. Up to this point, we have the layers of a classical network. However, optical networks transfer optical signals in optical fibers, thus they need an additional layer under the above mentioned ones: the *optical (physical) layer*. When the sender application generates a new connection demand, the electronic physical layer creates an electronic signal to be sent through the network. This electronic signal must be converted to an optical signal at the entry terminal of the DWDM network. Thus the physical transfer itself happens in the optical layer. Similarly, when the optical signal arrives to its destination, the end terminal converts it back to an electronic signal and passes it to the electronic layer which forwards it to the application. Each layer has its own responsibility. Its functionality can be implemented without any knowledge on the details of other layers.

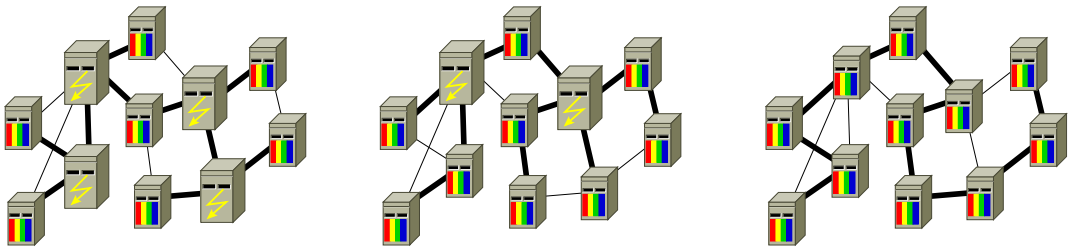
In *transparent DWDM* networks, the whole connection forwarding and routing is handled by full optical devices: *optical repeaters* and *optical cross connects (OXC's)*. Repeaters only forward the data stream without changing it. OXC's, in contrast, can route the connections and can execute wavelength multiplexing, demultiplexing, and conversions. However, the elevated price of OXC's makes their mass use inefficient. Therefore, network designers tend to use cheaper devices for the same task. These devices, called *electronic cross connects (EXC's)*, execute the routing and wavelength manipulation functionalities in the electronic layer. They convert the incoming optical signals to electronic ones, route connections, and then convert back the electronic signals to optical ones. Though their cost is significantly lower, they slow down the connection routing process and lose the transparency of the optical layer.

DWDM networks using EXC's for routing are called *opaque DWDM* networks. In these networks, the data is transferred in the form of optical signals using optical repeaters. However, all routing functionality is implemented in the electronic layer with the help of EXC's. We aim to build opaque networks where the routing is fully implemented by EXC's, but at the same time we want to use as few of these devices as possible.

The design problem considered is the following. We have an existing infrastructure composed of network nodes connected by optical fibers. Some of these nodes have a special role: they are connection terminals, that is, they input and output user requests to and from the network. We also have a traffic matrix which shows the estimated amount of data to be transferred between each pair of terminals. We have to place EXC's to the nodes and route all connections such that the estimated traffic can be sent through the network without major congestion. One can come up with different cost functions including the cost of devices and network links used, routing delays, loss of transparency, etc. If a combination of these cost functions is present, exact mathematical discussion becomes hardly possible though soft-computing techniques can still perform well. In [C1] we present a genetic algorithm based approach which deals with many of these cost functions at a time.

The thesis focuses on a more theoretical approach. We use a single cost function, that is, we want to minimize the number of EXC's used. As per our model we want to build a network where every node can communicate to every other node and there is no need to build protection paths to handle network failures. This means that we must ensure the existence of a single path between each terminal pair.

Our mathematical model is based on graphs. Network nodes are represented by the vertices of a graph G. The existing optical fiber links between nodes give the edges of G. The requirement that each terminal pair must be connected is satisfied by looking for a spanning tree T of G. We can suppose that we need routing functionality and wavelength manipulation only in the network nodes which communicate to at least three adjacent nodes. Therefore we have to put costly switch devices only to these nodes. This means that our aim is to minimize the number of at-least-3-degree vertices of T [C4, C7]. Figure 1 shows an example how



(a) (b) (c)

Figure 1: Eliminating EXC's by connection redesign

to decrease the number of EXC's needed in a communication network.

Mathematical Model

First we define what we consider to be an optimization problem.

Definition. [2] *An* optimization problem $\mathcal{P}$ *is characterized by the following quadruple of objects* $(I_{\mathcal{P}}, \mathrm{SOL}_{\mathcal{P}}, m_{\mathcal{P}}, \mathrm{goal}_{\mathcal{P}})$*, where:*

1. $I_{\mathcal{P}}$ *is the set of* instances *of* $\mathcal{P}$*;*
2. $\mathrm{SOL}_{\mathcal{P}}$ *is a function that associates to any input instance* $x \in I_{\mathcal{P}}$ *the set of* feasible solutions *of* x*;*
3. $m_{\mathcal{P}}$ *is the* measure function*, defined for pairs* (x, y) *such that* $x \in I_{\mathcal{P}}$ *and* $y \in \mathrm{SOL}_{\mathcal{P}}(x)$*. For every such pair* (x, y)*,* $m_{\mathcal{P}}(x, y)$ *provides the value of the feasible solution* y*. This value is also called* cost *in minimization problems and* utility *in maximization problems;*
4. $\mathrm{goal}_{\mathcal{P}} \in \{\mathrm{MIN}, \mathrm{MAX}\}$ *specifies whether* $\mathcal{P}$ *is a minimization or a maximization problem.*

A *spanning tree optimization problem* is to find a spanning tree T of a given undirected connected graph G which, depending on the problem, minimizes or maximizes a measure function $m(.)$. To be more general we allow weights to be put on the vertices and/or on the edges of G. If such weights are present, they can be taken into consideration when calculating $m(T)$. Examples for the spanning tree optimization problems are the Minimum Weight Spanning Tree problem [6, 19, 24], or the Minimum Diameter Spanning Tree problem [14, 15]. For a good survey on spanning tree optimization problems, the reader is referred to [27].

A spanning tree optimization problem is *degree-based* if $m(T)$ depends only on the vertex-degree distribution of T. If G has weights on its vertices then $m(T)$ can also depend on these weights. More precisely, if $d_T(v)$ is the degree of a vertex v in T and $c(v)$ is the weight of vertex v then $m(T)$ is determined by the pairs $(d_T(v); c(v))$.

In the thesis, we restrict ourselves to the special case where $m(T)$ can be written in the form of

$$m(T) = \sum_{i=1}^{n-1} f_i \sum \{c(v) : d_T(v) = i\}, \tag{1}$$

for some f_i, with $c(v) \equiv 1$ used for the unweighted case.

Notation and Definitions

By a *graph* $G = (V, E)$ we mean an undirected simple graph on vertex set $V(G) = V$ and edge set $E(G) = E$. We use $n = |V(G)|$ to denote the number of vertices and $m = |E(G)|$ to denote the number of edges of G. Every graph in the thesis is supposed to be connected unless explicitly stated otherwise. Let X and Y be subsets of vertices of G. Then $G[X]$ is the subgraph of G spanned by X, $\text{comp}_G(X)$ or simply $\text{comp}(X)$ is the number of components of $G[X]$.

A *spanning tree* T of a graph G is an acyclic connected subgraph of G containing all of its vertices. Edges of G are called G-edges, edges of T are called T-edges or *tree-edges*, elements of $E(G) \setminus E(T)$ are called *non-tree edges*. Vertices u and v are *G-neighbors* if they are adjacent in G, that is, $(u, v) \in E(G)$, and *T-neighbors* if they are adjacent in T, that is, $(u, v) \in E(T)$. The *G-degree* (*T-degree*) of a vertex v is the number of its G-neighbors (T-neighbors) and is denoted by $d_G(v)$ ($d_T(v)$). When it causes no confusion, we might leave the G out from these notions to abbreviate G-neighbors of v as neighbors of v, or $N(v)$, and G-degree of v as degree of v, or $d(v)$. A vertex v is called *pendant* if $d_G(v) = 1$. The degree of the highest G-degree vertex is denoted by $\Delta(G)$, or simply by Δ. The graph G is *r-regular* if all of its vertices have G-degree r. A 3-regular graph is also called *cubic*. A vertex v is a *leaf*, a *forwarding vertex*, or a *branching* of the spanning tree T, if its T-degree is 1, 2, or at least 3, respectively. Forwarding vertices and branchings are called *internal vertices*. We use $L(T)$ and $I(T)$ to denote the set of leaves and internal vertices of T, respectively. We denote by $\text{ml}(G)$ the minimum number of leaves that a spanning tree of G can have. A vertex set is said to be *G-independent* (or independent) if it spans no edges of G. Similarly, a vertex set X is *T-independent* if it spans no edges of T. Note that in this latter case X is still allowed to span non-tree edges. The size of the highest cardinality G-independent set is denoted by $\alpha(G)$. The spanning tree T of G is called an *independence tree* if its leaves are G-independent. The notion of independence tree is crucial as it is used throughout the thesis to establish approximation algorithms.

A *Hamiltonian path* of a graph is a simple path containing all vertices of the graph and a *Hamiltonian cycle* is a cycle with the same property. If G has a Hamiltonian path, it is called *traceable*. K_n is a complete graph and C_n is a cycle on n vertices, while K_{n_1,n_2} is the complete bipartite graph with color classes of size n_1 and n_2. Particularly, $K_{1,3}$ is called a *claw*. A graph is *claw-free* if it does not contain a $K_{1,3}$ as an induced subgraph.

2 Research goals

The research presented in the thesis is motivated both by telecommunication network design applications and by theoretical combinatorics. Our goal is to obtain results which can be used in both areas: to build algorithms based on simple steps and then to prove theoretical bounds on their goodness. Beside a mathematical analysis, we also aim to implement some of our algorithms to investigate their behavior on randomly generated inputs.

A part of our research is devoted to graph vulnerability parameters and their connection to spanning tree leaves. This direction relates our work to vulnerability and hamiltonicity theories and becomes a useful tool for proving approximation ratios of our algorithms.

Let us give a formal definition of the degree-based spanning tree optimization problems considered in the thesis. Their measure function is in the form of (1).

Problem: Minimum Branching Spanning Tree
Input: An undirected connected graph G.
Goal: Find a spanning tree T of G with a minimum number of (≥ 3)-degree vertices (branchings), that is, minimize $m(T) = \lvert\{v : d_T(v) \geq 3\}\rvert$.

Problem: Minimum Leaf Spanning Tree
Input: An undirected connected graph G.
Goal: Find a spanning tree T of G with a minimum number of 1-degree vertices (leaves), that is, minimize $m(T) = \lvert\{v : d_T(v) = 1\}\rvert$.

Problem: Maximum Internal Spanning Tree
Input: An undirected connected graph G.
Goal: Find a spanning tree T of G with a maximum number of (≥ 2)-degree vertices (internal vertices), that is, maximize $m(T) = \lvert\{v : d_T(v) \geq 2\}\rvert$.

Problem: Maximum Weighted Internal Spanning Tree
Input: An undirected connected graph G with a measure function $c : V(G) \to \mathbb{Q}$ on its vertices.
Goal: Find a spanning tree T of G with a maximum total weight of (≥ 2)-degree vertices (internal vertices), that is, maximize $m(T) = \sum \{c(v) : d_T(v) \geq 2\}$.

Observe that all of these problems can be viewed as a generalization of the Hamiltonian Path problem, and so are NP-hard. Therefore, our aim is to explore their approximability properties. We give efficient approximation algorithms for some of them and negative approximability results for some others to show that no such approximation exists, unless P=NP.

For the Minimum Branching Spanning Tree problem we show that there is no efficient approximation algorithm for general graphs. Thus we aim to give an

approximation for a subclass of input graphs, namely for evenly dense graphs, and also to find good heuristics for general graphs. These heuristics are based on the idea of looking for a spanning tree with only a few 1-degree vertices. Though the number of 1-degree vertices does not determine the number of branchings, decreasing their number helps, in most cases, decreasing the number of branchings, too.

Therefore, we also deal with degree based spanning tree optimization problems in a bit more general way. We consider the Minimum Leaf Spanning Tree problem, where the task is to find a spanning tree with a minimum number of leaves. Lu and Ravi [20] showed that this problem has no constant factor approximation algorithm, unless P=NP. However, if we complement the measure function and we count the non-leaves (internal vertices) instead of the leaves then the situation is better. The obtained Maximum Internal Spanning Tree problem is trivially equivalent to the Minimum Leaf Spanning Tree problem as long as we focus on the set of optimal solutions. From an approximation point of view, however, the two problems behave differently. Our aim is to explore the approximability of the Maximum Internal Spanning Tree problem. By giving efficient approximation algorithms we obtain good heuristics both for the Minimum Leaf Spanning Tree and for the Minimum Branching Spanning Tree problems.

We also aim to provide results in the field of vulnerability theory. Vulnerability parameters measure how much damage can be caused to a graph by removing some of its "important" parts. These parameters have strong connection to hamiltonicity theory and, as we show in the thesis, to the number of leaves of spanning trees.

3 Overview of Research Methods and Results

In the thesis, we consider several degree-based spanning tree optimization problems all being a generalization of the Hamiltonian Path problem. Our results are organized in four groups of theses.

Thesis group 1: Chapter 3 deals with the Minimum Branching Spanning Tree problem. We obtain both positive and negative approximability results. Namely, on one hand we present Algorithm MinBST which yields a spanning tree having at most $3\left\lceil \log_{\frac{1}{1-c}} n \right\rceil + 1$ branchings for evenly dense graphs (of which every vertex has a degree of at least cn), see Theorem 3.3.1. On the other hand, we show that this approximation ratio is very likely the best possible. We give an approximation ratio preserving reduction from the Minimum Set Cover problem to the Minimum Branching Spanning Tree problem thus proving that any ratio better than $\Omega(\log n)$ implies P=NP, see Theorem 3.2.3. Our results discussed in Chapter 3 have originally been published in [C4].

Thesis group 2: Chapter 4 is focusing on our work on the connection of spanning tree leaves and two graph vulnerability parameters: scattering number [17, 28],

and cut-asymmetry (Definition 4.3.1). Some of these results are then used in Chapter 5 to build our approximation algorithms for the Maximum Internal Spanning Tree problem. In Section 4.1 we generalize the well-known necessary condition of traceability by proving that every spanning tree of a graph G has at least one more leaf than its scattering number $\mathrm{sc}(G)$ (Theorem 4.1.5). If the graph itself is a tree, its scattering number can be used to upper bound the number of leaves (Theorem 4.1.7). In Section 4.3 we first provide some basic properties of cut-asymmetry $\mathrm{ca}(G)$ of a graph G. Namely, we show that $\mathrm{ca}(G) = 0$ if and only if G is either a complete graph or a cycle (Theorem 4.3.2). We also prove that $\mathrm{ca}(G) \leq 1$ is a sufficient condition for the existence of a Hamiltonian path in G (Theorem 4.3.6). Unfortunately, even a graph with a Hamiltonian path can have a big cut-asymmetry, as shown in Theorem 4.3.7. Later in Section 4.3, we define leaf-independence $\mathrm{li}(G)$ as the maximum number of independent leaves in a spanning tree of G. It turns out that this measure can be considered as an equivalent definition of cut-asymmetry, since $\mathrm{li}(G) = \mathrm{ca}(G) + 1$ always holds (Theorem 4.3.10). Corollary 4.3.13 shows that leaf-independence and the cardinality of a minimum connected vertex cover sums up to n and thus proves that both cut-asymmetry and leaf-independence are NP-hard to compute (Theorem 4.3.14). Our results presented in Chapter 4 have originally been published in [J3], in [J4], and in [C5].

Thesis group 3: Chapter 5 is about the Maximum Internal Spanning Tree problem. First we present Algorithm ILST that yields a spanning tree with independent leaves, and then we prove that such a spanning tree always forms a 2-approximation for the Maximum Internal Spanning Tree problem (Theorem 5.2.2). In Section 5.2 we provide three different proofs for this fact, one direct proof, one based on the results of Chapter 4, and one based on primal-dual linear programming techniques. Algorithm RDFS, a refined version of Algorithm ILST, has even better approximation properties when applied on special graph classes. It is a 3/2-approximation for claw-free graphs (Theorem 5.4.2) and a 6/5-approximation for cubic graphs (Theorem 5.4.4). In Section 5.5, we develop Algorithm LOST, and using an improved version of the above mentioned linear programming approach, we prove that it is a 7/4-approximation for the Maximum Internal Spanning Tree problem in graphs with no pendant vertices (Theorem 5.5.2). Section 5.6 deals with the case when vertices are weighted and we have to maximize the weighted sum of the internal vertices of the obtained spanning tree (Maximum Weighted Internal Spanning Tree problem). To solve this problem (in graphs with no pendant vertices), we present Algorithm WLOST which yields a $(2\Delta - 3)$-approximation (Theorem 5.6.2). We then further improve this algorithm obtaining Algorithm RWLOST which is a 2-approximation if the input graph is claw-free (Theorem 5.6.6). In Section 5.7 we consider the Maximum Internal Spanning Tree problem from another point of view: instead of looking for a spanning tree with few leaves, we try to cover as many vertices as possible with a $\leq q$-leaf subtree. This approach is a generalization of searching for a long path in graphs, see Theorem 5.7.4. In Section

5.8, we show that if any $(q+1)$-element independent set of a claw-free graph (on n vertices) has a degree-sum of at least $n-q$, then the graph has a spanning tree with at most q leaves (Theorem 5.8.1). Our results discussed in Chapter 5 have originally been published in [C4], in [C6], in [J3], in [J4], and in [J5].

Thesis group 4: Section 5.9 discusses the results of our experimental analysis on our algorithms paying a particular attention to compare different traversal algorithms for obtaining the initial spanning tree.

4 New Results

Thesis group 1: Minimum Branching Spanning Tree problem

In Chapter 3 we investigate the Minimum Branching Spanning Tree problem which aims to find a spanning tree T of a given input graph G such that T has a minimum number of branchings among all spanning trees of G. As spanning trees with no branchings are exactly the Hamiltonian paths of G, we cannot expect exact polynomial-time solution for this problem. Instead, we explore the approximability of the problem. The Minimum Branching Spanning Tree problem was first considered by Gargano et al. [11] as a degree-based generalization of the Hamiltonian Path problem. They have proved that it is NP-complete to decide whether a spanning tree with at most k branchings exists (for any fixed k). They have also given an algorithm [10] that finds a single-branching spanning tree (a so called spanning spider) if each 3-element independent set of the input graph G has a degree sum of at least $|V(G)|-1$. The case when the input graph is bipartite is discussed in [9]. Flandrin et al. [8] considered the problem of finding a spanning spider having its branching fixed in advance. They proved that a graph G has a spanning spider whenever the sum of its minimum and maximum vertex degree is at least $|V(G)|$.

Our main negative approximability result on the Minimum Branching Spanning Tree problem is based on an approximation ratio preserving reduction from the Minimum Set Cover problem. Recall the following definition [16]:

Problem: Minimum Set Cover
Input: A ground set $\mathcal{S}$ and a set $\Sigma = \{S_j\}_{j=1}^{s}$ of its subsets.
Goal: Find a minimum number of subsets of Σ whose union contains each element of $\mathcal{S}$.

Alon et al. [1] proved that the Minimum Set Cover problem is not approximable better than a multiplicative ratio of $\Omega(\log|\mathcal{S}|)$, unless P=NP, that is, the Minimum Set Cover problem is not in APX (which is the class of constant factor approximable problems). Based on their result our reduction yields to

Thesis 1.1. **[C4]** *The* Minimum Branching Spanning Tree *problem is not in APX. Moreover, it is not approximable better than a multiplicative ratio of* $\Omega(\log |V(G)|)$*, unless P=NP.*

We then present a positive approximability result on the Minimum Branching Spanning Tree problem. We provide Algorithm MinBST, the first approximation algorithm which achieves the approximation ratio of $\mathcal{O}(\log n)$ whenever the input graph is non-traceable and all of its vertices have a degree of $\Omega(n)$. More precisely, in evenly dense graphs Algorithm MinBST produces a spanning tree with $\mathcal{O}(\log n)$ branchings.

Thesis 1.2. **[C4]** *Let G be a connected graph on n vertices and m edges. If the G-degree of each vertex is at least cn (for some number $c \in \mathbb{R}$) then Algorithm MinBST yields a spanning tree with at most $3\left\lceil \log_{\frac{1}{1-c}} n \right\rceil + 1$ branchings in $\mathcal{O}(m + n \log n)$ time.*

The basic idea of Algorithm MinBST is the following. Given an input graph G, our approximation algorithm starts with an empty graph $H = (V, \emptyset)$. Then it subsequently adds G-edges to H, until H becomes a spanning forest without isolated vertices. In each iteration of this spanning forest building process, we greedily select a vertex v such that there is a maximum number of isolated vertices of H among the G-neighbors of v. Then we add to H all G-edges which connects v to an isolated vertex of H. When H has no more isolated vertices, some additional G-edges are used to connect the components of H forming the output spanning tree.

Thesis group 2: Spanning Tree Leaves and Vulnerability

In Chapter 4 we focus on graph vulnerability parameters. They are used to measure how much structural damage can be caused in a graph by removing some "important parts" of it [3, 4, 12, 13, 28]. Both hamiltonicity theory and network design applications widely use these parameters to describe the structure of graphs. Our work fits into both of these categories. We first prove some theoretical results on the connection between the number of spanning tree leaves and vulnerability parameters. Then, in Chapter 5, we use these results to build an approximation algorithm for the Maximum Internal Spanning Tree problem. Besides, our framework yields a new 2-approximation algorithm for the Minimum Connected Vertex Cover problem, too.

The first connection between hamiltonicity and vulnerability was given in a well-known theorem of basic graph theory stating that if a graph is traceable then it cannot be split to more than $k + 1$ components by removing at most k of its vertices. This theorem, giving only a necessary condition of traceability, is based on a vulnerability property of the graph. Therefore, it is worth investigating how vulnerability parameters can provide sufficient conditions of traceability. A considerable amount of research followed this approach [4].

In our work, we use two closely related vulnerability parameters: scattering number and cut-asymmetry. Scattering number shows how many components we can get by removing a few vertices [17, 18, 28]. Cut-asymmetry does the same when a few *connected subgraphs* are removed [J3, J4]. We use scattering number to lower bound the number of leaves in a spanning tree, and cut-asymmetry to upper bound the number of G-independent leaves of a spanning tree.

Jung defined the scattering number of a graph as follows:

Definition. [17] *The* scattering number *of a non-complete graph $G = (V, E)$ is*

$$\mathrm{sc}(G) = \max_{X \subset V, X \neq \emptyset} \{\mathrm{comp}(G[V \setminus X]) - |X| : \mathrm{comp}(G[V \setminus X]) \geq 2\}.$$

By definition, the scattering number of the complete graph K_n is $\mathrm{sc}(K_n) = -\infty$.

We introduce cut-asymmetry in a similar way. However, we count the connected subgraphs (instead of individual vertices) to be removed. The definition takes the maximum over all non-trivial cuts of G. Cut-asymmetry counts how much the two sides of a cut can differ in terms of the number of components.

Definition. [J3] *The cut-asymmetry of a graph $G = (V, E)$ is*

$$\mathrm{ca}(G) = \max_{X \subset V, X \neq \emptyset} \{\mathrm{comp}(G[V \setminus X]) - \mathrm{comp}(G[X])\}.$$

The above theorem can be reformulated by means of scattering number: if a graph is traceable then its scattering number is at most one, that is, the scattering number provides a necessary condition of traceability. Investigating the properties of cut-asymmetry, we show that it gives a sufficient condition for the existence of a Hamiltonian path, namely, if its value is at most one then the graph is traceable. Unfortunately, cut-asymmetry of a traceable graph can be greater than 1. These are our main hamiltonicity related results.

Thesis 2.1. [J3, J4, C5] *Basic properties of cut-asymmetry:*

- $\mathrm{ca}(G) = 0$ *if and only if G is either a complete graph or a cycle;*
- $\mathrm{ca}(G) \leq 1$ *implies that G is traceable;*
- *if G is a traceable graph on n vertices then* $\mathrm{ca}(G) \leq \lfloor \frac{n-1}{2} \rfloor$*;*
- *if k is an arbitrary integer such that $0 \leq k \leq \lfloor \frac{n-1}{2} \rfloor$ then there exists a traceable graph G on n vertices for which* $\mathrm{ca}(G) = k$*;*
- *if Z is a maximum size independent vertex set of G for which $G[V \setminus Z]$ is connected then $|Z| = \mathrm{ca}(G) + 1$;*
- $\mathrm{ca}(G)$ *is NP-hard to compute.*

Chapter 4 contains our results regarding the connection of vulnerability parameters and the number of leaves of spanning trees.

We show that the scattering number of a q-leaf tree on at least 3 vertices is between $q/2$ and $q-1$, that is, these measures determine each other up to a multiplicative factor of 2. Contrary, the cut-asymmetry of a tree exactly determines the number of its leaves. A joint result with Gábor Wiener proves that the cut-asymmetry of a q-leaf tree is $q-1$.

In the thesis, we show that the leaves of any minimum leaf spanning tree which is not a Hamiltonian path are G-independent. This fact inspires us to examine how many G-independent leaves a spanning tree can have. For this reason we define leaf-independence of a graph as follows.

Definition. [J3] *Let G be any connected graph and T be a spanning tree of G. The* leaf-independence *of T in G, denoted by $\mathrm{li}_G(T)$, is the cardinality of a maximum G-independent subset of $L(T)$. The* leaf independence $\mathrm{li}(G)$ of the graph G *is the maximum of $\mathrm{li}_G(T)$ over all spanning trees of G.*

The notion of leaf-independence has a strong relation to cut-asymmetry and to connected vertex covers. The following thesis is about these relations. Besides, it establishes the computational complexity of leaf-independence. Let $\mathrm{cvc}(G)$ denote the minimum size connected vertex set of G which covers all G-edges.

> **Thesis 2.2. [J3, J4, C5]** *For the leaf-independence of an n-vertex connected graph G, the followings hold:*
>
> - $\mathrm{li}(G) = \mathrm{ca}(G) + 1$*;*
> - *if G is not complete and not a cycle then* $\mathrm{li}(G) \geq \mathrm{ml}(G)$*;*
> - $\mathrm{li}(G) = n - \mathrm{cvc}(G)$*;*
> - $\mathrm{li}(G)$ *is NP-hard to compute.*

The main results of Chapter 4 can be summarized in the following series of inequalities. They directly prove that any spanning tree with independent leaves is a 2-approximation for the Maximum Internal Spanning Tree problem and that the internal vertices of such a spanning tree form a 2-approximation for the Minimum Connected Vertex Cover problem. This latter was first proved by Savage [25].

> **Thesis 2.3. [J3, J4, C5]** *Let G be a connected graph on n vertices. If G is not complete and not a cycle then*
>
> $$n - \mathrm{sc}(G) - 1 \geq n - \mathrm{ml}(G) \geq n - \mathrm{li}(G) = \mathrm{cvc}(G)$$
> $$= n - \mathrm{ca}(G) - 1 \geq n - \alpha(G) \geq \frac{1}{2}(n - \mathrm{sc}(G)).$$

This directly implies that any independence tree is a 2-approximation for the Maximum Internal Spanning Tree *problem, and that the internal vertices of such a tree provide a 2-approximation for the* Minimum Connected Vertex Cover *problem.*

Thesis group 3: Maximum Internal Spanning Tree problem

Chapter 5 deals with the Maximum Internal Spanning Tree problem which was already considered in the literature from different points of view. Fernau et al. [7] gave exponential-time exact algorithms to solve it. Prieto and Sloper proved that it is fixed parameter tractable [22, 23], that is, a spanning tree with at least k internal vertices can be found in $f(k)\mathcal{O}(n^c)$ time whenever it exists. Our work uses the approach of approximation algorithms, we aim to find suboptimal solutions in polynomial time. We consider general input graphs, as well as some special input graph classes.

Given an input graph G, we provide Algorithm ILST (Independent Leaves Spanning Tree), a modified version of Depth First Search, which yields a spanning tree with G-independent leaves. Then we prove that Algorithm ILST is a 2-approximation algorithm for the Maximum Internal Spanning Tree problem. We give three different proofs for the approximation ratio. The first one is based on vulnerability parameters and uses the results of Chapter 4. The second one is a direct proof, while the third one is a linear programming based approach which is further enhanced in Section 5.5 where we give a 7/4-approximation algorithm for the Maximum Internal Spanning Tree problem for graphs with no pendant vertices. We also consider the case when we run Algorithm ILST in r-regular graphs.

Thesis 3.1. [J3, C4] *Algorithm ILST is an $\mathcal{O}(m)$-time 2-approximation algorithm for the* Maximum Internal Spanning Tree *problem. Moreover, it provides an $\frac{r+1}{3}$-approximation for r-regular graphs. In particular, the approximation factor is 4/3 for cubic graphs and 5/3 for 4-regular graphs.*

As a result of a joint work with Gábor Wiener we give Algorithm RDFS (Refined DFS) in which we specify how to choose the next vertex of the traversal in the cases when Algorithm DFS itself would choose arbitrarily from several candidates. The main idea is to select the vertex that has the minimum number of non-visited neighbors. We prove that for the Maximum Internal Spanning Tree problem, Algorithm RDFS is a 3/2-approximation for claw-free graphs and a 6/5-approximation for 3-regular graphs.

The main result of Chapter 5 is Algorithm LOST (Locally Optimal Spanning Tree). It starts by building a spanning tree then it successively executes local changes (defined by improvement rules) as far as possible. When no more rule can be executed we obtain a locally optimal spanning tree (LOST).

The proof of the approximation ratio is based on a primal-dual technique of linear programming: we build a primal program such that each of its integer solutions is composed of the characteristic vector of a spanning tree and its internal vertices. Thus, an optimum solution T^* of the Maximum Internal Spanning Tree problem defines a feasible integer primal solution with a value of $|I(T^*)|$. Then we use dual solutions to upper bound this quantity by means of $|I(T)|$, where T is the LOST that our algorithm outputs. Finally, the upper bounds are used to prove the approximation ratio.

> **Thesis 3.2. [J5, C6]** *Algorithm LOST is an $\mathcal{O}(|V|^4)$-time 7/4-approximation for the* Maximum Internal Spanning Tree *problem in graphs that have no pendant vertices.*

We also consider the Maximum Weighted Internal Spanning Tree problem and give two algorithms, both based on local improvement steps.

> **Thesis 3.3. [J5, C6]** *There exists an $\mathcal{O}(|V|^4)$-time $(2\Delta-3)$-approximation for the* Maximum Weighted Internal Spanning Tree *problem in graphs with no pendant vertices.*
>
> *There exists an $\mathcal{O}(|V|^4)$-time 2-approximation for the* Maximum Weighted Internal Spanning Tree *problem for claw-free graphs that have no pendant vertices.*

Apart from focusing on spanning trees with few leaves we also follow the opposite approach. We fix the number of leaves to q and examine how many vertices can be spanned by an $\leq q$-leaf subtree. This approach is a generalization of finding a longest path in a graph.

Let G be a graph on n vertices and let $\sigma_q(G)$ denote the minimum degree-sum of a q-element independent subset of $V(G)$. Ore's theorem [21] states that if $\sigma_2(G) \geq n$ then G has a Hamiltonian path. Bermond [5] showed that if G is 2-connected then it has a path of length $\min\{n, \sigma_2(G)\}$. Broersma and Tuinstra proved that if $\sigma_2(G) \geq n - q + 1$, for some integer $2 \leq q \leq n - 1$, then G has a q-leaf spanning tree.

Our research on $\leq q$-leaf subtrees fits into these series of results. To formulate our statement on subtrees we use the following notation. Let S_q be a q-element independent set and let x_1, x_2 be the two highest degree vertices of S_q. Then we denote by $\rho_{q,2}(G)$ the minimum $\min_{S_q}\{d(x_1) + d(x_2)\}$, where the minimum is taken over all q-element independent sets. Our results on $\leq q$-leaf subtrees are summarized in the following thesis.

Thesis 3.4. [J4, T2]

- *Let G be a connected graph on n vertices and let $2 \leq q < \alpha(G)$ be an integer. Then G has a subtree with at most q leaves that spans at least $\min\{\rho_{q,2}(G) + q - 1, n\}$ vertices of G. Such a subtree can be found algorithmically.*

  As a corollary, if $q' \geq 2$ is an integer such that $q' \geq \alpha(G)$ or $\rho_{q',2}(G) \geq n - q' + 1$ then G has a spanning tree with at most q' leaves.

- *Let G be a connected claw-free graph on n vertices. For any integer $2 \leq q$, if $\sigma_{q+1}(G) \geq n - q$ then G has a spanning tree with at most q leaves.*

- *Let T be a tree having more than q leaves. Let T_q be a maximum size q-leaf subtree of T. Then there exists a $(q+1)$-leaf subtree T_{q+1} of T such that T_q is a subtree of T_{q+1}, and T_{q+1} has a maximum number of vertices among all $(q+1)$-leaf subtrees of T. Moreover, T_{q+1} can be obtained by adding to T_q a longest path of $E(T) - E(T_q)$ that has one of its ends in T_q.*

Thesis group 4: Experimental analysis

Section 5.9 presents the results of our experimental analysis on the Maximum Internal Spanning Tree problem. We compare the performance of different traversal algorithms used for obtaining the initial spanning tree. We consider four algorithms. Each starts with the construction of an initial spanning tree and then applies some local improvement rules of Algorithm LOST as long as possible. The difference among the four algorithms is the way of finding the initial spanning tree: Algorithm 1 creates a random spanning tree; Algorithm 2 creates a DFS-tree; Algorithm 3 creates a so called FIFO-DFS-tree (see Section 2.2); Algorithm 4 creates a so called RDFS-tree (see Section 5.4). To create the set of input graphs, we use two different graph generation methods. Both of these methods start with the creation of a simple path and then add extra edges randomly. As a result, we can test our algorithms on traceable graphs where the value of the optimum solution is known.

Thesis 4.1. [T2] *We performed an experimental analysis in order to compare different methods for creating an initial spanning tree for Algorithm LOST. We examined how local improvement rules enhance the quality of solutions for each initial tree. The tests were done on several randomly generated traceable graphs having $100, 300, 500$ vertices.*

The results showed that the theoretical approximation factors are overperformed and so our algorithms can be efficiently used in practical applications to solve the Maximum Internal Spanning Tree *, the* Minimum Leaf Spanning Tree *, or the* Minimum Branching Spanning Tree *problems. Our main observations are summarized below.*

- There is a strong connection between the average degree of the input graph and the average number of leaves of the output spanning tree. Our algorithms performs the best on dense graphs. This is somewhat we expect, as in these graphs the traversals themselves need to step back much more rarely and the local improvement rules can be executed for more subgraphs. On the other hand, if the average degree of the input graph is very close to 2 then we find a good enough solution with high probability since the number of edges and so the number of spanning trees is low. The most interesting part is when the average degree is between these two extremities. Observe that even in this range, there is no significant difference in the results obtained using the two different input graph generation methods.

- The algorithm using Algorithm RDFS for finding the initial spanning tree overperforms the others for every input graph. Moreover, not considering the small and sparse graphs, this is true even if we do not execute any local improvement rule on the initial RDFS-tree. This surprising fact shows that in many situations we can gain more by the appropriate choice of the initial traversal than by the application of the more and more sophisticated local improvements. Algorithm RDFS provides a good approximation factor for claw-free and cubic graphs. As our experiments show, it can be used effectively for general graphs, as well.

- The ranking of the four algorithms based on their average performance is the same for almost all considered input graphs. If we compare the number of leaves after the local improvement steps then we find the RDFS-tree approach being the best choice, followed by random tree, FIFO-DFS-tree, and DFS-tree approaches, respectively. The fact that the random-based approach yields better results than the DFS-tree and the FIFO-DFS-tree ones shows the power of the applied local improvement technique. It is important to see that the theoretical approximation factor of 7/4 for the Maximum Internal Spanning Tree problem is highly overperformed in terms of the average behavior of our algorithms for the considered input graphs.

- Algorithm RDFS has the weakest performance for input graphs with an average degree of about 3.5–4, while this measure is about 4–4.5 for the other 3 algorithms. Algorithm RDFS gives a solution being close to the optimum for most of the cases where this average degree is at least 10. From this point of view, it is much better than the other 3 algorithms: the random tree approach

usually gives a close-to-optimum tree when the average degree of the input graph is at least $n/5$, and the FIFO-DFS- and DFS-tree approaches give such a tree when the average degree is at least $n/3$.

Acknowledgement

I would like to express my gratitude to everyone who helped me, in one way or another, preparing the thesis. First of all, I thank my family for their long lasting patience, support and inspiration. I am particularly grateful to my supervisor, András Recski for teaching me many interesting topics in graph theory, for giving me the possibility to become a member of Department of Computer Science and Information Theory, for introducing me to the research topic of Steiner- and spanning trees, and last but not least, for supporting my research with his indispensable pieces of idea and advice. My special thanks to my co-author, Gábor Wiener for working with me on some topics discussed in the thesis, and for giving me his professional support. I wish he could play Carcassonne much better in the future :) . I thank Katalin Friedl, Jácint Szabó, and Ferenc Wettl for improving the quality of the thesis by reading it through and making several valuable comments on it. I am grateful to András Sebő for his help and indications in the field of linear programming and approximation algorithms, and also for welcoming me as a visitor of the Graph Theory and Combinatorial Optimization Group of IMAG, in the beautiful city of Grenoble, France. I also express my thanks to András Frank whose classes on linear programming, combinatorial optimization and graph theory gave a solid foundation for my research work. At last, let me mention that the research leading to the results discussed in the thesis was supported by Grant Nos. 042559, 044733, and 67651 of the Hungarian National Science Foundation (OTKA), and by the Grant No. 2003-5044438 of the European MCRTN Adonet Contract.

Publications of the Author

Journal Papers

[J5] G. Salamon. Approximating the Maximum Internal Spanning Tree problem. *Theoretical Computer Science, MFCS 2007 special issue*, 410:5273–5284, 2009.

[J4] G. Salamon. Vulnerability bounds on the number of spanning tree leaves. *Ars Mathematica Contemporanea*, 2:77–92, 2009.

[J3] G. Salamon and G. Wiener. On finding spanning trees with few leaves. *Information Processing Letters*, 105:164–169, 2008.

[J1–J2] A. Recski, G. Salamon, and D. Szeszlér. Improving size-bounds for subcases of square-shaped switchbox routing. *Periodica Polytechnica, Electrical Engineering, BUTE, Budapest*, 48:55–60, 2003. Full version in *Annales Universitatis Scientiarum Budapestinensis, Sectio Mathematica*, 49:15–24, 2006.

Conference Papers

[C8] G. Salamon. A Survey on Algorithms for the Maximum Internal Spanning Tree and Related Problems. Accepted to *International Symposium on Combinatorial Optimization (ISCO 2010)*, March 2010.

[C7] G. Salamon. Feszítőfa optimalizálási problémák a Hamilton utak általánosítására, (Spanning tree optimization problems for generalizing Hamiltonian paths, in Hungarian). In *XIII. Fiatal Műszakiak Tudományos Ülésszaka, Erdélyi Múzeum-Egyesület, (Proc.)*, pages 203–206, March 2008.

[C6] G. Salamon. Approximation algorithms for the Maximum Internal Spanning Tree problem. In *Proc. of the* 32^{nd} *International Symposium on Mathematical Foundations of Computer Science (MFCS 2007)*, volume 4708 of *LNCS*, pages 90–102, August 2007.

[C5] G. Salamon and G. Wiener. Leaves of spanning trees and vulnerability. In *Proc. of the* 5^{th} *Hungarian-Japanese Symposium on Discrete Mathematics and Its Applications (HJ 2007)*, pages 225–235, April 2007.

[C4] G. Salamon. Spanning tree optimization problems with degree-based objective functions. In *Proc. of the* 4^{th} *Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications (JH 2005)*, pages 309–315, June 2005.

[C3] A. Recski, G. Salamon, and D. Szeszlér. Improving size-bounds for subcases of square-shaped switchbox routing. In *Proc. of the John von Neumann PhD Conference, BUTE, Budapest*, pages 43–46, October 2003.

[C2] A. Pataricza, G. Salamon, and D. Varró. Formal verification of model transformation systems. In *Fast Abstracts of the* 4^{th} *European Dependable Computing Conference (EDCC 2002)*, pages 15–16, October 2002.

[C1] T. Cinkler, S. Győri, J. Harmatos, and G. Salamon. Dimensioning WDM-based multi-layer transport networks with grooming by genetic algorithm. In *Proc. of the* 7^{th} *European Conference on Networks and Optical Communication (NOC 2002)*, pages 44–51, June 2002.

Theses

[T2] G. Salamon. *Degree-Based Spanning Tree Optimization.* PhD thesis, Budapest University of Technology and Economics, Hungary, 2010.

[T1] G. Salamon. *Formal Verification of Model Transformation Systems.* Master's thesis, Budapest University of Technology and Economics, Hungary, 2002.

Independent References to Publications

On finding spanning trees with few leaves [J3]:

[R1] D. Eppstein. Paired approximation problems and incompatible inapproximabilities. Arxiv preprint arXiv:0909.1870, 2009.

[R2] H. Fernau, S. Gaspers, and D. Raible. Exact and parameterized algorithms for Max Internal Spanning Tree. In *Proc. of the* 35^{th} *International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2009)*, volume 5911 of *LNCS*, pages 100–111, December 2009.

[R3] H. Fernau, S. Gaspers, D. Raible, and A. A. Stepanov. Exact exponential time algorithms for Max Internal Spanning Tree. Arxiv preprint arXiv:0811.1875, 2008.

[R4] M. Knauer and J. Spoerhase. Better approximation algorithms for the maximum internal spanning tree problem. In *Proc. of the* 11^{th} *Workshop on Algorithms and Data Structures (WADS 2009)*, volume 5664 of *LNCS*, pages 459–470, July 2009.

[R5] W. Yang, H.-R. Tseng, R.-H. Jan, and B.-Y. Shen. Broadcasting with the least energy is an NP-complete problem. *International Journal of Multimedia and Ubiquitous Engineering*, 3(3):55–65, 2008.

Approximation algorithms for the Maximum Internal Spanning Tree problem [C6]:

[R6] H. Fernau, S. Gaspers, and D. Raible. Exact and parameterized algorithms for Max Internal Spanning Tree. In *Proc. of the* 35^{th} *International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2009)*, volume 5911 of *LNCS*, pages 100–111, December 2009.

[R7] H. Fernau, S. Gaspers, D. Raible, and A. A. Stepanov. Exact exponential time algorithms for Max Internal Spanning Tree. Arxiv preprint arXiv:0811.1875, 2008.

[R8] M. Knauer and J. Spoerhase. Better approximation algorithms for the maximum internal spanning tree problem. In *Proc. of the* 11^{th} *Workshop on Algorithms and Data Structures (WADS 2009)*, volume 5664 of *LNCS*, pages 459–470, July 2009.

Approximating the Maximum Internal Spanning Tree problem [J5]:

[R9] M. Knauer and J. Spoerhase. Better approximation algorithms for the maximum internal spanning tree problem. In *Proc. of the* 11^{th} *Workshop on Algorithms and Data Structures (WADS 2009)*, volume 5664 of *LNCS*, pages 459–470, July 2009.

Further References

[1] N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms*, 2(2):153–177, 2006.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation.* Springer-Verlag, 1999.

[3] C. A. Barefoot, R. Entringer, and H. Swart. Vulnerability in graphs — a comparative survey. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 1:13–22, 1987.

[4] D. Bauer, H. Broersma, and E. Schmeichel. Toughness in graphs — a survey. *Graphs and Combinatorics*, 22:1–35, 2006.

[5] J-C. Bermond. On Hamiltonian walks. In *Proc. of the* 5^{th} *British Combinatorial Conference*, pages 41–51, 1975.

[6] O. Borůvka. O jistém problému minimálním (About a certain minimal problem, in Czech). *Práca Moravské Přirodovědecké Společnosti*, 3:37–58, 1926.

[7] H. Fernau, S. Gaspers, D. Raible, and A. A. Stepanov. Exact exponential time algorithms for Max Internal Spanning Tree. Arxiv preprint arXiv:0811.1875, 2008.

[8] E. Flandrin, T. Kaiser, R. Kužel, H. Li, and Z. Ryjáček. Neighborhood unions and extremal spanning trees. *Discrete Mathematics*, 308:2343–2350, 2008.

[9] L. Gargano and M. Hammar. There are spanning spiders in dense graphs (and we know how to find them). In *Proc. of the* 30^{th} *International Colloquium on Automata, Languages and Programming (ICALP 2003)*, volume 2719 of *LNCS*, pages 802–816, July 2003.

[10] L. Gargano, M. Hammar, P. Hell, L. Stacho, and U. Vaccaro. Spanning spiders and light-splitting switches. *Discrete Mathematics*, 285:83–95, 2004.

[11] L. Gargano, P. Hell, L. Stacho, and U. Vaccaro. Spanning trees with bounded number of branch vertices. In *Proc. of the* 29^{th} *International Colloquium on Automata, Languages and Programming (ICALP 2002)*, volume 2380 of *LNCS*, pages 355–365, July 2002.

[12] W. Goddard. Measures of vulnerability — the integrity family. *Networks*, 24(4):207–213, 1994.

[13] W. Goddard and H. C. Swart. Integrity in graphs: bounds and basics. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 7:139–151, 1990.

[14] R. Hassin and A. Tamir. On the Minimum Diameter Spanning Tree problem. *Information Processing Letters*, 53:109–111, 1995.

[15] J.-M. Ho, D.T. Lee, C.-H. Chang, and C.K. Wong. Minimum diameter spanning trees and related problems. *SIAM Journal of Computing*, 20:987–997, 1991.

[16] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer System Science*, 9:256–278, 1974.

[17] H. A. Jung. On a class of posets and the corresponding comparability graphs. *Journal of Combinatorial Theory Ser. B*, 24:125–133, 1978.

[18] A. Kirlangiç. Scattering number in graphs. *International Journal of Mathematics and Mathematical Sciences*, 30(1):1–8, 2002.

[19] J. B. Kruskal. On the shortest spanning subtree of a graph and the Travelling Salesman problem. *Proc. of American Mathematical Society*, 7:48–50, 1956.

[20] H.-I. Lu and R. Ravi. The power of local optimization: Approximation algorithms for maximum-leaf spanning tree (draft). Technical Report CS-96-05, Department of Computer Science, Brown University, Providence, Rhode Island, 1996.

[21] O. Ore. Note on Hamiltonian circuits. *American Mathematical Monthly*, 67:55, 1960.

[22] E. Prieto. *Kernelization in FPT Algorithm Design*. PhD thesis, The University of Newcastle, Australia, 2005.

[23] E. Prieto and C. Sloper. Either/or: Using vertex cover structure in designing FPT-algorithms — the case of k-internal spanning tree. In *Proc. of the* 8^{th} *Workshop on Algorithms and Data Structures (WADS 2003)*, volume 2748 of *LNCS*, pages 465–483, July 2003.

[24] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[25] C. Savage. Depth-first search and the Vertex Cover problem. *Information Processing Letters*, 14:233–235, 1982.

[26] K. M. Sivalingam and S. Subramaniam. *Optical WDM Networks: Principles and Practice*. Kluwer Academic Publishers, London, 2000.

[27] B. Y. Wu and K.-M. Chao. *Spanning Trees and Optimization Problems*. Chapman & Hall / CRC, 2004.

[28] S. Zhang and Z. Wang. Scattering number in graphs. *Networks*, 37:102–106, 2001.