(6)

Interactive Proof System Variants and Approximation Algorithms for Optical Networks

by

Ravi Sundaram

B. Tech., IIT Madras India (1991)
S.M., MIT Cambridge USA (1993)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1996
[June 1996]

Author .....
Department of Electrical Engineering and Computer Science
May 03, 1996

Certified by .....
Michael Sipser
Professor of Mathematics
Thesis Supervisor

Accepted by .....
F. R. Morgenthaler
Chair, Department Committee on Graduate Students

Interactive Proof System Variants and Approximation Algorithms for Optical Networks

by

Ravi Sundaram

Submitted to the Department of Electrical Engineering and Computer Science
on May 03, 1996, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

The discovery of a surprising connection between interactive proof systems and nonapproximability of combinatorial optimization problems has given great impetus to the field of theoretical computer science. In this thesis we focus on the class of optimization problems arising from the design and use of optical networks and some alternate models of interactive proof systems.

Optical networks are increasingly coming to be regarded as the technology of choice for the next generation of wide-area backbone networks. The two issues of algorithmic importance that arise here are: (a) designing good networks, and (b) (once the network has been designed) devising good routing algorithms.

Optical networks must be designed to minimize the total cost of fiber. The network must also satisfy the additional constraint that every site is serviced, i.e., every site is within a prespecified distance of an optoelectronic access node. We propose a general and robust bicriteria network design framework for modeling and solving such multiobective optimization problems e.g., diameter-constrained minimum-cost Steiner tree problem, service-constrained minimum-cost Steiner tree problem, etc. We give efficient approximation algorithms and corresponding hardness of approximation results. These results also have applications to the problem of designing networks capable of carrying multimedia traffic in a multicast environment.

We consider routing algorithms that conserve bandwidth both at a local and a global level. All-optical wavelength-division multiplexed networks employ Latin routers for assigning wavelengths to routes. We present fast algorithms for maximizing wavelength utilization at Latin routers. We also consider the problem of wavelength conservation on topologies such as trees and fixed networks.

The hardness of approximation results we obtain for the above problems derive from standard interactive proof systems. In an attempt to develop superior lower bounds, we study alternate models of interactive proof systems. The standard model posits a situation in which the provers collaborate in their attempt to convince the verifier. We consider a variant to this situation – one in which the provers compete against each other. There are two possible models here: (a) an informationally symmetric model, and (b) an informationally asymmetric model. We study both these models and obtain characterizations of the language classes accepted by them in terms of more standard classes.

Thesis Supervisor: Michael Sipser
Title: Professor of Mathematics

Acknowledgments

``The time has come," Mike Sipser said
``To talk of many things
Papers, postdocs and making tracks
Fiber-optics and rings
And why the job market is tight
And the money it brings."

``Your advice, your guidance," I cried
``Deeply grateful for that.
Your insight, your intuition -
I just take off my hat.
And though I graduate, I hope
We continue to chat."

Alex, I will miss you the most.
Squash games, our daily run,
The problems solved, papers written,
Foreign trips - so much fun;
Eight peaks over eight thousand metres.
We will climb at least one.

I thank Steve, Adi, Eric, Andrew Chou,
Wolf, and Marcos Kiwi,
Mike Klugerman, Ravi Kumar.
Other friends you do not see.
Only those names you read above
That with this rhyme agree.

Madhav, Achla, little Darrburr,
Nooks, Satyan, Anu, and pSki
R. Ravi, Chaks, and Mauricio -
I thank - Ron and Shafi
Although busy, agree they did
To be on my committee.

Appa, Amma, and Radha - thanks
You support all I do.
``O Reader - this acknowledgement
I did write just for you.
The rest of this boring thesis
Will make you turn blue."

Contents

List of Figures

Chapter 1

Introduction

1.1 Optical Networks

Developments in fiber-optic networking technology using *wavelength division multiplexing* (WDM) have finally reached the point where it is considered the most promising candidate for the next generation of wide-area backbone networks. These are projected to be highly flexible networks capable of supporting tens of thousands of users and providing capacities on the order of gigabits-per-second per user [CNW90, Gre92, Ram93].

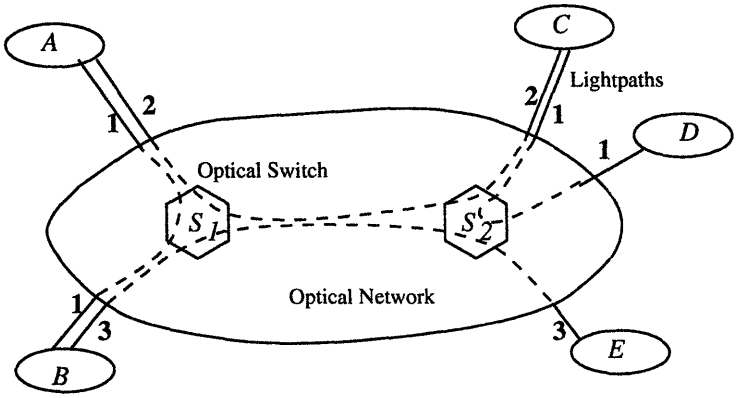

Figure 1-1: Example of an all-optical wavelength routing network

The WDM optical network is a pure data transmission medium. All the computing and processing continues to be done in the electronic world. The typical optical network consists of routing nodes interconnected by point-to-point fiber-optic links (See Fig. 1-1). The large bandwidth available in these fiber-optic links is utilized by partitioning it into several channels each at a different optical wavelength [BH92, CNW90, IEE93, IK92]. A point-to-point channel between two nodes on a specific wavelength (e.g., the channel between A and B on wavelength 1 in Fig.1-1) is referred to as a *lightpath*. The routing nodes or *optical switches* (See Fig. 1-2 are capable of setting up these lightpaths employing photonic switching [CB95, CGK92, ZA94]. The dynamic assignment of wavelengths, from an incoming port to an outgoing port of an optical switch, is

controlled electronically.

The deployment of fiber-optic technology on a large scale for use in long-haul wide-area networks has brought into sharp relief questions related to the design and use of these networks.

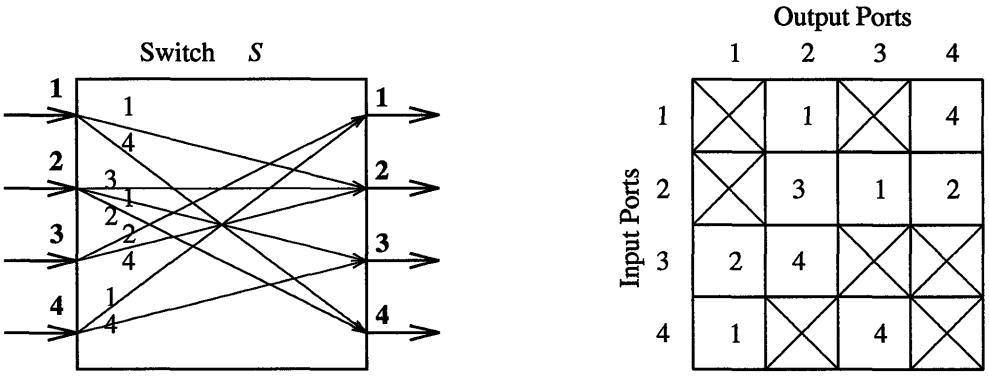

Figure 1-2: Example of optical switch and port assignment

1.1.1 Design

The design of optical networks is an issue of great algorithmic importance. Standard network design algorithms fail to provide effective solutions since the design of optical networks for wide-area backbones often involves the simultaneous minimization of multiple measures under different cost functions. Network design problems where even one cost measure must be minimized are often **NP**-hard [GJ79a]. We illustrate the nature of these *multicriteria* problems by providing two specific examples.

An important issue in interfacing these two worlds – the electronic and the optic – is that of designing the optical network subject to location-based constraints imposed by the electronic world. Given a set of sites in a network we wish to select a subset of the sites at which to place optoelectronic switches and routers [KRS96b]. The major requirement is that every site should be within a prespecified distance (under one cost measure) from an optoelectronic access node and the chosen sites should be connected together using fiber-optic links as a minimum cost tree (under another cost measure).

The carrying of multimedia (both audio and video) traffic is one of the primary uses of these wide-area networks. Our second example is motivated by this application. The total cost of the fiber is an important measure that must be minimized. In addition, since multimedia data needs to be delivered in real time, the point-to-point transmission delay must be minimized. As argued in [KPP92a], one of the popular solutions to this problem involves tree construction where two optimization criteria – (a) the minimum worst-case transmission delay, and (b) the minimum total cost – are sought to be minimized.

Motivated by the above examples, we consider a general and robust framework in which to study both multicriteria optimization problems as well as the quality of approximation algorithms.

1.1.2 Routing

Bandwidth is a very valuable resource in WDM optical networks. The problem of finding an optimal assignment of wavelengths to requests is of fundamental importance in bandwidth utilization. We consider the problem of conserving bandwidth both at the level of an individual switch and at the level of the network.

Conflict-free wavelength routing in wide-area optical networks is achieved by utilizing *Latin routers* [BH93]. These are routing devices that employ the concept of a *Latin square* (LS) (See Fig. 1-3). A partial Latin square L is an $n \times n$ matrix with entries from the set $\{0\} \bigcup \{1, 2, \ldots, n\}$ such that no nonzero number is repeated twice in the same row or in the same column. Latin squares naturally model the state of optical switches with the (i, j)-th entry denoting that wavelength L_{ij} is routed from input port i to output port j (0 is used as a placeholder to denote emptiness). The requirement that there be no wavelength conflict at the input or output ports is captured in the property that no value repeats in any row or column. We consider the question of conserving bandwidth at these Latin routers.

1	2	4	3
2	4	3	1
3	1	2	4
4	3	1	2

Figure 1-3: A 4×4 LS

We also consider the problem of wavelength utilization across the entire network. Though the total number of wavelengths is limited, nevertheless, it is possible to build a wide-area optical network by spatial reuse of wavelengths. For instance there are only three usable wavelengths in Fig. 1-1, yet it is still possible to set up all the shown interconnections by using wavelength **1** both for the $A - B$ and $C - D$ routes. We show that the problem of minimizing the total number of wavelengths used is solvable for constant degree undirected trees and constant size networks but is hard for constant degree directed trees.

1.2 Approximations and Nonapproximability: A Brief History

The last few years have seen major advances in our understanding of performance guarantees for approximation algorithms for NP-hard combinatorial optimization problems. Essentially for the twenty years following the path-breaking work of Cook [Coo71], Karp [Kar72], and Levin [Lev73], there was no progress. Recently new techniques have been developed for proving that certain approximation algorithms are unlikely to exist. This in turn has spurred significant recent advances in the design and analysis of approximation algorithms. We present a short summary of the history of work in this area.

Computational complexity was born in a 1965 paper of Hartmanis and Stearns [HS65]. Edmonds [Edm65]

formulated the notion of polynomial time **P** by defining an *efficient* algorithm to be one that runs in time polynomial in the size of the input. Nondeterministic time assumed significance when Cook [Coo71] showed that satisfiability is as hard as any other problem in **NP**, nondeterministic polynomial time. Soon after, Karp [Kar72] showed that a number of well-known combinatorial problems also shared this property. Since these problems, which include traveling salesman, clique, set cover, and others, have great practical significance, the **P** vs **NP** question has become the most famous open problem of theoretical computer science.

Following the recent discovery of **NP**-completeness, Johnson [Joh74b, Joh74a] suggested that one approach to coping with the intractability of a problem is to design and analyze approximation algorithms. He considered a variety of problems that have since become the central questions of the area. This work of Johnson was followed by a spate of approximation algorithms for a bunch of combinatorial optimization problems [Hoc95].

In the mid-to-late 1980s the spate of approximation results had slowed to a trickle. But there was excitement on the other side of the fence in the complexity theory community. Interactive proof systems were coming to be established as a significant area of study. This line of research owed its origins to both the notions of probabilistic computations [Gil77] and modern cryptography [DH76]. The formal interactive proof system model was developed concurrently by Goldwasser, Micali, and Rackoff [GMR85] and Babai [Bab85]. Great excitement was aroused by the discovery of a most surprising connection between interactive proof systems and approximation algorithms by Feige, Goldwasser, Lovasz, Safra, and Szegedy [FGL+91a]. This line of research culminated in the celebrated result of Arora, Lund, Motwani, Sudan, and Szegedy [ALM+92a] that all languages in **NP** can be accepted by a verifier with access to logarithmic randomness and only a constant number of answer bits. Improved hardness of approximation results were obtained by a number of researchers for clique, 3-SAT, set cover and a host of other problems. We do not attempt to detail the current state of this area since it is in a rapid state of flux.

1.3 Competing Provers

We proved hardness of approximation results for the problems obtained from the area of optical networks by reductions from set cover and standard interactive proof systems. However, the hardness factors obtained do not quite match the factors obtained by the corresponding approximation algorithms. In an attempt to improve the hardness factors we study alternate models of interactive proof systems.

The standard study of multi-prover interactive proof systems have concentrated on *cooperating* provers. These are systems in which a verifier interacts with a set of provers who collaborate in their attempt to convince the verifier that a word is in a prespecified language. Results on probabilistically checkable proofs coupled with parallel repetition techniques characterize **NP** in terms of multi-prover proof systems: languages in **NP** can be verified through a one round interaction with two cooperating provers using limited randomness and communication.

Less attention has been paid to the study of *competition* in the complexity theoretic setting of interactive proof systems. An interesting question is the following: consider one-round proof systems where the first prover is trying to convince the verifier to accept and the second prover is trying to make the verifier reject. What class of languages is characterized by this system? There are two possible models here - (a) an informationally symmetric model, and (b) an informationally asymmetric model. It is an interesting problem to study these two systems with a view to obtaining a characterization of the classes of languages accepted, in terms of more conventional complexity classes. The ultimate hope of this endeavor is to obtain new and improved hardness results from these proof system variants.

1.4 Summary of Thesis

In Chapter 2, motivated by applications to construction of optical networks, we study a general class of bicriteria network design problems. A generic problem in this class is as follows: given an undirected graph and two minimization objectives (under different cost functions), with a budget specified on the first, find a subgraph from a given subgraph class that minimizes the second objective subject to the budget on the first. We consider three different criteria – the total edge cost, the diameter, and the maximum degree of the network. We present the first polynomial-time approximation algorithms for a large class of bicriteria network design problems for the above mentioned criteria.

In Chapter 3 we continue our study of optical network design problems in the bicriteria framework set up in Chapter 2. We focus on the following problem (and its variants): find a low-cost network, under one cost function, that *services* every node in the graph, under another cost function, (i.e., every node of the graph is within a prespecified distance from the network). This study has important applications to the problems of optical network design and the efficient maintenance of distributed databases.

In Chapter 4 we consider the problem of conserving bandwidth on optical networks. These networks incorporate optical switches that employ the concept of Latin routers [BH93] for assigning wavelengths to routes. The issue of maximizing wavelength utilization at these switching devices is of great importance since it leads to significant improvements in overall network performance [CB95]. We present two fast approximation algorithms for the problem of maximizing wavelength utilization at Latin routers. These algorithms are easily implementable and have very small constants in their running times making them eminently suitable for actual use in real-world optical switches. We provide strong experimental evidence to show that, in practice, these algorithms are near-optimal. We also consider the problem of conserving bandwidth across the entire network as opposed to just optimizing at the level of a router. We present a polynomial-time algorithm for this problem on fixed constant-size topologies. We combine this algorithm with ideas from Raghavan and Upfal [RU94] to obtain an optimal assignment of wavelengths on constant degree *undirected* trees. Mihail, Kaklamanis and Rao [MKR95] posed the following open question: what is the complexity of this problem on *directed* trees? We show that it is **NP**-hard even for constant degree directed trees.

In Chapter 5 we consider one-round proof systems where the first prover is trying to convince the verifier to accept and the second prover is trying to make the verifier reject. We build into these proof systems a crucial *asymmetry* between the provers. We show that such proof systems capture, with restrictions on communication and randomness, languages in $\mathbf{NP}$. We generalize this model by defining alternating prover proof systems. We show that they characterize every level of the polynomial hierarchy. Alternating oracle proof systems are also examined. We provide the first exact characterization of the polynomial hierarchy in terms of interactive proof systems.

In Chapter 6 we introduce the natural class $\mathbf{S}_2^P$ containing those languages which may be expressed in terms of two *symmetric quantifiers*. This class lies between Δ_2^P and $\Sigma_2^P \cap \Pi_2^P$ and naturally generates a "symmetric" hierarchy corresponding to the polynomial-time hierarchy. We demonstrate, using the probabilistic method, new containment theorems for $\mathbf{BPP}$. We show that $\mathbf{MA}$ (and hence $\mathbf{BPP}$) lies within $\mathbf{S}_2^P$, improving the constructions of [Sip83, Lau83] (which show that $\mathbf{BPP} \subset \Sigma_2^P \cap \Pi_2^P$). Symmetric alternation is shown to enjoy two strong structural properties which are used to prove the desired containment results. We offer some evidence that $\mathbf{S}_2^P \neq \Sigma_2^P \cap \Pi_2^P$ by demonstrating an oracle so that $\mathbf{S}_2^{P,O} \neq \Sigma_2^{P,O} \cap \Pi_2^{P,O}$ assuming that the machines make only "positive" oracle queries.

This thesis contains the results of four published papers and three unpublished manuscripts. Chapters 2 and 3 are based on [MRS+95] and [MRS96], respectively, and represent joint work with Madhav Marathe, R. Ravi and others. Chapter 4 is based on [KRS96a], [KRS96b], and [KPRS96] and is joint research with Ravi Kumar, Rina Panigrahy, and Alex Russell. Chapter 5 is based on joint research with Marcos Kiwi, Carsten Lund, Alex Russell, and Dan Spielman,[KLR+94]. Chapter 6 is joint work with Alex Russell [RS95].

Chapter 2

Bicriteria Network Design Problems

2.1 Motivation

With the information superhighway fast becoming a reality, the problem of designing networks capable of accommodating multimedia (both audio and video) traffic in a multicast (simultaneous transmission of data to multiple destinations) environment has come to assume paramount importance [Cho91, FWB85, KJ83, KPP92a, KPP92b, KPP93]. Fiber-optic networks are increasingly coming to be seen as the technology of choice for these networks. As argued in [KPP92a], one of the popular solutions to multicast routing involves tree construction. Since optical networks are circuit switched (as opposed to conventional networks that are packet switched) two optimization criteria under two very different cost functions come into play. The two optimization criteria are – (a) the minimum worst-case transmission delay, and (b) the minimum total construction cost. The issue of minimizing transmission delay takes on additional importance since multimedia data must be transmitted in real-time.

Network design problems where even one cost measure must be minimized, are often **NP**-hard [GJ79a]. But, in real-life applications, it is often the case that the network to be built is required to minimize multiple cost measures simultaneously, with different cost functions for each measure. For example, as pointed out in [KPP92a], in the problem of finding good multicast trees, each edge has associated with it two edge costs: the construction cost and the delay cost. The construction cost is typically a measure of the amount of buffer space or channel bandwidth used and the delay cost is a combination of the propagation, transmission, and queueing delays.

Such multi-criteria network design problems, with separate cost functions for each optimization criterion, also occur naturally in VLSI designs (see [ZPD94] and the references therein). With the advent of deep micron VLSI designs, the feature size has shrunk to sizes of 0.5 microns and less. As a result, the interconnect resistance, being proportional to the square of the scaling factor, has increased significantly. An increase in interconnect resistance has led to an increase in interconnect delays thus making them a dominant factor in the

timing analysis of VLSI circuits. Therefore VLSI circuit designers aim at finding minimum cost (spanning or Steiner) trees given delay bound constraints on source-sink connections.

The above applications set the stage for the formal definition of multicriteria network design problems. We explain this concept by giving a formal definition of a bicriteria network design problem. A generic bicriteria network design problem, (**A**, **B**, **S**), is defined by identifying two minimization objectives, **A** and **B**, from a set of possible objectives, and specifying a membership requirement in a class of subgraphs, **S**. The problem specifies a budget value on the first objective, **A**, under one cost function, and seeks to find a network having minimum possible value for the second objective, **B**, under another cost function, such that this network is within the budget on the first objective. The solution network must belong to the subgraph class **S**. For example, the problem of finding low-cost and low-transmission-delay multimedia networks [KPP92a, KPP93] can be modeled as the (Diameter, Total cost, Spanning tree)-bicriteria problem: given an undirected graph $G = (V, E)$ with two different weight functions c_e (modeling the construction cost) and d_e (modeling the delay cost) for each edge $e \in E$, and a bound $\mathcal{D}$ (on the total delay), find a minimum c-cost spanning tree such that the diameter of the tree under the d-costs is at most $\mathcal{D}$. It is easy to see that the notion of bicriteria optimization problems can be easily extended to the more general multicriteria optimization problems. In this chapter, we will be mainly concerned with bicriteria network design problems.

In the past, the problem of minimizing two cost measures was often dealt with by attempting to minimize some combination of the two, thus converting it into a unicriterion problem. This approach often fails, especially when the two criteria are very disparate. Therefore we have chosen to model bicriteria problems as that of minimizing one criterion subject to a budget on the other. We argue that this approach is both general as well as robust. It is more general because it subsumes the case where one wishes to minimize some functional combination of the two criteria. It is more robust because the quality of approximation is independent of which of the two criteria we impose the budget on. We elaborate on this more in Sections 2.5.1 and 2.5.2.

The organization of the rest of the chapter is as follows: Section 2.2 summarizes the results obtained in this chapter; Section 2.3 discusses related research work; Section 2.4 contains the hardness results; Section 2.5.1 shows that the two alternative ways of formulating a bicriteria problem are indeed equivalent; Section 2.5.2 demonstrates the generality of the bicriteria approach; Section 2.6 details the parametric search technique; Section 2.7 presents the approximation algorithm for diameter constrained Steiner trees; Section 2.8 contains the results on treewidth-bounded graphs; Section 2.9 contains some concluding remarks and open problems.

2.2 Summary of Results

In this chapter, we study the complexity and approximability of a number of bicriteria network design problems. The three objectives we consider are: (a) total cost, (b) diameter, and (c) degree of the network. These reflect the price of synthesizing the network, the maximum delay between two points in the network and the reliability of the network, respectively. The *Total cost* objective is the sum of the costs of all the edges in the

subgraph. The *Diameter* objective is the maximum distance between any pair of nodes in the subgraph. The *Degree* objective denotes the maximum over all nodes in the subgraph, of the degree of the node. The class of subgraphs we consider in this chapter are mainly *Steiner trees* (and hence *Spanning trees* as a special case); although several of our results extend to more general connected subgraphs such as generalized Steiner trees, etc.

As mentioned in [GJ79a], most of the problems considered in this chapter are **NP**-hard for arbitrary instances even when we wish to find optimum solutions with respect to a single criterion. Given the hardness of finding optimal solutions, we concentrate on devising approximation algorithms with worst case performance guarantees. Recall that an approximation algorithm for a minimization problem Π provides a **performance guarantee** of ρ if for every instance I of Π, the solution value returned by the approximation algorithm is within a factor ρ of the optimal value for I. Here, we extend this notion to apply to bicriteria optimization problems. An (α, β)-approximation algorithm for an $(\mathbf{A}, \mathbf{B}, \mathbf{S})$-bicriteria problem is defined as a polynomial-time algorithm that produces a solution in which the first objective ($\mathbf{A}$) value, is at most α times the budget, and the second objective ($\mathbf{B}$) value, is at most β times the minimum for any solution that is within the budget on $\mathbf{A}$. The solution produced must belong to the subgraph class $\mathbf{S}$. Analogous definitions can be given when $\mathbf{A}$ and/or $\mathbf{B}$ are maximization objectives.

2.2.1 General Graphs

Table 1 contains the performance guarantees of our approximation algorithms for finding spanning trees, $\mathbf{S}$, under different pairs of minimization objectives, $\mathbf{A}$ and $\mathbf{B}$; asterisks indicate results obtained in this chapter; $\gamma > 0$ is a fixed accuracy parameter. For each problem cataloged in the table, two different costs are specified on the edges of the undirected graph: the first objective is computed using the first cost function and the second objective, using the second cost function. The rows are indexed by the budgeted objective. For example the entry in row $\mathbf{A}$, column $\mathbf{B}$, denotes the performance guarantee for the problem of minimizing objective $\mathbf{B}$ with a budget on the objective $\mathbf{A}$. All the results in Table 1 extend to finding Steiner trees with at most a constant factor worsening in the performance ratios. For the diagonal entries in the table the extension to Steiner trees follows from Theorem 6. ALGORITHM DCST of Section 2.7 in conjunction with ALGORITHM BICRITERIA-EQUIVALENCE of Section 2.5.1 yields the (Diameter, Total cost, Steiner tree) and (Total cost, Diameter, Steiner tree) entries. The other nondiagonal entries can also be extended to Steiner trees and these extensions will appear in the journal versions of [RMR+93, Rav94]. Our results for arbitrary graphs can be divided into three general categories.

Cost Measures	Degree	Diameter	Total Cost
Degree	$(O(\log n), O(\log n))^*$	$(O(\log^2 n), O(\log n))$[Rav94]	$(O(\log n), O(\log n))$[RMR+93]
Diameter	$(O(\log n), O(\log^2 n))$[Rav94]	$(1+\gamma, 1+\frac{1}{\gamma})^*$	$(O(\log n), O(\log n))^*$
Total Cost	$(O(\log n), O(\log n))$[RMR+93]	$(O(\log n), O(\log n))^*$	$(1+\gamma, 1+\frac{1}{\gamma})^*$

Table 1. Performance Guarantees for finding spanning trees in an arbitrary graph on n nodes.

First, as mentioned before, there are two natural alternative ways of formulating general bicriteria problems: (i) where we impose the budget on the first objective and seek to minimize the second, and (ii) where we impose the budget on the second objective and seek to minimize the first. We show that an (α, β)-approximation algorithm for one of these formulations naturally leads to a (β, α)-approximation algorithm for the other. Thus our definition of a bicriteria approximation is independent of the choice of the criterion that is budgeted in the formulation. This makes it a robust definition and allows us to fill in the entries for the problems (**B**, **A**, **S**) by transforming the results for the corresponding problems (**A**, **B**, **S**).

The diagonal entries in the table follow as a corollary of a general result (Theorem 6) which is proved using a parametric search algorithm. The entry for (Degree, Degree, Spanning tree) follows by combining Theorem 6 with the $O(\log n)$-approximation algorithm for the degree problem in [RMR$^+$93]. In [RMR$^+$93] they actually provide an $O(\log n)$-approximation algorithm for the weighted degree problem. The weighted degree of a subgraph is defined as the maximum over all nodes of the sum of the costs of the edges incident on the node in the subgraph. Hence we actually get an $(O(\log n), O(\log n))$-approximation algorithm for the (Weighted degree, Weighted degree, Spanning tree)-bicriteria problem. Similarly, the entry for (Diameter, Diameter, Spanning tree) follows by combining Theorem 6 with the known exact algorithms for minimum diameter spanning trees [CG82, RSM$^+$94]; while the result for (Total cost, Total cost, Spanning tree) follows by combining Theorem 6 with an exact algorithm to compute a minimum spanning tree [Kru56, Pri57]. Theorem 6 is very general. Given *any* ρ-approximation algorithm for minimizing the objective **A** in the subgraph class **S**, Theorem 6 allows us to produce a $(2\rho, 2\rho)$-approximation algorithm for the (**A**, **A**, **S**)-bicriteria problem. Consider, for example, the conjunction of this theorem with the results of Goemans et al [GGP$^+$94]. This leads to a host of bicriteria approximation results when two costs are specified on edges for finding minimum-cost generalized Steiner trees, minimum k-edge connected subgraphs, or any other network design problems specified by weakly supermodular functions. Thus for example, we get $(O(1), O(1))$-approximation algorithms for the (Total cost, Total cost, Generalized Steiner tree) and (Total cost, Total cost, k-edge connected subgraph)-bicriteria problems. (See [AKR95, GW92, KV94] for the results on the corresponding unicriterion problems.) Similarly, given an undirected graph with two costs specified on each node we can get logarithmic approximations for the minimum node-cost Steiner tree using the result of Klein and Ravi [KR95]. As another example, with two edge-cost functions, and an input number k, we can use the result of Blum et al. [BRV96] to obtain an $(O(1), O(1))$-approximation for the minimum cost (under both functions) tree spanning at least k nodes.

Finally, we present a cluster based approximation algorithm and a solution based decomposition technique for devising approximation algorithms for problems when the two objectives are different. Our techniques yield $(O(\log n), O(\log n))$-approximation algorithms for the (Diameter, Total cost, Steiner tree) and the (Degree, Total cost, Steiner tree) problems[1].

[1] The result for (Degree, Total cost, Steiner tree) can also be obtained as a corollary of the results in [RMR$^+$93].

2.2.2 Treewidth-Bounded Graphs

We also study the bicriteria problems mentioned above for the class of treewidth-bounded graphs. Examples of treewidth-bounded graphs include trees, series-parallel graphs, k-outerplanar graphs, chordal graphs with cliques of size at most k, bounded-bandwidth graphs, etc. We use a dynamic programming technique to show that for the class of treewidth-bounded graphs, there are either polynomial-time or pseudopolynomial-time algorithms (when the problem is **NP**-complete) for several of the bicriteria network design problems studied here. Recall that a **polynomial time approximation scheme** (**PTAS**) for problem Π is a family of algorithms $\mathcal{A}$ such that, for all $\epsilon > 0$, given an instance I of Π, there is a polynomial time algorithm $A \in \mathcal{A}$ that returns a solution which is within a factor $(1+\epsilon)$ of the optimal value for I. A **PTAS** in which the running time grows as a polynomial function of ϵ is called a **fully polynomial time approximation scheme** (**FPAS**). Here we show how to convert these pseudopolynomial-time algorithms for problems restricted to treewidth-bounded graphs into **FPAS**s using a general scaling technique. Stated in our notation, we obtain **PTAS**s with performance of $(1, 1+\epsilon)$, for all $\epsilon > 0$. The results for treewidth-bounded graphs are summarized in Table 2. As before, the rows are indexed by the budgeted objective. All algorithmic results in Table 2 also extend to Steiner trees in a straightforward way.

Our results for treewidth-bounded graphs have an interesting application in the context of finding optimum broadcast schemes. Kortsarz and Peleg [KP95] give $O(\log n)$-approximation algorithms for the minimum broadcast time problem for series-parallel graphs. Combining our results for the (Degree, Diameter, Spanning tree) for treewidth-bounded graphs with the techniques in [Rav94], we obtain an $O(\frac{\log n}{\log \log n})$-approximation algorithm for the minimum broadcast time problem for treewidth-bounded graphs (series-parallel graphs have a treewidth of 2), improving and generalizing the result in [KP95].

Cost Measures	Degree	Diameter	Total Cost
Degree	polynomial-time	polynomial-time	polynomial-time
Diameter	polynomial-time	(weakly NP-hard) $(1, 1+\epsilon)$	(weakly NP-hard) $(1, 1+\epsilon)$
Total Cost	polynomial-time	(weakly NP-hard) $(1, 1+\epsilon)$	(weakly NP-hard) $(1, 1+\epsilon)$

Table 2. Bicriteria spanning tree results for treewidth-bounded graphs.

2.3 Previous Work

2.3.1 General Graphs

The area of unicriterion optimization problems for network design is vast and well-explored (See [Hoc95, CK95] and the references therein.). Ravi et al. [RMR+93] studied the degree-bounded minimum cost span-

ning tree problem and provided an approximation algorithm with performance guarantee $(O(\log n), O(\log n))$. Though they were doing bicriteria optimization they did not state it as such in their paper.

The (Degree, Diameter, Spanning tree) problem was studied in [Rav94] in the context of finding good broadcast networks. There he provides an approximation algorithm for the (Degree, Diameter, Spanning tree) problem with performance guarantee $(O(\log^2 n), O(\log n))$[2].

The (Diameter, Total cost, Spanning tree) entry in Table 1 corresponds to the diameter-constrained minimum spanning tree problem introduced earlier. It is known that this problem is **NP**-hard even in the special case where the two cost functions are identical [HLCW89]. Awerbuch, Baratz, and Peleg [ABP90] gave an approximation algorithm with $(O(1), O(1))$ performance guarantee for this problem – i.e., the problem of finding a spanning tree that has simultaneously small diameter (i.e., shallow) and small total cost (i.e., light), both under the same cost function. Khuller, Raghavachari, and Young [KRY95] studied an extension called *Light, approximate Shortest-path Trees (LAST)* and gave an approximation algorithm with $(O(1), O(1))$ performance guarantee. Kadaba and Jaffe [KJ83], Kompella et al. [KPP92a], and Zhu et al. [ZPD94] considered the (Diameter, Total cost, Steiner tree) problem with two edge costs and presented heuristics without any guarantees. It is easy to construct examples to show that the solutions produced by these heuristics in [ZPD94, KPP92a], can be arbitrarily bad with respect to an optimal solution. A closely related problem is that of finding a diameter-constrained shortest path between two pre-specified vertices s and t, or (Diameter, Total cost, s-t path). This problem, termed the multi-objective shortest path problem (MOSP) in the literature, is **NP**-complete and Warburton [War87] presented the first **FPAS** for it. Hassin [Has92] provided a strongly polynomial **FPAS** for the problem which improved the running time of Warburton [War87]. This result was further improved by Philips [Phi93].

The (Total cost, Total cost, Spanning tree)-bicriteria problem has been recently studied by Ganley et al. [GGS95]. They consider a more general problem with more than two weight functions. They also gave approximation algorithms for the restricted case when each weight function obeys triangle inequality. However, their algorithm does not have a bounded performance guarantee with respect to each objective.

2.3.2 Treewidth-Bounded Graphs

Treewidth-bounded graphs were introduced by Robertson and Seymour [RS86]. Many **NP**-hard problems have exact solutions when attention is restricted to the class of treewidth-bounded graphs and much work has been done in this area (See [ACPS93, ALS91, BLW87] and the references therein). Independently, Bern, Lawler and Wong [BLW87] introduced the notion of decomposable graphs. Later, it was shown [ACPS93] that the class of decomposable graphs and the class of treewidth-bounded graphs are equivalent. Bicriteria network design problems restricted to treewidth-bounded graphs have been previously studied in [ALS91, Bod88].

[2]The result in [Rav94] is actually somewhat stronger - given a budget, D, on the degree he finds a tree whose total cost is at most $O(\log n)$ times the optimal and whose degree is at most $O(D \log n + \log^2 n)$.

2.4 Hardness results

The problem of finding a minimum degree spanning tree is strongly **NP**-hard [GJ79a]. This implies that all spanning tree bicriteria problems, where one of the criteria is degree, are also strongly **NP**-hard. In contrast, it is well known that the minimum diameter spanning tree problem and the minimum cost spanning tree problems have polynomial time algorithms [CG82, HLCW89, Kru56, Pri57, RSM+94].

The (Diameter, Total Cost, Spanning tree)-bicriteria problem is strongly **NP**-hard even in the case where both cost functions are identical [HLCW89]. Here we give the details of the reduction to show that (Diameter, Total Cost, Spanning tree) is weakly **NP**-hard even for series-parallel graphs (i.e., graphs with treewidth at most 2 [Bod92]). Similar reductions can be given to show that (Diameter, Diameter, Spanning tree) and (Total cost, Total cost, Spanning tree) are also weakly **NP**-hard for series-parallel graphs.

We first recall the definition of the **PARTITION** problem [GJ79a]. As an instance of the **PARTITION** problem we are given a set $T = \{t_1, t_2, \cdots, t_n\}$ of positive integers and the question is whether there exists a subset $X \subseteq A$ such that $\sum_{t_i \in X} t_i = \sum_{t_j \in T-X} t_j = (\sum_{t_j \in T} t_j)/2$.

Theorem 1 *(Diameter, Total cost, Spanning tree) is* **NP***-hard for series-parallel graphs.*

Proof: Reduction from the **PARTITION** problem. Given an instance $T = \{t_1, t_2, \cdots, t_n\}$ of the **PARTITION** problem, we construct a series parallel graph G with $n+1$ vertices, $v_1, v_2, \cdots v_{n+1}$ and $2n$ edges. We attach a pair of parallel edges, e_i^1 and e_i^2, between v_i and v_{i+1}, $1 \leq i \leq n$. We now specify the two cost functions f and g on the edges of this graph; $c(e_i^1) = t_i, c(e_i^2) = 0, d(e_i^1) = 0, d(e_i^2) = t_i, 1 \leq i \leq n$ (see Figure 2-1). Let $\sum_{t_i \in T} t_i = 2H$. Now it is easy to show that G has a spanning tree of d-diameter at most H and total c-cost at most H if and only if there is a solution to the original instance T of the **PARTITION** problem. □

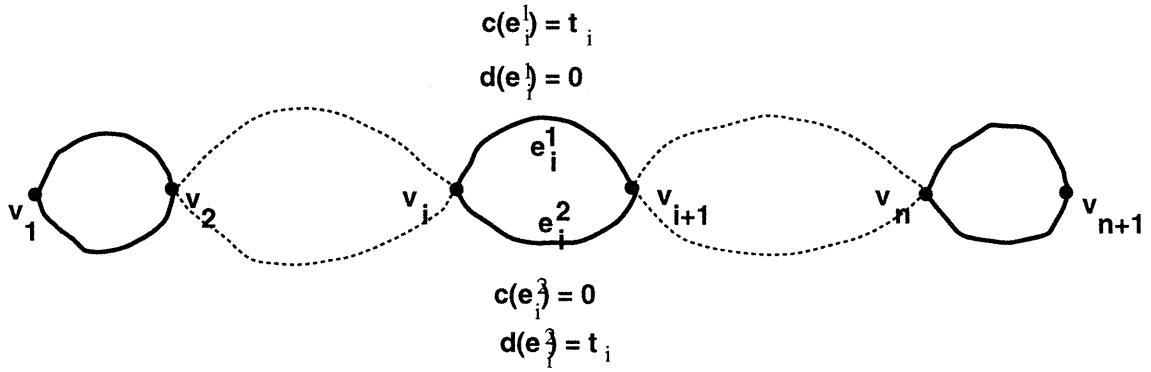

Figure 2-1: Reduction from the **PARTITION** problem to (Diameter, Total cost, Spanning tree) for series parallel graphs.

We now show that the (Diameter, Total-cost, Steiner tree) problem is hard to approximate within a logarithmic factor. This is in contrast to the approximation algorithm provided in Section 2.7. There is however a

gap between the results of Theorems 3 and 9. Our non-approximability result is obtained by an approximation preserving reduction from the **MIN SET COVER**. An instance (T, X) of the **MIN SET COVER** problem consists of a universe $T = \{t_1, t_2, \ldots, t_k\}$ and a collection of subsets $X = \{X_1, X_2, \ldots, X_m\}$, $X_i \subseteq T$, each set X_i having an associated cost c_i. The problem is to find a minimum cost collection of the subsets whose union is T. Recently Feige [Fei96] has shown the following non-approximability result:

Theorem 2 *Unless* $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\log \log n})$, *the* **MIN SET COVER** *problem, with a universe of size k, cannot be approximated to better than a* $\ln k$ *factor.*

Theorem 3 *Unless* $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\log \log n})$, *given an instance of the (Diameter, Total Cost, Steiner tree) problem with k sites, there is no polynomial-time approximation algorithm that outputs a Steiner tree of diameter at most the bound D, and cost at most R times that of the minimum cost diameter-D Steiner tree, for* $R < \ln k$.

Proof: The following is an approximation preserving reduction from the **MIN SET COVER** problem to the (Diameter, Total Cost, Steiner tree) problem. Given an instance (T, X) of the **MIN SET COVER** problem where $T = \{t_1, t_2, \ldots, t_k\}$ and $X = \{X_1, X_2, \ldots, X_m\}$, $X_i \subseteq T$, where the cost of the set X_i is c_i, we construct an instance G of the (Diameter, Total Cost, Steiner tree) problem as follows. The graph G has a node t_i for each element t_i of T (there is a mild abuse of notation here but it should not lead to any confusion.), a node x_i for each set X_i, and an extra "enforcer-node" n. For each set X_i, we attach an edge between nodes n and x_i of c-cost c_i, and d-cost 1. For each element t_i and set X_j such that $t_i \in X_j$ we attach an edge (t_i, x_j) of c-cost, 0, and d-cost, 1. In addition to these edges, we add a path P made of two edges of c-cost, 0, and d-cost, 1, to the enforcer node n (See Fig. 2-2. The c-cost of the edges are shown. All these edges have d-cost of 1.). All other edges in the graph are assigned infinite c and d-costs. The nodes t_i along with n and the two nodes of P are specified to be the terminals for the Steiner tree problem instance. We claim that G has a c-cost Steiner tree of diameter at most 4 and cost C if and only if the original instance (T, X) has a solution of cost C.

It is easy to see that any Steiner tree of diameter at most 4 must contain a path from t_i to n, for all i, that uses an edge (x_j, n) for some X_j such that $t_i \in X_j$. Hence any Steiner tree of diameter at most 4 provides a feasible solution of equivalent c-cost to the original Set cover instance. The proof follows from Theorem 2. □

2.5 Bicriteria Formulations: Properties

In Section 2.1, we claimed that our formulation for bicriteria problems is robust and general. In this section, we justify these claims.

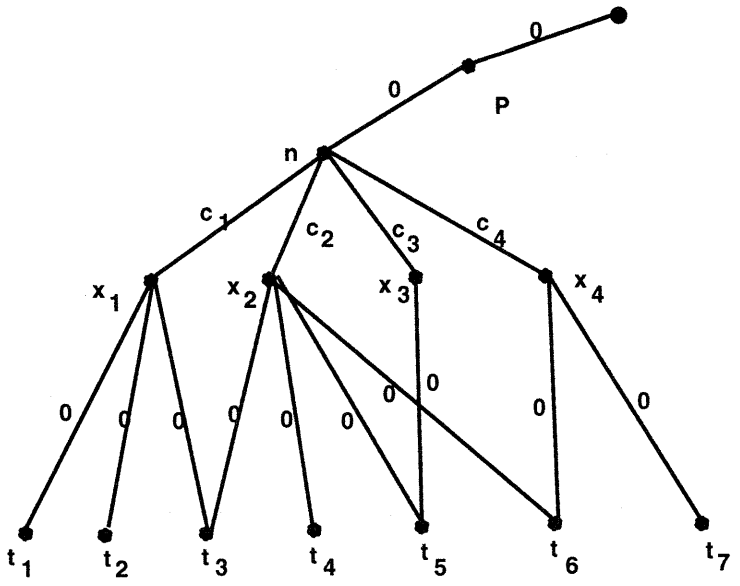

Figure 2-2: Reduction from the **MIN SET COVER** problem to (Diameter, Total cost, Steiner tree) problem.

2.5.1 Robustness

We claimed that our formulation is robust because the quality of approximation is independent of which of the two criteria we impose the budget on. We elaborate on this claim in the following.

Let G be a graph with two (integral)[3] cost functions, c and d. c and d are typically edge costs or node costs. Let $\mathbf{A}$ ($\mathbf{B}$) be a minimization objective computed using cost function c (d). Let the budget bound on the c-cost[4] (d-cost) of a solution subgraph be denoted by $\mathcal{C}$ ($\mathcal{D}$).

There are two natural ways to formulate a bicriteria problem: (i) ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{S}$) – find a subgraph in $\mathbf{S}$ whose $\mathbf{A}$-objective value (under the c-cost) is at most $\mathcal{C}$ and which has minimum $\mathbf{B}$-objective value (under the d-cost), (ii) ($\mathbf{B}$, $\mathbf{A}$, $\mathbf{S}$) – find a subgraph in $\mathbf{S}$ whose $\mathbf{B}$-objective value (under the d-cost) is at most $\mathcal{D}$ and which has minimum $\mathbf{A}$-objective value (under the c-cost).

Note that bicriteria problems are meaningful only when the two criteria are *hostile* with respect to each other – the minimization of one criterion conflicts with the minimization of the other. A good example of hostile objectives are the degree and the total edge cost of a spanning tree in an unweighted graph [RMR+93]. Two minimization criteria are formally defined to be hostile whenever the minimum value of one objective is monotonically nondecreasing as the budget (bound) on the value of the other objective is decreased.

Let $\mathbf{XYZ}(G, \mathcal{C})$ be any (α, β)-approximation algorithm for ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{S}$) on graph G with budget $\mathcal{C}$ under the c-cost. We now show that there is a transformation which produces a (β, α)-approximation algorithm for ($\mathbf{B}$, $\mathbf{A}$, $\mathbf{S}$). The transformation uses binary search on the range of values of the c-cost with an application of the given approximation algorithm, $\mathbf{XYZ}$, at each step of this search. Let the minimum c-cost of a $\mathcal{D}$-bounded

[3] In case of rational cost functions, our algorithms can be easily extended with a small additive loss in the performance guarantee.

[4] We use the term "cost under c" or "c-cost" in this section to mean the value of the objective function computed using c, and not to mean the total of all the c costs in the network.

subgraph in **S** be OPT_c. Let $\mathcal{C}_{hi}$ be an upper bound on the c-cost of any $\mathcal{D}$-bounded subgraph in **S**. Note that $\mathcal{C}_{hi}$ is at most some polynomial in n times the maximum c-cost (of an edge or a node). Hence $\log(\mathcal{C}_{hi})$ is at most a polynomial in terms of the input specification. Let HEU_c (HEU_d) denote the c-cost (d-cost) of the subgraph output by ALGORITHM BICRITERIA-EQUIVALENCE given below.

ALGORITHM BICRITERIA-EQUIVALENCE:

- *Input:* G - graph, $\mathcal{D}$ - budget on criterion **B** under the d-cost, **XYZ** - an (α, β)-approximation algorithm for (**A**, **B**, **S**).
- 1. Let $\mathcal{C}_{hi}$ be an upper bound on the c-cost of any $\mathcal{D}$-bounded subgraph in **S**.
  2. Do binary search and find a $\mathcal{C}'$ in $[0, \mathcal{C}_{hi}]$ such that
     (a) $\mathbf{XYZ}(G, \mathcal{C}')$ returns a subgraph with d-cost greater than $\beta\mathcal{D}$, and
     (b) $\mathbf{XYZ}(G, \mathcal{C}' + 1)$ returns a subgraph with d-cost at most $\beta\mathcal{D}$.
  3. If the binary search in Step 2 fails to find a valid $\mathcal{C}'$ then output "NO SOLUTION" else output $\mathbf{XYZ}(G, \mathcal{C}' + 1)$.
- *Output:* A subgraph from **S** such that its c-cost is at most α times that of the minimum c-cost $\mathcal{D}$-bounded subgraph and its d-cost is at most $\beta\mathcal{D}$.

Claim 1 *If G contains a $\mathcal{D}$-bounded subgraph in* **S** *then* ALGORITHM BICRITERIA-EQUIVALENCE *outputs a subgraph from* **S** *whose c-cost is at most α times that of the minimum c-cost $\mathcal{D}$-bounded subgraph and whose d-cost is at most $\beta\mathcal{D}$.*

Proof: Since **A** and **B** are hostile criteria it follows that the binary search in Step 2 is well defined. Assume that **S** contains a $\mathcal{D}$-bounded subgraph. Then, since $\mathbf{XYZ}(G, \mathcal{C}_{hi})$ returns a subgraph with d-cost at most $\beta\mathcal{D}$, it is clear that ALGORITHM BICRITERIA-EQUIVALENCE outputs a subgraph in this case. As a consequence of Step 2a and the performance guarantee of the approximation algorithm **XYZ**, we get that $\mathcal{C}' + 1 \leq \text{OPT}_c$. By Step 2b we have that $\text{HEU}_d \leq \beta\mathcal{D}$ and $\text{HEU}_c \leq \alpha(\mathcal{C}' + 1) \leq \alpha\text{OPT}_c$. Thus ALGORITHM BICRITERIA-EQUIVALENCE outputs a subgraph from **S** whose c-cost is at most α times that of the minimum c-cost $\mathcal{D}$-bounded subgraph and whose d-cost is at most $\beta\mathcal{D}$.

□

Note however that in general the resulting (β, α)-approximation algorithm is, not *strongly* polynomial since it depends on the range of the c-costs. But it is a *polynomial-time* algorithm since its running time is linearly dependent on $\log \mathcal{C}_{hi}$ the largest c-cost. The above discussion leads to the following theorem.

Theorem 4 *Any (α, β)-approximation algorithm for (***A**, **B**, **S***) can be transformed in polynomial time into a (β, α)-approximation algorithm for (***B**, **A**, **S***).*

2.5.2 Generality

In Section 2.1, we said that our formulation is more general because it subsumes the case where one wishes to minimize some functional combination of the two criteria. We briefly comment on this next. For the purposes of illustration let **A** and **B** be two objective functions and let us say that we wish to minimize the sum of the two objectives **A** and **B**. Call this an (**A** + **B**, **S**) problem. Let $\mathbf{XYZ}(G, \mathcal{C})$ be any (α, β)-approximation algorithm for (**A**, **B**, **S**) on graph G with budget $\mathcal{C}$ under the c-cost. We show that there is a polynomial time $\max\{\alpha, \beta\}$-approximation algorithm for the (**A** + **B**, **S**) problem. The transformation uses binary search on the range of values of the c-cost with an application of the given approximation algorithm, **XYZ**, at each step of this search. Let the optimum value for the (**A** + **B**, **S**) problem on a graph G be $\text{OPT}_{c+d} = (V_c + V_d)$, where V_c and V_d denote respectively the contribution of the two costs c and d for **A** and **B**. Let $\text{HEU}_c(\mathcal{C})$ ($\text{HEU}_d(\mathcal{C})$) denote the c-cost (d-cost) of the subgraph output by $\mathbf{XYZ}(G, \mathcal{C})$. Finally, let $\text{HEU}_{c+d}(\mathcal{C})$ denote the value computed by ALGORITHM CONVERT.

ALGORITHM CONVERT:

- *Input:* G - graph, **XYZ** - an (α, β)-approximation algorithm for (**A**, **B**, **S**).
- 1. Let $\mathcal{C}_{hi}$ be an upper bound on the c-cost of any subgraph in **S**.
  2. Employ binary search to find a $\mathcal{C}'$ in $[0, \mathcal{C}_{hi}]$ such that
     (a) $\mathbf{XYZ}(G, \mathcal{C}')$ returns a subgraph with the minimum value of $\text{HEU}_c(\mathcal{C}') + \text{HEU}_d(\mathcal{C}')$.
- *Output:* A subgraph from **S** such that the sum of its c-cost and its d-costs is at most $\max\{\alpha, \beta\}(\text{OPT}_{c+d})$.

Theorem 5 *Let* $\mathbf{XYZ}(G, \mathcal{C})$ *be any* (α, β)*-approximation algorithm for* (**A**, **B**, **S**) *on graph* G *with budget* $\mathcal{C}$ *under the* c*-cost. Then, there is a polynomial time* $\max\{\alpha, \beta\}$*-approximation algorithm for the* (**A** + **B**, **S**) *problem.*

Proof: Consider the iteration of the binary search in which the bound on the c-cost is V_c. Then as a consequence of the performance guarantee of the approximation algorithm **XYZ**, we get that $\text{HEU}_c(V_c) \leq \alpha V_c$. By Step 2a and the performance guarantee of the algorithm **XYZ**, we have that $\text{HEU}_d(V_c) \leq \beta V_d$. Thus $\text{HEU}_{c+d}(V_c) \leq \alpha V_c + \beta V_d \leq \max\{\alpha, \beta\}(V_c + V_d)$. Since ALGORITHM CONVERT outputs a subgraph from **S** the sum of whose c-cost and d-cost is minimized, we have that

$$\min_{\mathcal{C}' \in [0, \mathcal{C}_{hi}]} (\text{HEU}_c(\mathcal{C}') + \text{HEU}_d(\mathcal{C}')) \leq \max\{\alpha, \beta\}(\text{OPT}_{c+d}).$$

□

A similar argument shows that an (α, β)-approximation algorithm $\mathbf{XYZ}(G, C)$, for a (**A**, **B**, **S**) problem can be used to find devise a polynomial time $\alpha\beta$ approximation algorithm for the (**A** $\times$ **B**, **S**) problem. A similar argument can also be given for other basic functional combinations. We make two additional remarks.

1. The use of approximation algorithms for (**A**, **B**, **S**)-bicriteria problems, to solve (f(**A**, **B**), **S**) problems (f denotes a function combination of the objectives) does not always yield the best possible solutions. For example problems such as (Total Cost + Total Cost , Spanning Tree) and (Total Cost/Total Cost , Spanning Tree) [Cha77, Meg83] can be solved exactly in polynomial time by direct methods but can only be solved approximately using any algorithm for the (Total Cost, Total Cost , Spanning Tree)-bicriteria problem.[5]

2. Algorithms for solving (f(**A**, **B**), **S**) problems can not in general guarantee any bounded performance ratios for solving the (**A**, **B**, **S**) problem. For example, a solution for the (Total Cost + Total Cost , Spanning Tree) problem or the (Total Cost/Total Cost , Spanning Tree) problem can not be directly used to find a good (α, β)-approximation algorithm for the (Total Cost, Total Cost, Spanning Tree)-bicriteria problem.

The above discussion points out that while a good solution to the (**A**, **B**, **S**)-bicriteria problem yields a "good" solution to any unicriterion version, the converse is not true. It is in this sense that we say our formulation of bicriteria network design problems is general and subsumes other functional combinations.

2.6 Parametric Search

In this section, we present approximation algorithms for a broad class of bicriteria problems where both the objectives in the problem are of the same type (e.g., both are total edge costs of some network computed using two different costs on edges, or both are diameters of some network calculated using two different costs, etc.).

As before, let G be a graph with two (integral) cost functions, c and d. Let C denote the budget on criteria **A**. We assume that the c and d cost functions are of the same kind, i.e., they are both costs on edges or, costs on nodes. Let $\mathbf{UVW}(G, f)$ be any ρ-approximation algorithm that on input G produces a solution subgraph in **S** minimizing criterion **A**, under the single cost function f. In a mild abuse of notation, we also let $\mathbf{UVW}(G, f)$ denote the (f-)cost of the subgraph output by $\mathbf{UVW}(G, f)$ when running on input G under cost function f. We use the following additional notation in the description of the algorithm and the proof of its performance guarantee. Given constants a and b and two cost functions f and g, defined on edges (nodes) of a graph, $af+bg$ denotes the composite function that assigns a cost $af(e) + bg(e)$ to each edge (node) in the graph. Let $h(\hat{D})$ denote the cost of the subgraph, returned by $\mathbf{UVW}(G, (\frac{\hat{D}}{C})c + d)$ (under the $((\frac{\hat{D}}{C})c + d)$-cost function). Let the minimum d-cost of a C-bounded subgraph in **S** be OPT_d. Let HEU_c (HEU_d) denote the c-cost (d-cost) of the subgraph output by ALGORITHM PARAMETRIC-SEARCH given below.

[5] This is true since the (Total Cost, Total Cost, Spanning Tree)-bicriteria problem is **NP**-complete and therefore unless **P** = **NP** can not be solved in polynomial time.

Let $\gamma > 0$ be a fixed accuracy parameter. In what follows, we devise a $((1+\gamma),(1+\frac{1}{\gamma}))$-approximation algorithm for (**A**, **A**, **S**), under the two cost functions c and d. The algorithm consists of performing a binary search with an application of the given approximation algorithm, **UVW**, at each step of this search.

ALGORITHM PARAMETRIC-SEARCH:

- *Input:* G - graph, $\mathcal{C}$ - budget on criteria **A** under the c-cost, **UVW** - a ρ-approximation algorithm that produces a solution subgraph in **S** minimizing criterion **A**, under a single cost function, γ - an accuracy parameter.
- 1. Let $\mathcal{D}_{hi}$ be an upper bound on the d-cost of any $\mathcal{C}$-bounded subgraph in **S**.
  2. Do binary search and find a $\mathcal{D}'$ in $[0, \frac{\mathcal{D}_{hi}}{\gamma}]$ such that
     (a) $\mathbf{UVW}(G, (\frac{\mathcal{D}'}{\mathcal{C}})c + d)$ returns a subgraph such that $\frac{h(\mathcal{D}')}{\mathcal{D}'} > (1+\gamma)\rho$, and
     (b) $\mathbf{UVW}(G, (\frac{\mathcal{D}'+1}{\mathcal{C}})c + d)$ returns a subgraph such that $\frac{h(\mathcal{D}'+1)}{(\mathcal{D}'+1)} \leq (1+\gamma)\rho$.
  3. If the binary search in Step 2 fails to find a valid $\mathcal{C}'$ then output "NO SOLUTION" else output $\mathbf{UVW}(G, (\frac{\mathcal{D}'+1}{\mathcal{C}})c + d)$.
- *Output:* A subgraph from **S** such that its d-cost is at most $(1+\frac{1}{\gamma})\rho$ times that of the minimum d-cost $\mathcal{C}$-bounded subgraph and its c-cost is at most $(1+\gamma)\rho\mathcal{C}$.

Claim 2 *The binary search, in Step 2 of* ALGORITHM PARAMETRIC-SEARCH *is well-defined.*

Proof: Since $(\frac{1}{R}\mathbf{UVW}(G, f))$ is the same as $\mathbf{UVW}(G, \frac{f}{R})$, we get that $\frac{h(\hat{\mathcal{D}})}{\hat{\mathcal{D}}} = \frac{1}{\hat{\mathcal{D}}}\,\mathbf{UVW}(G, (\frac{\hat{\mathcal{D}}}{\mathcal{C}})c+d) = \mathbf{UVW}(G, (\frac{1}{\mathcal{C}})c + \frac{1}{\hat{\mathcal{D}}}d)$, Hence $\frac{h(\hat{\mathcal{D}})}{\hat{\mathcal{D}}}$ is a monotone nonincreasing function of $\hat{D}$. Thus the binary search in Step 2 of ALGORITHM PARAMETRIC-SEARCH is well-defined.

□

Claim 3 *If G contains a $\mathcal{C}$-bounded subgraph in* **S** *then* ALGORITHM PARAMETRIC-SEARCH *outputs a subgraph from* **S** *whose d-cost is at most $(1+\frac{1}{\gamma})\rho$ times that of the minimum d-cost $\mathcal{C}$-bounded subgraph and whose c-cost is at most $(1+\gamma)\rho\mathcal{C}$.*

Proof: By Claim 2 we have that the binary search in Step 2 of ALGORITHM PARAMETRIC-SEARCH is well-defined.

Assume that **S** contains a $\mathcal{C}$-bounded subgraph. Then, since $\mathbf{UVW}(G, (\frac{\mathcal{D}_{hi}}{\gamma\mathcal{C}})c+d)$ returns a subgraph with cost at most $(1+\gamma)\rho\mathcal{D}_{hi}$, under the $((\frac{\mathcal{D}_{hi}}{\gamma\mathcal{C}})c + d)$-cost function, it is clear that ALGORITHM PARAMETRIC-SEARCH outputs a subgraph in this case.

As a consequence of Step 2a and the performance guarantee of the approximation algorithm **UVW**, we get that

$$\mathcal{D}' + 1 \leq \frac{\text{OPT}_d}{\gamma}.$$

By Step 2b we have that the subgraph output by ALGORITHM PARAMETRIC-SEARCH has the following bounds on the c-costs and the d-costs.

$$\text{HEU}_d \leq h(\mathcal{D}'+1) \leq \rho(1+\gamma)(\mathcal{D}'+1) \leq (1+\frac{1}{\gamma})\rho\text{OPT}_d$$

$$\text{HEU}_c \leq (\frac{\mathcal{C}}{\mathcal{D}'+1})h(\mathcal{D}'+1) \leq (\frac{\mathcal{C}}{\mathcal{D}'+1})(1+\gamma)\rho(\mathcal{D}'+1) = (1+\gamma)\rho\mathcal{C}.$$

Thus ALGORITHM PARAMETRIC-SEARCH outputs a subgraph from $\mathbf{S}$ whose d-cost is at most $(1+\frac{1}{\gamma})\rho$ times that of the minimum d-cost $\mathcal{C}$-bounded subgraph and whose c-cost is at most $(1+\gamma)\rho\mathcal{C}$.

□

Note however that the resulting $((1+\gamma)\rho, (1+\frac{1}{\gamma})\rho)$-approximation algorithm for $(\mathbf{A}, \mathbf{A}, \mathbf{S})$ may not be *strongly* polynomial since it depends on the range of the d-costs. But it is a *polynomial-time* algorithm since its running time is linearly dependent on $\log D_{hi}$. Note that $\mathcal{D}_{hi}$ is at most some polynomial in n times the maximum d-cost (of an edge or a node). Hence $\log(\mathcal{D}_{hi})$ is at most a polynomial in terms of the input specification.

The above discussion leads to the following theorem.

Theorem 6 *For all $\gamma > 0$, any ρ-approximation algorithm that produces a solution subgraph in $\mathbf{S}$ minimizing criterion $\mathbf{A}$ can be transformed into a $((1+\gamma)\rho, (1+\frac{1}{\gamma})\rho)$-approximation algorithm for $(\mathbf{A},\mathbf{A},\mathbf{S})$.*

It is obvious that the above theorem can be generalized from the bicriteria case to the multicriteria case (with appropriate worsening of the performance guarantees) where all the objectives are of the same type but with different cost functions.

2.7 Diameter-Constrained Trees

In this section, we describe ALGORITHM DCST, our $(O(\log n), O(\log n))$-approximation algorithm for (Diameter, Total cost, Steiner tree) or the diameter-bounded minimum Steiner tree problem. Note that (Diameter, Total cost, Steiner tree) includes (Diameter, Total cost, Spanning tree) as a special case. We first state the problem formally: given an undirected graph $G = (V, E)$, with two cost functions c and d defined on the set of edges, diameter bound D and terminal set $K \subseteq V$, the (Diameter, Total cost, Steiner tree) problem is to find a tree of minimum c-cost connecting the set of terminals in K with diameter at most D under the d-cost.

The technique underlying ALGORITHM DCST is very general and has wide applicability. Hence, we first give a brief synopsis of it. The basic algorithm works in $(\log n)$ phases. Initially the solution consists of the empty set. During each phase of the algorithm we execute a subroutine Ω to choose a subgraph to add to the solution. The subgraph chosen in each iteration is required to possess two desirable properties. First, it must not increase the budget value of the solution by more than D; second, the solution cost with respect

to $\mathbf{B}$ must be no more than OPT_c, where OPT_c denotes the minimum c-cost of a $\mathcal{D}$ bounded subgraph in $\mathbf{S}$. Since the number of iterations of the algorithm is $O(\log n)$ we get a $(\log n, \log n)$-approximation algorithm. The basic technique is fairly straightforward. The non-trivial part is to devise the right subroutine Ω to be executed in each phase. Ω must be chosen so as to be able to prove the required performance guarantee of the solution. We use the solution based decomposition technique [KR95, Rav94, RMR+93] in the analysis of our algorithm. The basic idea (behind the solution based decomposition technique) is to use the existence of an optimal solution to prove that the subroutine Ω finds the desired subgraph in each phase.

We now present the specifics of ALGORITHM DCST. The algorithm maintains a set of connected subgraphs or *clusters* each with its own distinguished vertex or *center*. Initially each terminal is in a cluster by itself. In each phase, clusters are merged in pairs by adding paths between their centers. Since the number of clusters comes down by a factor of 2 each phase, the algorithm terminates in $\lceil \log_2 |K| \rceil$ phases with one cluster. It outputs a spanning tree of the final cluster as the solution.

ALGORITHM DIAMETER-CONSTRAINED-STEINER-TREE (DCST):

- *Input:* $G = (V, E)$ - graph with two edge cost functions, c and d, D - a bound on the diameter under the d-cost, $K \subseteq V$ - set of terminals, ϵ - an accuracy parameter.

- 1. Initialize the set of clusters $\mathcal{C}_1$ to contain $|K|$ singleton sets, one for each terminal in K. For each cluster in $\mathcal{C}$, define the single node in the cluster to be the center for the cluster. Initialize the phase count $i := 1$.

  2. Repeat until there remains a single cluster in $\mathcal{C}_i$

     (a) Let the set of clusters $\mathcal{C}_i = \{C_1 \ldots, C_{k_i}\}$ at the beginning of the i'th phase (observe that $k_1 = |K|$).

     (b) Construct a complete graph G_i as follows: The node set V_i of G_i is $\{v : v \text{ is the center of a cluster in } \mathcal{C}\}$. Let path P_{xy} be a $(1 + \epsilon)$-approximation to the minimum c-cost diameter D-bounded path between centers v_x and v_y in G. Between every pair of nodes v_x and v_y in V_i, include an edge (v_x, v_y) in G_i of weight equal to the c-cost of P_{xy}.

     (c) Find a minimum-weight matching of largest cardinality in G_i.

     (d) For each edge $e = (v_x, v_y)$ in the matching, merge clusters C_x and C_y, for which v_x and v_y were centers respectively, by adding path P_{xy} to form a new cluster C_{xy}. The node (edge) set of the cluster C_{xy} is defined to be the union of the node (edge) sets of C_x, C_y and the nodes (edges) in P_{xy}. One of v_x and v_y is (arbitrarily) chosen to be the center v_{xy} of cluster C_{xy}. C_{xy} is added to the cluster set $\mathcal{C}_{i+1}$ for the next phase.

     (e) $i := i + 1$.

  3. Let C', with center v' be the single cluster left after Step 2. Output a shortest path tree of C' rooted at v' using the d-cost.

- *Output:* A Steiner tree connecting the set of terminals in K with diameter at most $2\lceil \log_2 n \rceil D$ under the d-cost and of total c-cost at most $(1+\epsilon)\lceil \log_2 n \rceil$ times that of the minimum c-cost diameter D-bounded Steiner tree.

We make a few points about ALGORITHM DCST:

1. The clusters formed in Step 2d need not be disjoint.

2. All steps, except Step 2b, in algorithm DCST can be easily seen to have running times independent of the weights. We employ Hassin's strongly polynomial **FPAS** for Step 2b [Has92]. Hassin's approximation algorithm for the D-bounded minimum c-cost path runs in time $O(|E|(\frac{n^2}{\epsilon} \log \frac{n}{\epsilon}))$. Thus ALGORITHM DCST is a strongly polynomial time algorithm.

3. Instead of finding an exact minimum cost matching in Step 2c, we could find an approximate minimum cost matching [GW92]. This would reduce the running time of the algorithm at the cost of adding a

factor of 2 to the performance guarantee.

We now state some observations that lead to a proof of the performance guarantee of ALGORITHM DCST. Assume, in what follows, that G contains a diameter D-bounded Steiner tree.

Claim 4 *Algorithm DCST terminates in* $\lceil \log_2 |K| \rceil$ *phases.*

Proof: Let k_i denote the number of clusters in phase i. Note that $k_{i+1} = \lceil \frac{k_i}{2} \rceil$ since we pair up the clusters (using a matching in Step 2d). Hence we are left with one cluster after phase $\lceil \log_2 |K| \rceil$ and algorithm DCST terminates.

□

Claim 5 *Let $C \in \mathcal{C}_i$ be any cluster in phase i of algorithm DCST. Let v be the center of C. Then any node u in C is reachable from v by a diameter-iD path in C under the d-cost.*

Proof: Note that the existence of a diameter D-bounded Steiner tree implies that all paths added in Step 2d have diameter at most D under d-cost. The proof now follows in straightforward fashion by induction on i.

□

Lemma 7 *Algorithm DCST outputs a Steiner tree with diameter at most* $2\lceil \log_2 |K| \rceil \cdot D$ *under the d-cost.*

Proof: The proof follows from Claims 4 and 5.

□

This completes the proof of performance guarantee with respect to the d-cost. We now proceed to prove the performance guarantee with respect to the c-costs. We first recall the following pairing lemma.

Claim 6 *[KR95, RMR+93] Let T be an edge-weighted tree with an even number of marked nodes. Then there is a pairing $(v_1, w_1), \ldots, (v_k, w_k)$ of the marked nodes such that the $v_i - w_i$ paths in T are edge-disjoint.*

Proof: A pairing of the marked nodes that minimizes the sum of the lengths of the tree-paths between the nodes paired up can be seen to obey the property in the claim above.

□

Claim 7 *Let* OPT *be any minimum c-cost diameter-D bounded Steiner tree and let* OPT_c *denote its c-cost. The weight of the largest cardinality minimum-weight matching found in Step 2d in each phase i of algorithm DCST is at most* $(1 + \epsilon) \cdot \text{OPT}_c$.

Proof: Consider the phase i of algorithm DCST. Note that since the centers at stage i are a subset of the nodes in the first iteration, the centers v_i are terminal nodes. Thus they belong to OPT. Mark those vertices

in OPT that correspond to the matched vertices, $v_1, v_2, \ldots, v_{2\lfloor \frac{k_i}{2} \rfloor}$, of G_i in Step 2c. Then by Claim 6 there exists a pairing of the marked vertices, say $(v_1, v_2), \ldots, (v_{2\lfloor \frac{k_i}{2} \rfloor - 1}, v_{2\lfloor \frac{k_i}{2} \rfloor})$, and a set of edge-disjoint paths in OPTbetween these pairs. Since these paths are edge-disjoint their total c-cost is at most OPT_c. Further these paths have diameter at most D under the d-cost. Hence the sum of the weights of the edges $(v_1, v_2), \ldots, (v_{2\lfloor \frac{k_i}{2} \rfloor - 1}, v_{2\lfloor \frac{k_i}{2} \rfloor})$ in G_i , which forms a perfect matching on the set of matched vertices, is at most $(1+\epsilon) \cdot \text{OPT}_c$. But in Step 2c of ALGORITHM DCST, a minimum weight perfect matching in the graph G_i was found. Hence the weight of the matching found in Step 2d in phase i of ALGORITHM DCST is at most $(1+\epsilon) \cdot \text{OPT}_c$.

□

Lemma 8 *Let* OPT *be any minimum c-cost diameter-D bounded Steiner tree and let* OPT_c *denote itsc-cost.* ALGORITHM DCST *outputs a Steiner tree with total c-cost at most* $(1+\epsilon)\lceil \log_2 |K| \rceil \cdot \text{OPT}_c$.

Proof: From Claim 7 we have that the c-cost of the set of paths added in Step 2d of any phase is at most $(1+\epsilon) \cdot \text{OPT}_c$. By Claim 4 there are a total of $\lceil \log_2 |K| \rceil$ phases and hence the Steiner tree output by ALGORITHM DCST has total c-cost at most $(1+\epsilon)\lceil \log_2 |K| \rceil \cdot C_D$.

□

From Lemmas 7 and 8 we have the following theorem.

Theorem 9 *There is a strongly polynomial-time algorithm that, given an undirected graph* $G = (V, E)$*, with two cost functions c and d defined on the set of edges, diameter bound D, terminal set* $K \subseteq V$ *and a fixed* $\epsilon > 0$*, constructs a Steiner tree of G of diameter at most* $2\lceil \log_2 |K| \rceil D$ *under the d-costs and of total c-cost at most* $(1+\epsilon)\lceil \log_2 |K| \rceil$ *times that of the minimum-c-cost of any Steiner tree with diameter at most D under d.*

2.8 Treewidth-Bounded Graphs

In this section we consider the class of treewidth-bounded graphs and give algorithms with improved time bounds and performance guarantees for several bicriteria problems mentioned earlier. We do this in two steps. First we develop pseudopolynomial-time algorithms based on dynamic programming. We then present a general method for deriving **FPAS**s from the pseudopolynomial-time algorithms. We also demonstrate an application of the above results to the minimum broadcast time problem.

A class of treewidth-bounded graphs can be specified using a finite number of primitive graphs and a finite collection of binary composition rules. We use this characterization for proving our results. A class of treewidth-bounded graphs Γ is inductively defined as follows [BLW87].

1. The number of primitive graphs in Γ is finite.

2. Each graph in Γ has an ordered set of special nodes called **terminals**. The number of terminals in each graph is bounded by a constant, say k.

3. There is a finite collection of binary composition rules that operate only at terminals, either by identifying two terminals or adding an edge between terminals. A composition rule also determines the terminals of the resulting graph, which must be a subset of the terminals of the two graphs being composed.

2.8.1 Exact Algorithms

Theorem 10 *Every problem in Table 2 can be solved exactly in $O((n \cdot \mathcal{C})^{O(1)})$-time for any class of treewidth bounded graphs with no more than k terminals, for fixed k and a budget $\mathcal{C}$ on the first objective.*

The above theorem states that there exist pseudopolynomial-time algorithms for all the bicriteria problems from Table 2 when restricted to the class of treewidth-bounded graphs. The basic idea is to employ a dynamic programming strategy. In fact, this dynamic programming strategy (in conjunction with Theorem 4) yields polynomial-time (not just pseudopolynomial-time) algorithms whenever one of the criteria is the degree. We illustrate this strategy by presenting in some detail the algorithm for the diameter-bounded minimum cost spanning tree problem.

Theorem 11 *For any class of treewidth-bounded graphs with no more than k terminals, there is an $O(n \cdot k^{2k+4} \cdot \mathcal{C}^{O(k^4)})$-time algorithm for solving the diameter $\mathcal{C}$-bounded minimum c-cost spanning tree problem.*

Proof: Let d be the cost function on the edges for the first objective (diameter) and c, the cost function for the second objective (total cost). Let Γ be any class of decomposable graphs. Let the maximum number of terminals associated with any graph G in Γ be k. Following [BLW87], it is assumed that a given graph G is accompanied by a parse tree specifying how G is constructed using the rules and that the size of the parse tree is linear in the number of nodes.

Let π be a partition of the terminals of G. For every terminal i let d_i be a number in $\{1, 2, \ldots, \mathcal{C}\}$. For every pair of terminals i and j in the same block of the partition π let d_{ij} be a number in $\{1, 2, \ldots, \mathcal{C}\}$. Corresponding to every partition π, set $\{d_i\}$ and set $\{d_{ij}\}$ we associate a cost for G,

$Cost^{\pi}_{\{d_i\},\{d_{ij}\}}$ = Minimum total cost under the c function of any forest containing a tree for each block of π, such that the terminal nodes occurring in each tree are exactly the members of the corresponding block of π, no pair of trees is connected, every vertex in G appears in exactly one tree, d_i is an upper bound on the maximum distance (under the d-function) from i to any vertex in the same tree and d_{ij} is an upper bound the distance (under the d-function) between terminals i and j in their tree.

For the above defined cost, if there is no forest satisfying the required conditions the value of $Cost$ is defined to be ∞.

Note that the number of cost values associated with any graph in Γ is $O(k^k \cdot C^{O(k^2)})$. We now show how the cost values can be computed in a bottom-up manner given the parse tree for G. To begin with, since Γ is fixed, the number of primitive graphs is finite. For a primitive graph, each cost value can be computed in constant time, since the number of forests to be examined is fixed. Now consider computing the cost values for a graph G constructed from subgraphs G_1 and G_2, where the cost values for G_1 and G_2 have already been computed. Notice that any forest realizing a particular cost value for G decomposes into two forests, one for G_1 and one for G_2 with some cost values. Since we have maintained the best cost values for all possibilities for G_1 and G_2, we can reconstruct for each partition of the terminals of G the forest that has minimum cost value among all the forests for this partition obeying the diameter constraints. We can do this in time independent of the sizes of G_1 and G_2 because they interact only at the terminals to form G, and we have maintained all relevant information.

Hence we can generate all possible cost values for G by considering combinations of all relevant pairs of cost values for G_1 and G_2. This takes time $O(k^4)$ per combination for a total time of $O(k^{2k+4} \cdot C^{O(k^4)})$. As in [BLW87], we assume that the size of the given parse tree for G is $O(n)$. Thus the dynamic programming algorithm takes time $O(n \cdot k^{2k+4} \cdot C^{O(k^4)})$. This completes the proof.

□

2.8.2 Fully Polynomial-Time Approximation Schemes

The pseudopolynomial-time algorithms described in the previous section can be used to design **FPAS**s for these same problems for the class of treewidth-bounded graphs. We illustrate our ideas once again by devising an **FPAS** for the (Diameter, Total cost, Spanning tree)-bicriteria problem for the class of treewidth-bounded graphs. The basic technique underlying our algorithm, ALGORITHM FPAS-DCST, is approximate binary search using rounding and scaling - a method similar to that used by Hassin [Has92] and Warburton [War87].

As in the previous section, let G be a treewidth-bounded graph with two (integral) edge-cost functions c and d. Let D be a bound on the diameter under the d-cost. Let ϵ be an accuracy parameter. Without loss of generality we assume that $\frac{1}{\epsilon}$ is an integer. We also assume that there exists a D-bounded spanning tree in G. Let OPT be any minimum c-cost diameter D-bounded spanning tree and let OPT_c denote its c-cost. Let $\mathbf{TCSTonTW}(G, c, d, C)$ be a pseudopolynomial time algorithm for the (Total cost, Diameter, Spanning tree) problem on treewidth-bounded graphs, i.e., **TCSTonTW** outputs a minimum diameter spanning tree of G with total cost at most C (under the c-costs). Let the running time of **TCSTonTW** be $p(n, C)$ for some polynomial p. For carrying out our approximate binary search we need a testing procedure PROCEDURE TEST(V) which we detail below:

PROCEDURE TEST(V):

- *Input:* G - treewidth bounded graph, D - bound on the diameter under the d-cost, V - testing parameter, **TCSTonTW** - a pseudopolynomial time algorithm for the (Total cost, Diameter, Spanning tree) problem on treewidth-bounded graphs, ϵ - an accuracy parameter.
- 1. Let $\lfloor \frac{c}{V\epsilon/(n-1)} \rfloor$ denote the cost function obtained by setting the cost of edge e to $\lfloor \frac{c_e}{V\epsilon/(n-1)} \rfloor$.
  2. If there exists a C in $[0, \frac{n-1}{\epsilon}]$ such that **TCSTonTW**$(G, \lfloor \frac{c}{V\epsilon/(n-1)} \rfloor, d, C)$ produces a spanning tree with diameter at most D under the d-cost then output LOW otherwise output HIGH.
- *Output:* HIGH/LOW.

We now prove that PROCEDURE TEST(V) has the properties we need for helping us do a binary search.

Claim 8 *If* $\text{OPT}_c \leq V$ *then* PROCEDURE TEST(V) *outputs LOW. And, if* $\text{OPT}_c > V(1+\epsilon)$ *then* PROCEDURE TEST(V) *outputs HIGH.*

Proof: If $\text{OPT}_c \leq V$ then since

$$\sum_{e \in \text{OPT}} \lfloor \frac{c_e}{V\epsilon/(n-1)} \rfloor \leq \sum_{e \in \text{OPT}} \frac{c_e}{V\epsilon/(n-1)} \leq \frac{\text{OPT}_c}{V\epsilon/(n-1)} \leq \frac{n-1}{\epsilon}$$

therefore PROCEDURE TEST(V) outputs LOW.

Let T_c be the c-cost of any diameter D bounded spanning tree. Then we have $T_c \geq \text{OPT}_c$. If $\text{OPT}_c > V(1+\epsilon)$ then since

$$\sum_{e \in T} \lfloor \frac{c_e}{V\epsilon/(n-1)} \rfloor \geq \sum_{e \in T} (\frac{c_e}{V\epsilon/(n-1)} - 1) \geq \frac{T_c}{V\epsilon/(n-1)} - (n-1) \geq \frac{\text{OPT}_c}{V\epsilon/(n-1)} - (n-1) > \frac{n-1}{\epsilon}$$

therefore PROCEDURE TEST(V) outputs HIGH.

□

Claim 9 *The running time of* PROCEDURE TEST(V) *is* $O(\frac{n}{\epsilon} p(n, \frac{n}{\epsilon}))$.

Proof: PROCEDURE TEST(V) invokes **TCSTonTW** only $\frac{n-1}{\epsilon}$ times. And each time the budget C is bounded by $\frac{n-1}{\epsilon}$, hence the running time of PROCEDURE TEST(V) is $O(\frac{n}{\epsilon} p(n, \frac{n}{\epsilon}))$.

□

We are ready to describe ALGORITHM FPAS-DCST – which uses PROCEDURE TEST(V) to do an approximate binary search.

ALGORITHM FPAS-DCST:

- *Input:* G - treewidth-bounded graph, D - bound on the diameter under the d-cost, **TCSTonTW** - a pseudopolynomial time algorithm for the (Total cost, Diameter, Spanning tree) problem on treewidth-bounded graphs, ϵ - an accuracy parameter.

- 1. Let C_{hi} be an upper bound on the c-cost of any D-bounded spanning tree. Let $LB = 0$ and $UB = C_{hi}$.
  2. While $UB \geq 2LB$ do
     (a) Let $V = (LB + UB)/2$.
     (b) If PROCEDURE TEST(V) returns HIGH then set $LB = V$ else set $UB = V(1+\epsilon)$.
  3. Run $\mathbf{TCSTonTW}(G, \lfloor \frac{c}{LB\epsilon/(n-1)} \rfloor, d, C)$ for all C in $[0, 2(\frac{n-1}{\epsilon})]$ and among all the trees with diameter at most D under the d-cost output the tree with the lowest c-cost.

- *Output:* A spanning tree with diameter at most D under the d-cost and with c-cost at most $(1+\epsilon)$ times that of the minimum c-cost D-bounded spanning tree.

Lemma 12 *If G contains a D-bounded spanning tree then* ALGORITHM FPAS-DCST *outputs a spanning tree with diameter at most D under the d-cost and with c-cost at most* $(1+\epsilon)\text{OPT}_c$.

Proof: It follows easily from Claim 8 that the loop in Step 2 of ALGORITHM FPAS-DCST executes $O(\log C_{hi})$ times before exiting with $LB \leq \text{OPT}_c \leq UB < 2LB$.

Since

$$\sum_{e \in \text{OPT}} \lfloor \frac{c_e}{LB\epsilon/(n-1)} \rfloor \leq \sum_{e \in \text{OPT}} \frac{c_e}{LB\epsilon/(n-1)} \leq \frac{\text{OPT}_c}{LB\epsilon/(n-1)} \leq 2(\frac{n-1}{\epsilon})$$

we get that Step 3 of ALGORITHM FPAS-DCST definitely outputs a spanning tree. Let HEU be the tree output. Then we have that

$$\text{HEU}_c = \sum_{e \in \text{HEU}_c} c_e \leq LB\epsilon/(n-1) \sum_{e \in \text{HEU}_c} \frac{c_e}{LB\epsilon/(n-1)} \leq LB\epsilon/(n-1)(\sum_{e \in \text{HEU}_c} \lfloor \frac{c_e}{LB\epsilon/(n-1)} \rfloor + 1).$$

But since Step 3 of ALGORITHM FPAS-DCST outputs the spanning tree with minimum c-cost we have that

$$\sum_{e \in \text{HEU}_c} \lfloor \frac{c_e}{LB\epsilon/(n-1)} \rfloor \leq \sum_{e \in \text{OPT}} \lfloor \frac{c_e}{LB\epsilon/(n-1)} \rfloor.$$

Therefore

$$\text{HEU}_c \leq LB\epsilon/(n-1) \sum_{e \in \text{OPT}} \lfloor \frac{c_e}{LB\epsilon/(n-1)} \rfloor + \epsilon LB \leq \sum_{e \in \text{OPT}} c_e + \epsilon \text{OPT}_c \leq (1+\epsilon)\text{OPT}_c.$$

This proves the claim.

□

Lemma 13 *The running time of* ALGORITHM FPAS-DCST *is* $O(\frac{n}{\epsilon}p(n, \frac{n}{\epsilon})\log C_{hi})$.

Proof: From Claim 9 we see that Step 2 of ALGORITHM FPAS-DCST takes time $O(\frac{n}{\epsilon}p(n, \frac{n}{\epsilon})\log C_{hi})$ while Step 3 takes time $O(\frac{2n}{\epsilon}p(n, \frac{2n}{\epsilon}))$. Hence the running time of ALGORITHM FPAS-DCST is $O(\frac{n}{\epsilon}p(n, \frac{n}{\epsilon})\log C_{hi})$.
□

Lemmas 12 and 13 yield:

Theorem 14 *For the class of treewidth-bounded graphs, there is an* **FPAS** *for the (Diameter, Total cost, Spanning tree)-bicriteria problem with performance guarantee* $(1, 1+\epsilon)$.

As mentioned before, similar theorems hold for the other problems in Table 2 and all these results extend directly to Steiner trees.

2.8.3 Near-Optimal Broadcast Schemes

The polynomial-time algorithm for the (Degree, Diameter, Spanning tree))-bicriteria problem for treewidth-bounded graphs can be used in conjunction with the ideas presented in [Rav94] to obtain near-optimal broadcast schemes for the class of treewidth-bounded graphs. As mentioned earlier, these results generalize and improve the results of Kortsarz and Peleg [KP95].

Given an unweighted graph G and a root r, a *broadcast scheme* is a method for communicating a message from r to all the nodes of G. We consider a telephone model in which the messages are transmitted synchronously and at each time step, any node can either transmit or receive a message from at most one of its neighbors. The *minimum broadcast time problem* is to compute a scheme that completes in the minimum number of time steps. Let $\text{OPT}_r(G)$ denote the minimum broadcast time from root r and let $\text{OPT}(G) = \max_{r \in G} \text{OPT}_r(G)$ denote the minimum broadcast time for the graph from any root. The problem of computing $\text{OPT}_r(G)$ – the *minimum rooted broadcast time problem* – and that of computing $\text{OPT}(G)$ – the *minimum broadcast time problem* are both **NP**-complete for general graphs [GJ79a]. It is easy to see that any approximation algorithm for the minimum rooted broadcast time problem automatically yields an approximation algorithm for the minimum broadcast time problem with the same performance guarantee.

Let $\mathcal{T}(G)$ denote the set of all spanning trees of graph G. For spanning tree $T \in \mathcal{T}(G)$ let ΔT denote the maximum, over all nodes, of the degree of the node in T. Let the r-eccentricity of T, $\mathcal{R}_r(T)$, denote the maximum, over all nodes, of the distance of the node from r in T. We generalize the definition of the *poise* of a graph from [Rav94] to that of the *rooted poise* of a graph:

$$P_r(G) = \min_{T \in \mathcal{T}(G)} (\Delta(T) + \mathcal{R}_r(T))$$

Theorem 15 *[Rav94]*

$$\Omega(P_r(G)) \leq \text{OPT}_r(G) \leq O(P_r(G) \cdot \frac{\log n}{\log \log n})$$

The proof of this theorem is constructive and implies that a good solution to the rooted poise of a graph can be used to construct a good solution for the minimum rooted broadcast time problem with an $O(\frac{\log n}{\log \log n})$-factor overhead.

Lemma 16 *Given a treewidth bounded graph G and root r, in polynomial time a spanning tree T can be found such that $(\Delta(T) + \mathcal{R}_r(T)) = P_r(G)$.*

Proof: Since G is an unweighted treewidth-bounded graph we can obtain a (strongly) polynomial-time algorithm to solve the (Degree, r-eccentricity, Spanning tree)-bicriteria problem on G, using the ideas outlined in Section 2.8.1. Run this algorithm with all possible degree bounds and choose the tree T with the least value of $(\Delta(T) + \mathcal{R}_r(T))$.

□

From Theorem 15 and Lemma 16 we get the following theorem

Theorem 17 *For any class of treewidth-bounded graphs there is a polynomial-time $O(\frac{\log n}{\log \log n})$-approximation algorithm for the minimum rooted broadcast time problem and the minimum broadcast time problem.*

2.9 Concluding Remarks

We have obtained the first polynomial-time approximation algorithms for a large class of bicriteria network design problems. The objective function we considered were (i) degree, (ii) diameter, and (iii) total cost. The connectivity requirements we considered were spanning trees, Steiner trees, and (in several cases) generalized Steiner trees. Our results were based mostly on the following three main ideas:

1. A binary search method to convert an (α, β)-approximation algorithm for (**A**, **B**, **S**)-bicriteria problems to a (β, α)-approximation algorithm for (**B**, **A**, **S**)-bicriteria problems.
2. A parametric search technique to devise approximation algorithms for (**A**,**A**,**S**)-bicriteria problems.
3. A cluster based approach for devising approximation algorithms for certain categories of (**A**,**B**,**S**)-bicriteria problems.

We also devised pseudopolynomial time algorithms and fully polynomial time approximation schemes for a number of bicriteria network design problems on the class of treewidth-bounded graphs.

The following are some natural open questions and directions suggested by our work:

1. Devise algorithms that improve upon the performance guarantees presented in this chapter. As a step in this direction, recently Ravi and Goemans [RG96] have devised a $(1, 1 + \epsilon)$ approximation scheme for the (Total Cost, Total Cost, Spanning tree) problem.

2. Devise approximation algorithms for the (Diameter, Total cost, Generalized Steiner tree) problem. For more information on the generalized Steiner tree problem look at [AKR95] and the references therein.

3. Find good broadcast schemes which minimize the total cost of the spanning tree. In our language this problem translates to finding an approximation algorithm for the (Degree, Diameter, Total Cost, Spanning tree) problem.

Chapter 3

Service-Constrained Network Design Problems

3.1 Motivation

The optical network is a pure data transmission medium. All the computing and processing continues to be done in the electronic world. An important issue in interfacing these two worlds – the electronic and the optic – is that of designing the optical network subject to location-theoretic constraints imposed by the electronic world. Given a set of sites in a network we wish to select a subset of the sites at which to place optoelectronic switches and routers [KRS96b]. As before, the major requirement is that every site should be within a pre-specified distance from an optoelectronic access node and the chosen sites should be connected together using fiber-optic links as a minimum cost tree (See Fig. 3-1. Circles represent prespecified distances).

A problem of a similar nature comes up in the area of distributed database management. The problem [WM91] can be stated as follows: given a set of sites in a network we wish to select a subset of the sites at which to place copies of the database. The major requirement is that each site should be able to access a copy of the database within a prespecified service time, and the chosen sites should be connected together as a minimum cost tree so that updates to one of the copies can be propagated to the other copies in an inexpensive manner (See Fig. 3-2. Circles represent prespecified service times).

These problems can be thought of as particular instances of a more general class of "service-constrained network design problems." Informally, service-constrained network design problems involve both a location-theoretic objective and a cost-minimization objective subject to connectivity constraints. The location-theoretic objective requires that we choose a subset of nodes at which to "locate" services such that each node is within a bounded distance from at least one chosen location. The cost-minimization objective requires that the chosen locations be connected by a network minimizing parameters such as diameter or total cost. The two objectives are measured under two (possibly) different cost functions.

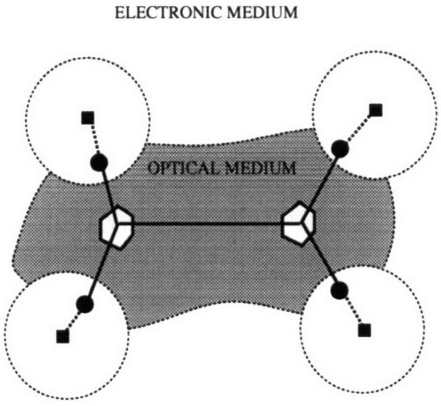

Figure 3-1: Optoelectronic switches linked by a network.

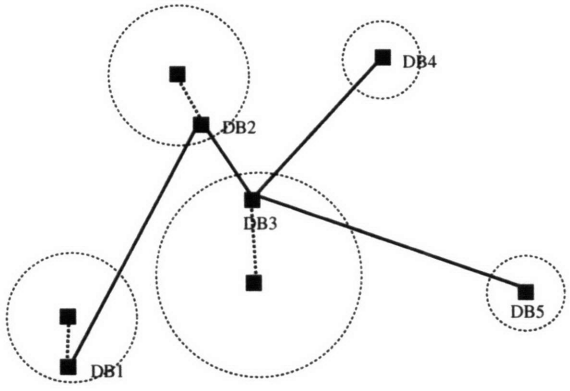

Figure 3-2: Database copies linked by a network.

3.2 Summary of Results

The prototypical problem we consider in this chapter is the following: given an undirected graph $G = (V, E)$ with two different cost functions c_e (modeling the service cost) and d_e (modeling the construction cost) for each edge $e \in E$, and a bound $\mathcal{S}_v$ (on the service constraint for each vertex v), find a minimum d-cost tree such that every node v in the graph is *serviced* by some node in the tree, i.e. every node v is within distance $\mathcal{S}_v$ (under the c-costs) of some node in the tree.

In this chapter, we study the complexity and approximability of a number of service-constrained network design problems using the bicriteria framework developed in Chapter 2. The two versions of the location-theoretic or service cost objective that we consider are: (i) Non-uniform service cost and (ii) Uniform service cost. In the *Non-uniform service cost* version a service constraint $\mathcal{S}_{v_k}$ is specified for each vertex. The *Uniform service cost* version is a special case where $\mathcal{S}_{v_k} = \mathcal{S}, \forall v_k$, i.e., all vertices have the same service constraint.

For the cost-minimization objective we focus our attention on the total cost of the network. The *Total cost* objective is the sum of the costs of all the edges in the tree. We also consider the *Diameter* objective – the maximum distance between any pair of nodes in the tree – and the *Bottleneck* objective – the maximum value of any edge in the tree.

As mentioned before, the two objectives are measured with respect to different edge-cost functions. The (budgeted) service cost objective is measured using the c-cost function while the cost-minimization objective is measured using the d-cost function. As stated before, a node u is said to *service* node v if u is within distance $\mathcal{S}_v$ of v, under the c-cost. The *service-degree* of a node is defined to be the number of nodes it services. All our results come in two flavors: (i) Different cost functions, and (ii) Identical cost functions. The *Identical cost functions* version is a special case of the *Different cost functions* case where the two cost functions are the same, i.e., $c_e = d_e, \forall e$.

We now state our main results.

Definition 1 *A node u is said to* **service** *a node v if u is within distance $\mathcal{S}_v$ of v. The service-degree of a node is the number of nodes it services. The service-degree of the graph is the maximum over all nodes of the service-degree of the node and is denoted by $\tilde{\Delta}$.*

Theorem 18 *There is a $(1, O(\tilde{\Delta} \cdot \ln n))$-approximation algorithm for the (Non-uniform service cost, Total cost, Tree)-bicriteria problem with different cost functions.*

We provide a counterbalancing hardness of approximation result by showing that:

Theorem 19 *Unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\log \log n})$, the (Uniform service cost, Total cost, Tree)-bicriteria problem, with different cost functions, cannot be approximated to within (α, β), for any $\alpha \geq 1$ and any $\beta < \ln n$.*

We consider the identical cost functions case and show that

Theorem 20 *For any $\epsilon > 0$ there is a $(2(1+\epsilon), 2(1+\frac{1}{\epsilon}))$-approximation algorithm for the (Non-uniform service cost, Total cost, Tree)-bicriteria problem with identical cost functions.*

This is opposed by

Theorem 21 *Unless* $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\log \log n})$*, the (Uniform service cost, Total cost, Tree)-bicriteria problem, with identical cost functions, cannot be approximated to within* (α, β)*, for* $\alpha = 1$ *and any* $\beta < \ln n$*.*

We also show that

Theorem 22 *(Non-uniform service cost, Diameter, Tree) and (Non-uniform service cost, Bottleneck, Tree)-bicriteria problems, with different cost functions, are solvable exactly in polynomial-time.*

We consider the (Uniform service cost, Total cost, Generalized Steiner forest)-bicriteria problem – a generalization that arises from its application to the optical network design problems. If we had a number of sites with connectivity requirements only among some subsets of the sites, rather than all the sites, then we need the solution subgraph to be a forest and not necessarily a tree. This motivates the service-constrained generalized Steiner forest problem. The formal statement of the (Uniform service cost, Total cost, Generalized Steiner forest)-bicriteria problem is as follows: given an undirected graph $G = (V, E)$ with two different cost functions c_e (modeling the service cost) and d_e (modeling the construction cost) for each edge $e \in E$, a set of k site pairs (s_i, t_i), and a bound $\mathcal{S}$ (on the maximum service constraint), find a minimum d-cost forest $\mathcal{F}$ such that for each site pair (s_i, t_i) there exists a tree $T \in \mathcal{F}$ with the property that both s_i and t_i are within distance $\mathcal{S}$ of the tree.

Our main result for the service-constrained generalized Steiner forest problem is the following:

Theorem 23 *There is a* $(2(1+\epsilon), 6(1+\frac{1}{\epsilon}))$*-approximation algorithm for the (Uniform service cost, Total cost, Generalized Steiner forest)-bicriteria problem with identical cost functions.*

3.3 Previous Work

Variants of the service-constrained *tour* problem have been considered in [AFMP94, AH91, CS89]. Current and Schilling [CS89] consider the *covering salesperson problem* and present a heuristic for it without providing any performance guarantees. In this problem, nodes represent customers and the service radius represents the distance a customer is willing to travel to meet the salesperson. The goal is to find a minimum length tour connecting a subset of the nodes so that the nodes are strictly serviced. Arkin and Hassin [AH91] considered geometric versions of the problem, where the service neighborhood (i.e., the neighborhood the customer is willing to travel) is modeled as a region in the plane. For convex neighborhoods, they present heuristics that provide constant performance guarantees. They also show how their heuristics can be extended to nonconvex regions. Arkin et. al. [AFMP94] considered additional geometric variations of the covering tour problem including the *lawn mower problem*, where the goal is to find a tour such that each given point is within a circle of unit radius from at least one point on the tour. They provide an approximation algorithm for this problem with a constant performance guarantee.

References [AFMP94, AH91] considered only the geometric versions of the problems. In the identical cost functions case, where the edge costs satisfy the triangle inequality, our approximation algorithm for the (Non-uniform service cost, Total cost, Tree)-bicriteria problem can be used to produce a tour such that for any $\epsilon > 0$, the service constraint for each node is violated by a factor of at most $2(1+\epsilon)$ and the cost of the tour is at most $2(1+\frac{1}{\epsilon})$ times that of a minimum cost tour which strictly services the nodes.

The paper by Marathe et al ([MRS+95]) and the references therein are an extensive source on the area of bicriteria and multicriteria network design. The framework that we employ was developed in [MRS+95].

The organization of the rest of the chapter is as follows: Section 3.4 contains the proofs of Theorems 18 and 19, where the cost functions are different; Section 3.5 contains the proofs of Theorems 20 and 21, where the cost functions are identical; Section 3.6 contains the proof of Theorem 22; Section 3.7 contains the proof of Theorem 23; Section 3.8 contains some concluding remarks and open problems.

3.4 Different Cost Functions

Before we give the proof of Theorem 18, we need to state two facts. The first fact is that we may assume that the graph is complete and that the edge cost functions, c and d, obey triangle inequality. The reason for this is as follows: consider the (complete) graph obtained on the same set of vertices by adding edges between every pair of vertices of c and d-costs equal to that of the shortest c and d-cost paths between the corresponding vertices in the original graph; then any solution on this new graph transforms to a solution of identical value in the original graph.

We need some more preliminaries before we can state the second fact. Given a graph G with edge weights and node weights, we define the **ratio weight** of a simple cycle C in G to be

$$\frac{\sum_{e \in C} wt_e}{\sum_{v \in C} wt_v}.$$

Here wt_e denotes the weight of an edge and wt_v denotes the weight of a vertex. In other words, the ratio weight of a cycle is the ratio of the edge weight of the cycle to the node weight of the cycle.

The following problem is **NP**-hard:

Definition 2 *MIN-RATIO-ROOTED-CYCLE (**MRRC**) Problem : Given a graph $G = (V, E)$ with edge and node weights, and a distinguished vertex $r \in V$ called the root, find a simple cycle in G that contains r and has minimum ratio weight.*

Now we state our second fact which is obtained by a slight modification of the ideas in [BRV96].

Theorem 24 *There is a polynomial-time approximation algorithm with constant factor, ρ, performance guarantee for the* **MRRC** *problem.*

Having stated the two facts we are now in a position to present the main idea behind our algorithm. To begin with, we may assume that a specific node r belongs to the optimal tree. By running over all possible r's and picking the best we find the required (approximate) tree. The algorithm runs in phases. Initially, the solution contains only the node r. At any stage only a subset of the nodes are serviced by the set of solution nodes. Each phase the algorithm finds a nearly optimal minimum ratio weight cycle that services some of the remaining unserviced nodes in the graph. The cycle is contracted to a single node and the algorithm moves to the next phase. Termination occurs when all the nodes in the graph are serviced. A logarithmic performance guarantee is obtained by assuring that the cycle added in each phase has low cost.

ALGORITHM DIFFERENT:

- *Input:* A graph $G = (V, E)$, edge cost functions c and d, service budget $\mathcal{S}_{v_k}$ for vertex v_k under the c-cost function.
- 1. At any point in the algorithm, for each vertex $v_k \in V$, let B_{v_k} denote the set of vertices that are within c-distance of at most $\mathcal{S}_{v_k}$ from v_k.
  2. For $i = 1$ to n do
     (a) Set $r = v_i$. Set $j = 0$. Set $G_0 = G - B_r + r$.
     (b) While $G_j <> r$.
        i. Set $j = j + 1$.
        ii. Compute C_j, a ρ-approximate solution to the **MRRC** problem on G_j where the edge weights are the d-edge-costs and the node weights are $|B_{v_k}|$ for $k \neq i$ and 0 for r.
        iii. Modify G_j by contracting C_j into a supernode. Set r to be the new supernode.
        iv. Set $G_j = G_j - B_r + r$.
     (c) Let $\mathcal{T}_i$ be a minimum spanning tree on $\bigcup_j C_j$ under the d-cost.
  3. Let HEU $= \min_i \mathcal{T}_i$. Output HEU.
- *Output:* A tree HEU such that every vertex $v_i \in V$ is within a distance $\mathcal{S}_{v_i}$ from some node in HEU, under the c-cost, and the d-cost of HEU is at most $O(\tilde{\Delta} \cdot \ln n)$ times that of an optimal service-constrained tree.

It is easy to see that ALGORITHM DIFFERENT outputs a tree that services all the nodes. It remains to show that the d-cost of HEU is within a factor of $O(\tilde{\Delta} \cdot \ln n)$ of the optimal. We prove this in the following two lemmas.

Let OPT denote an optimal tree. In what follows, let i denote that iteration of Step 2 in which $r = v_i \in$ OPT. Let f denote the number of iterations of Step 2b for this particular value of i. Let the set of cycles chosen in Step 2(b)ii of the algorithm be $C_1, \ldots, C_f$, in order. We use C_j to denote both the cycle as well as the d-

cost of the cycle. We also use OPT and HEU to denote the d-costs of the corresponding tree. Let ϕ_j denote the number of nodes in G that are not serviced by r after choosing the cycle C_j in the jth iteration of Step 2b. Alternatively, ϕ_j is the number of vertices in $G - r$ after Step 2(b)iv in the j'th iteration.. Thus, $\phi_0 \leq n$ while $\phi_f = 0$. Let cycle C_j service t_j new nodes.

Lemma 25

$$\frac{C_j}{t_j} \leq \frac{2\rho\tilde{\Delta}\text{OPT}}{\phi_{j-1}}.$$

Proof: Focus on the graph G_j at the end of iteration $j-1$. Since OPT services all the vertices we have that $\sum_{v_k \in \text{OPT}} |B_{v_k}| = |V(G_j)| = \phi_{j-1}$.

We first observe that OPT induces a cycle (by doing an Euler walk along the outside of the OPT tree, see [CLR90a], pp 697–700) with a minimum ratio weight of $\frac{2\text{OPT}}{\phi_{j-1}}$. Hence, since in Step 2(b)ii we choose a ρ-approximate minimum ratio cycle C_j it follows that

$$\frac{C_j}{\sum_{v_k \in C_j} |B_{v_k}|} \leq \frac{2\rho\text{OPT}}{\phi_{j-1}}.$$

Since the service-degree of each vertex in G is at most $\tilde{\Delta}$, it follows that no vertex contributes more than $\tilde{\Delta}$ to the denominator of the left hand side in the above equation. Thus $\tilde{\Delta} \cdot t_j \geq \sum_{v_k \in C_j} |B_{v_k}|$. Hence

$$\frac{C_j}{\tilde{\Delta} \cdot t_j} \leq \frac{2\rho\text{OPT}}{\phi_{j-1}}.$$

The lemma follows. □

Lemma 26

$$\text{HEU} \leq 2\rho\tilde{\Delta}\, H_n \text{OPT}$$

where $H_n = 1 + \frac{1}{2} + \ldots + \frac{1}{n}$ *is the harmonic function.*

Proof: By definition of ϕ_j and t_j, we have that

$$\phi_j = \phi_{j-1} - t_j \tag{3.1}$$

and from Lemma 25, we have

$$t_j \geq \frac{C_j \phi_{j-1}}{2\rho\tilde{\Delta}\text{OPT}} \tag{3.2}$$

Substituting Equation (3.2) into (3.1) we get

$$C_j \leq (2\rho\tilde{\Delta}\text{OPT})\frac{t_j}{\phi_{j-1}} \leq (2\rho\tilde{\Delta}\text{OPT})(H_{\phi_{j-1}} - H_{\phi_j}).$$

Hence, since $\phi_0 \leq n$ and $\phi_f = 0$, we get

$$\sum_{j=1}^{f} C_j \leq (2\rho\tilde{\Delta}\text{OPT})(H_{\phi_0} - H_{\phi_f}) \leq (2\rho\tilde{\Delta}\text{OPT})H_n.$$

This completes the proof of the lemma since $\text{HEU} \leq \sum_{j=1}^{f} C_j$. □

Since $H_n \approx \ln n$ we obtain Theorem 18.

Remark. Note that the bounds of Theorem 18 also extend to the Steiner version where only a set of terminal sites need to be serviced. The Steiner version reduces to the regular version by setting the service budgets of the nonterminal nodes to some large value, such as the diameter of the graph.

3.4.1 Hardness

In this section we prove Theorem 19. We show that for any $\alpha \geq 1$, if there is a polynomial-time (α, β)-approximation algorithm for the (Uniform service cost, Total cost, Tree)-bicriteria problem, then there is a polynomial-time β-approximation algorithm for the **MIN SET COVER** problem.

As an instance of the **MIN SET COVER** problem we are given a universe $Q = \{q_1, q_2, \ldots, q_n\}$ and a collection $Q_1, Q_2, \ldots, Q_m$ of subsets of Q. The problem is to find a minimum cost collection of the subsets whose union is Q. Recently Feige [Fei96] has shown the following non-approximability result:

Theorem 27 *Unless* $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\log\log n})$, *the* **MIN SET COVER** *problem, with a universe of size* k, *cannot be approximated to better than a* $\ln k$ *factor.*

We now present the details of the approximation preserving reduction from **MIN SET COVER** to the (Uniform service cost, Total cost, Tree)-bicriteria problem. Corresponding to an instance of **MIN SET COVER** we construct the natural bipartite graph, one partition for set nodes and the other for element nodes, with edges representing element inclusion in the sets. To this bipartite graph, we add an enforcer node with edges to all the set nodes and also a mate node attached to the enforcer. Now we complete this *skeleton-graph* by throwing in all the edges. We set the d-cost of an edge from the enforcer to a set node to be 1. We set the d-cost of all other edges to be $\beta \cdot m + 1$. We now specify the c-costs (service costs) for the edges. We set the c-cost for the edge between the enforcer and the mate and for each edge between the enforcer and a set node to be S. We set the c-cost of an edge between a set node and the element nodes contained in this set to also be S. The d-cost of all other edges is set to $\alpha \cdot S + 1$. Let G denote the resulting instance (See Fig. 3-3) of the

(Uniform service cost, Total cost, Tree)-bicriteria problem with the c and d cost functions as specified above and a uniform service budget of S.

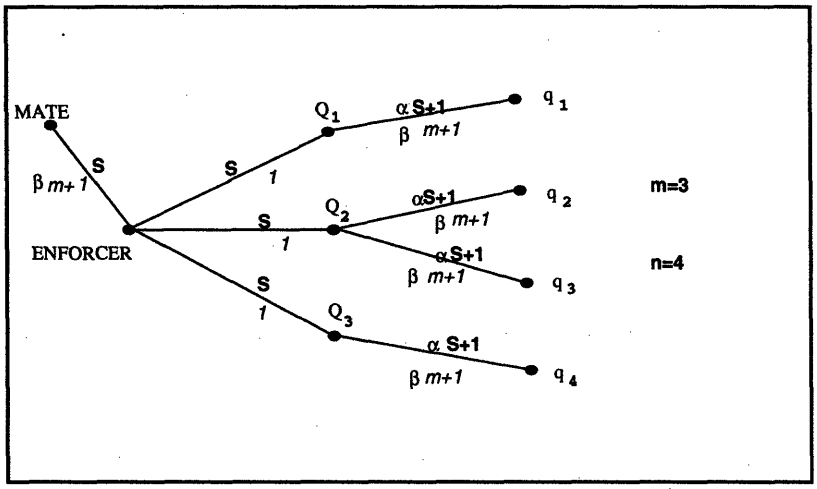

Figure 3-3: Skeleton-graph with the c-costs indicated above the edges and the d-costs indicated below.

It is easy to see that any collection of k subsets which form a set cover correspond to a tree in G that strictly services all the nodes and which has a d-cost of k. This is because the tree consisting of the enforcer and the nodes corresponding to the sets in the collection, strictly services all the nodes and has a d-cost of k.

Let OPT denote the size of a minimum set cover to the original instance. Now we show that if there exists a tree T which is an (α, β)-approximation to the resulting instance G of the (Uniform service cost, Total cost, Tree)-bicriteria problem, then from it we can derive a β-approximation to the original set cover instance. Such a tree T must satisfy the following properties:

1. The c-cost of T is at most $\beta \cdot$ OPT. This follows from the definition of β-approximation and the fact that there exists a tree in G corresponding to OPT with d-cost at most OPT.

2. The nodes of G must be serviced by T within budget S. This is because the c-cost of any edge is either S or $\alpha S + 1$, but T violates the budget constraint by at most a factor α.

3. The mate node cannot be in T. This is because the d-cost of any edge from the mate node is $\beta \cdot m + 1$ which is greater than the d-cost of T. Since only the enforcer node can service the mate node with a service cost of at most αS, the enforcer must be in T.

4. Using the same reasoning as that for the mate node, none of the nodes representing the ground elements can be in T. To service these nodes, some of the set nodes must be in T.

We thus conclude that T consists only of the enforcer node and some of the set nodes. Since the d-cost of T is at most $\beta \cdot$ OPT, it follows that the number of set nodes in T is at most $\beta \cdot$ OPT. Since the element nodes are serviced by the chosen set nodes with a service distance of at most αS, the corresponding sets must form a set cover. We thus have a β-approximation algorithm for set cover and this completes the proof.

3.5 Identical Cost Functions

3.5.1 Approximation Algorithm

In this section, we give a proof of Theorem 20. We provide the details of our approximation algorithm for finding a service-constrained tree with non-uniform service costs and identical cost functions.

ALGORITHM IDENTICAL:

- *Input:* An undirected graph $G = (V, E)$, edge cost function c, service radius $\mathcal{S}_{v_k}$ for vertex v_k, an accuracy parameter $\epsilon > 0$.
- 1. For each node $v_k \in V$, let B_{v_k} denote the set of vertices that are within distance of at most $(1+\epsilon)\mathcal{S}_{v_k}$ from v_k.
  2. Set $\mathcal{X}' = \{v_1, v_2, \ldots, v_n\}$ the set of vertices. Set $\mathcal{X} = \emptyset$.
  3. Repeat until $\mathcal{X}' = \emptyset$.
     (a) Let i be such that $\mathcal{S}_{v_i}$ is the least among all $v_i \in \mathcal{X}'$.
     (b) Set $\mathcal{X} = \mathcal{X} \cup \{v_i\}$.
     (c) Set $\mathcal{X}' = \mathcal{X}' \backslash \{v_k | B_{v_k} \cap B_{v_i} \neq \emptyset\}$.
  4. Construct a graph on the set of vertices in $\mathcal{X}$. Let the cost of an edge in this graph be the distance of the shortest path between the two vertices in G. Construct the minimum spanning tree T of this graph. Consider the subgraph corresponding to T formed by replacing each edge in T by a shortest path of G. Let HEU be a minimum spanning tree of this subgraph. Output HEU.
- *Output:* A tree HEU such that any vertex v_k is within a distance of $2(1+\epsilon)\mathcal{S}_{v_k}$ from some node in HEU and the cost of HEU is at most $2(1+\frac{1}{\epsilon})$ times that of any tree which contains a node within distance $\mathcal{S}_{v_k}$ of any vertex v_k.

Let OPT be an optimal solution. We also use OPT and HEU to denote the cost of the corresponding trees. We prove the performance guarantee of ALGORITHM IDENTICAL in the following lemmas. Let the vertices in $\mathcal{X}$ at the termination of ALGORITHM IDENTICAL be $v_1, v_2, \ldots, v_f$, i.e., $|\mathcal{X}| = f$.

Lemma 28 *Every vertex v_k is within a distance $2(1+\epsilon)\mathcal{S}_{v_k}$ of some vertex in $\mathcal{X}$.*

Proof: If $v_k \in \mathcal{X}$ then the lemma follows since HEU contains v_k. If $v_k \notin \mathcal{X}$ then $\exists v_i \in \mathcal{X}$ such that $B_{v_i} \cap Bv_k \neq \emptyset$ and $\mathcal{S}_{v_i} \leq \mathcal{S}_{v_k}$. In this situation, it is easy to see that v_k is within a distance $(1+\epsilon)\mathcal{S}_{v_i} + (1+\epsilon)\mathcal{S}_{v_k} \leq 2(1+\epsilon)\mathcal{S}_{v_k}$. This completes the proof. □

Lemma 29 $\text{OPT} \geq \sum_{i=1}^{f} \epsilon \mathcal{S}_{v_i}$.

Proof: By definition, OPT contains at least one node from each B_{v_i} for all $v_i \in \mathcal{X}$ that is within distance $\mathcal{S}_{v_i}$ from v_i. Since the B_{v_i} for all $v_i \in \mathcal{X}$ are disjoint, any tree connecting these nodes must cross the peripheral $\epsilon\mathcal{S}_{v_i}$ width and the lemma follows. □

Lemma 30 HEU $\leq 2(\text{OPT} + \sum_{i=1}^{f} \mathcal{S}_{v_i})$.

Proof: We can construct a tree Γ spanning all the $v_i \in \mathcal{X}$ as follows: for each $v_i \in \mathcal{X}$, join v_i to a vertex in OPT that is within distance $\mathcal{S}_{v_i}$ by a shortest path. The length of this path is no more than $\mathcal{S}_{v_i}$. Thus, the cost of Γ is at most OPT $+ \sum_{i=1}^{f} \mathcal{S}_{v_i}$. Note that Γ is a Steiner tree that spans all the vertices in $\mathcal{X}$. Since HEU is a minimum spanning tree on these same vertices, computed using shortest path distances between them, standard analysis of the minimum spanning tree heuristic for Steiner trees, yields that the cost of HEU is at most twice the cost of Γ. The lemma follows. □

Lemma 31 HEU $\leq 2(1 + \frac{1}{\epsilon})$OPT.

Proof: Follows from Lemmas 29 and 30. □

Theorem 20 follows from Lemmas 28 and 31.

3.5.2 Hardness

In this section we prove Theorem 21. We show that if there is a $(1, \beta)$-approximation algorithm for the (Uniform service cost, Total cost, Tree)-bicriteria problem then there is a polynomial-time β-approximation algorithm for the **CONNECTED DOMINATION** problem.

As an instance of the (optimization version of the) **CONNECTED DOMINATION** problem we are given a graph $G(V, E)$ and are required to find a a dominating set D of vertices of minimum size such that the subgraph induced on D is connected.

The result of Feige [Fei96] combined in straightforward fashion with the reduction in [GJ79a] yields the following non-approximability result:

Theorem 32 *Unless* $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\log \log n})$, *the* **CONNECTED DOMINATION** *problem, on a graph with n vertices cannot be approximated to better than a* $\ln n$ *factor.*

We now present the details of the approximation preserving reduction from **CONNECTED DOMINATION** to the (Uniform service cost, Total cost, Tree)-bicriteria problem. Corresponding to an instance $G = (V, E)$ of **CONNECTED DOMINATION**, we create a complete edge weighted graph $G'(V', E')$ as follows: we set $V' = V$. We set the c-cost of each edge in E' to be the length of the shortest path in G, and the uniform service budget S to be 1.

We claim that there exists a connected dominating set of size at most k in G if and only if there exists a solution to the (Uniform service cost, Total cost, Tree)-bicriteria problem with cost at most $(k-1)$. For the only if part, note that any spanning tree for connected dominating set of size k is a tree of cost $(k-1)$ that services all the nodes. Conversely, suppose we have a tree of cost $(k-1)$ servicing all the nodes in G'. Then, the tree has no more than k nodes, and all other nodes are at a distance of 1 from some node in the tree. So, the vertices in the tree form a connected dominating set for G. This completes the proof.

3.6 Diameter and Bottleneck

We first consider the (Non-uniform service cost, Bottleneck, Tree)-bicriteria problem – given a graph with two cost functions on the edges, and a service budget for each node, find a tree such that the service budget (under one cost function) for each node is satisfied and the tree has minimum bottleneck cost under the other cost function (i.e., the cost of the maximum edge in the tree is minimum). This problem can be solved by first sorting the edges in increasing order of the d-costs and adding the edges in that order until one of the connected components in the resulting subgraph satisfies the service constraints for all the nodes. The details are straightforward and so are omitted.

We now consider the (Non-uniform service cost, Diameter, Tree)-bicriteria problem. Using the ideas in [CG82, RSM+94], we can show that the the service-constrained minimum diameter tree problem can be solved in polynomial time. In this problem, we are given a graph $G(V, E)$ and a service radius $\mathcal{S}_{v_i}$ for each vertex v_i. We wish to find a tree with minimum diameter (under the d-costs) such that every vertex v_i is within distance $\mathcal{S}_{v_i}$ (under the c-cost) from some node in the tree.

We only sketch the main idea of the algorithm below. The algorithm uses the roof graph construction in [RSM+94]. Consider the case when the d-costs are integral and polynomially bounded in the size of the graph. Consider OPT – a minimum-diameter service-constrained tree. Let OPT have diameter D. Let x and y be the endpoints of a longest path (under d-cost) in the tree. The weight of this path, D, is the diameter of the tree. Consider the midpoint of this path between x and y. It either falls at a vertex or in an edge in which case we can subdivide the edge by adding a new vertex. First we guess the value of D (there are only a polynomial number of guesses). All the potential midpoints lie in half-integral points along edges of which there are only a polynomial number. From each candidate point we consider the set of nodes within distance $D/2$ and check whether they service all the vertices in the graph. We choose the the least such distance and the correspondingly suitable point and output the breadth-first tree rooted at this point appropriately truncated.

When the edge weights are arbitrary, the number of candidate midpoints are too many to check in this fashion. However, we can use a graphical representation (called the roof curve in [RSM+94]) of the distance of any node from any point along a given edge to bound the search for candidate points. Thus we get the required result in the diameter case.

3.7 Service-Constrained Generalized Steiner Forest

In this section we prove Theorem 23. Given a graph $G(V, E)$, edge cost function c, set of k site pairs (s_i, t_i), and a service constraint $\mathcal{S}$, the service-constrained generalized Steiner forest problem requires us to find a forest $\mathcal{F}$ of minimum cost such that for each site pair (s_i, t_i) there exists a tree $T \in \mathcal{F}$ with the property that *both* s_i and t_i are within a distance $\mathcal{S}$ from the tree T.

ALGORITHM GENERALIZED-STEINER:

- *Input:* An undirected graph $G = (V, E)$, edge cost function c, service budget $\mathcal{S}$, a set of site pairs (s_i, t_i) $1 \leq i \leq p$, an accuracy parameter $\epsilon > 0$.
- 1. Discard all site pairs for which the distance between sites is less than $2(1+\epsilon)\mathcal{S}$, from further consideration. Note that we have violated the service constraint only by a $2(1+\epsilon)$ factor. Henceforth we may assume that for all site pairs the distance between sites is at most $2(1+\epsilon)\mathcal{S}$.
  2. For each site $v_i \in G$, denote by $\mathcal{B}(v_i, r)$ (the ball around v_i of radius r), the set of vertices that are within distance at most r from v_i. (We will refer to v_i as the *center* of ball $\mathcal{B}(v_i, r)$). Consider the balls $\mathcal{B}(v_i, (1+\epsilon)\mathcal{S})$ for each site v_i. Let D be the set of centers of a maximal collection of disjoint balls.
  3. Create an auxiliary graph G' on the collection of centers in D. Set the cost on the edge between vertices v_i and v_j in G' to be the shortest distance between the balls $\mathcal{B}(v_i, (3+2\epsilon)\mathcal{S})$ and $\mathcal{B}(v_i, (3+2\epsilon)\mathcal{S})$. Associate sites v_i not in D to the closest site in D in graph G. This gives a natural association between site pairs in G and pairs of vertices in G'. Solve for an optimal generalized Steiner forest in G'. Let DUAL be the corresponding forest in G. For each $v_i \in D$ connect it to the appropriate tree in DUAL by a path of cost no more than $(3+2\epsilon)\mathcal{S}$. Output this forest HEU.
- *Output:* A forest HEU such that for all i, s_i and t_i are within a distance of $2(1+\epsilon)\mathcal{S}$ from some tree $T \in$ HEU and the cost of HEU is at most $3(1+\frac{1}{\epsilon})$ times that of any optimal service constrained generalized Steiner forest.

Let OPT denote an optimal service constrained forest. We also use OPT and HEU to denote the costs of the corresponding forests. We prove the performance guarantee of ALGORITHM GENERALIZED STEINER by proving a series of lemmas.

Lemma 33 HEU *is a generalized Steiner forest that services all sites within a distance* $2(1+\epsilon)\mathcal{S}$.

Proof: In Step 3 since each site $v_i \notin D$ is associated with the closest site $v_j \in D$ and the balls of radius $(1+\epsilon)\mathcal{S}$ around the sites in D form a maximally disjoint collection it follows that the distance between v_i and v_j is at most $2(1+\epsilon)\mathcal{S}$. □

Lemma 34 *For each site v_i,* OPT *must contain at least one node from the ball $\mathcal{B}(v_i, \mathcal{S})$.*

Proof: This follows from the definition of a service-constrained Steiner forest. □

Lemma 35 OPT $\geq \epsilon|D|S$

Proof: By Lemma 34 OPT strictly services each of the balls $\mathcal{B}(v_i, \mathcal{S})$. Any forest servicing these balls must cross the annular width of ϵS for each ball $\mathcal{B}(v_i, (1+\epsilon)\mathcal{S})$, $v_i \in D$ (since these balls are non-intersecting). This implies that the total peripheral distance covered is at least $\epsilon|D|S$.
□

Lemma 36 OPT $\geq$ DUAL.

Proof: This follows from the fact that the forest corresponding to OPTin G' is a valid generalized Steiner forest on G'. □

Hence, OPT $\geq \max\{|D|\epsilon S, \text{DUAL}\}$.

Lemma 37 HEU $\leq (3+2\epsilon)|D|S +$ DUAL

Proof: This follows from the way HEU is constructed in Step 3. Let $v_i \in D$ and $v_j \notin D$ be two sites such that v_j is associated with v_i in Step 3. From Lemma 33 we have that the distance between v_i and v_j is at most $2(1+\epsilon)\mathcal{S}$. Note that $3+2\epsilon = (1+\epsilon)+(1+\epsilon)+1$ was chosen because there must be a vertex of OPT which is within distance $1 \cdot \mathcal{S}$ of v_j, i.e., within distance $(2(1+\epsilon)+1)\mathcal{S}$ of v_i.
□

Theorem 23 follows from Lemma 33 and 37.

3.8 Concluding Remarks

In this chapter, we focused on the problem of service-constrained network design problems. We formulated a number of these problems and presented general approximation techniques along with nearly-tight hardness results. One natural open problem is to improve our approximation and the hardness results. Another extension that has many potential applications is the node-weighted version of the problems we consider.

Chapter 4

Optical Routing

We consider the problem of conserving bandwidth both at the level of an individual router (switch) and at the level of the network.

4.1 Latin routers: Local Bandwidth Conservation

4.1.1 Motivation

Typical fiber-optic networks are directed netowrks with each point-to-point link being serviced by two oppositely directed optic cables. The restriction that switches have an equal number of input and output ports is entirely natural. It is not *a priori* clear that each entry of the matrix must have *at most* one value, but switches that permit routing of *multiple* wavelengths between the same input and output ports require additional hardware that is both complex and costly [CB95]. Another entirely natural requirement is that of achieving *conflict-free* wavelength routing – the wavelengths assigned to messages at an input port are all distinct and there must be no wavelength conflict at any of the output ports. These restrictions were first modeled by utilizing *Latin routers* ([BH93]) – routing devices that employ the concept of a *Latin Square* (LS).

Definition 3 *An $n \times n$ LS L is an $n \times n$ matrix with the following properties:*

1. $\forall i, j, L_{i,j} \in \{1, 2, \ldots, n\}$
2. $\forall i, j_1 \neq j_2, L_{i,j_1} \neq L_{i,j_2}$
3. $\forall i_1 \neq i_2, j, L_{i_1,j} \neq L_{i_2,j}$

Stated informally, an $n \times n$ LS is a matrix with entries from $\{1, 2, \ldots, n\}$ such that no row or column contains the same element twice. A Latin router with n input ports, n output ports, and n wavelengths is associated with a *partial Latin square* (PLS), an $n \times n$ matrix that specifies the wavelength connections from the n input ports to the n output ports.

Definition 4 *An $n \times n$* Partial Latin Square (PLS) *is an $n \times n$ matrix with entries from the set $\{0\} \bigcup \{1, 2, \ldots, n\}$ (0 is used as a placeholder to denote emptiness) such that each row and each column never contains an element from the set $\{1, 2, \ldots, n\}$ more than once.*

If L is a PLS then a non-zero entry L_{ij} denotes that the wavelength L_{ij} is routed from input port i to output port j. A zero entry denotes an unassigned entry. Thus the state of a Latin router can be described by a PLS. See Fig. 4-1 for an example of a PLS.

0	0	4	3
2	4	0	1
3	1	0	4
4	3	1	2

Figure 4-1: A 4 × 4 PLS

Reducing the number of unassigned or zero entries in the PLS associated with a router is of paramount practical importance in optical networks as this ensures reduced wastage of the valuable resources of ports and wavelengths.

Definition 5 *A PLS S_1 is said to* **extend** *or be an* extension *of a PLS S_2 if S_1 can be obtained by altering only zero entries of S_2.*

Definition 6 *A PLS is said to be* **completable** *if it can be extended to an LS.*

1	2	4	3
2	4	3	1
3	1	2	4
4	3	1	2

Figure 4-2: A 4 × 4 LS

See Fig. 4-2 for an LS obtained by extending the PLS of Fig. 4-1. Not all PLSs can be completed (see Fig. 4-3).

Definition 7 Partial Latin Square Extension Problem (PLSE)*: Given a PLS S_1 find the largest number of zero entries that can be changed to obtain a PLS S_2 that is an extension of S_1.*

The PLSE problem as stated above is an optimization problem. The natural decision version of the problem – namely, given a PLS establish whether it is completable – has been shown to be **NP**-complete [Col84].

4.1.2 Summary of Results

We present two fast approximation algorithms for the problem of maximizing the number of entries that can be added to a partially filled Latin Square. We show that the greedy algorithm achieves a factor of $1/3$. We use insights derived from the linear relaxation of an integer program to obtain an algorithm based on matchings that achieves a better performance guarantee of $1/2$. These are the first known polynomial-time approximation algorithms for the Latin Square completion problem that achieve non-trivial worst-case performance guarantees. These algorithms are easily implementable and have very small constants in their running times thus making them eminently suitable for actual use in real-world optical switches. We also provide strong experimental evidence to show that, in practice, these algorithms are, in fact, near-optimal. In addition, we present BRANCHBOUND – a back-tracking algorithm that finds an optimal solution to the problem of maximizing the number of entries that can be added to a partially filled Latin Square. BRANCHBOUND dynamically prunes the search space to save time. The efficacy of this pruning is demonstrated by the fact that we are able to obtain performance results for *large* Latin routers in our simulations. Without some form of pruning, any attempts at simulation would fail to yield results in a reasonable amount of time.

All the algorithms we present possess provable *worst-case* guarantees. This is very important from a practical standpoint. Heuristics that do not have worst-case performance guarantees tend to be commercially unviable since they cannot provide any insurance against downside risks. And in fact, the results in Figure 4-6 indicate that our algorithms achieve near-optimal *average-case* performance. The average-case performance is of great significance since these algorithms are meant to be used over and over again in on-line situations.

4.1.3 Previous Work

The subject of LSs has been extensively developed by many eminent combinatorialists. Some of the most famous conjectures concerning LSs were proposed by no less than Euler himself. Denes and Keedwell [DK74, DK91] provide encyclopedic collections of results on the combinatorial aspects of LSs. Of special interest to us are results concerning the completion of PLSs. The most famous conjecture in this area was the Evans conjecture [Eva60] which was proved after a period of over 20 years by Smetaniuk [Sme81]. An excellent survey of the ongoing attempt to characterize completable PLSs appears in [LE77].

The computational aspect of completing PLSs was initiated by Rosa [Ros77] and Giles, Oyama, and Trotter [GOT77]. The issue was finally resolved by Colbourn [Col84] who proved that the problem of deciding whether a PLS is completable is **NP**-complete.

Barry and Humblet [BH93] were the first to recognize the applicability of LSs to the problem of wavelength assignment in optical networks. The various policies for wavelength routing, and the corresponding assignments of wavelengths along the routes, that have been reported in the literature [CGK92, LL93, RS94, ZA94] aim at minimizing congestion, average hop-distance, call blocking probability or maximizing traffic throughput, and clear channel density. Chen and Banerjee [CB95] realized that these policies often lead to

scenarios where a significant number of wavelengths, available at the input and output ports of the routers, cannot be used due to potential wavelength conflict. In [CB95], they focus on maximizing the wavelength utilization at a Latin router by obtaining good solutions to the PLSE problem. They show that achieving improved wavelength utilization also automatically improves overall network performance. They present two algorithms, Algorithm 1 – a backtracking algorithm that takes exponential time in the worst case and Algorithm 2 – a heuristic that could potentially modify preassigned routes, thus rendering it unfit for use in most situations of practical interest.

The rest of this part on Latin routers is organized as follows: Section 4.1.4 contains notation and some basic lemmas; Section 4.1.5 contains the factor $1/3$ approximation algorithms; Section 4.1.6 contains the factor $1/2$ approximation algorithms; Section 4.1.7 contains a description of algorithm BRANCHBOUND; Section 4.1.8 contains experimental results; Section 4.1.9 answers some natural questions regarding extensions of certain PLSs; and Section 4.1.10 closes with a conjecture that we would be interested in seeing settled.

4.1.4 Preliminaries

Let L be a PLS. If $L_{i,j} = 0$, we say the cell (i,j) is **empty**. Conversely, if $L_{i,j} \neq 0$, we say the cell (i,j) is **filled**. Two PLSs L and M are said to be **compatible** if

- $\forall i,j, L_{ij} = 0$ or $M_{ij} = 0$, and
- $L + M$ is a PLS.

When L and M are compatible LSs we shall denote $L+M$ by $L \oplus M$. For a PLS L, let $|L|$ denote the number of non-empty cells of L. We write $L \subseteq M$ ($L \subset M$) for two PLSs when $M = L \oplus A$ for some (non-trivial) PLS A. This is equivalent to saying that L may be extended to M. We call L **blocked** if $\nexists L' \supset L$. For PLS L, define $L^{\perp}$ to be a **canonical** compatible LS if $|L^{\perp}|$ is the largest over all compatible LSs.

The problem of extending a PLS can be viewed graph-theoretically as a coloring problem. Associate with an $n \times n$ PLS L the colored graph with n^2 vertices $\langle i,j \rangle, 1 \leq i,j \leq n$ and edges $\{(\langle i,j \rangle, \langle i',j' \rangle) \mid i = i' \text{ or } j = j'\}$ such that vertex $\langle i,j \rangle$ is assigned color $L_{ij} \neq 0$; vertices corresponding to zero entries of L are considered to be uncolored. The problem of PLS extension can now be viewed equivalently as the problem of coloring additional vertices given the corresponding partially colored graph. This motivates our use of the terminology *color* for the entries of a PLS. We use the terminology **degree of freedom** of a 0 entry in a PLS to refer to the number of colors that can be validly used in that entry. See Figure 4-1. The color of the entry $(1,3)$ is 4. The degree of freedom of the $(2,3)$ entry is 1 since there is only one color that can be validly used in that entry, namely the color 3.

Definition 8 *The* **density** *of an $n \times n$ PLS is the fraction of entries that are non-zero. $\mu(L)$ is used to denote the density of L.*

The PLS in Figure 4-1 has density $3/4$. An LS is a PLS that has density 1.

Combinatorial Results

Colbourn's result, [Col84], showing that PLS-completability is **NP**-complete has effectively destroyed any hopes of discovering a polynomial time algorithm for recognizing completable PLSs. It remains an intriguing problem to understand what can be salvaged. We take a combinatorial step in this direction by providing a quantitative characterization of minimally non-completable PLSs and minimally non-extendible or blocked PLSs.

Definition 9 *Let $f(n)$ be the largest number such that every $n \times n$ PLS L with $|L| \leq f(n)$ is completable.*

Lemma 38 $f(n) = n - 1$.

Proof: The Evans conjecture, ([Eva60]) made in 1960, states that any $n \times n$ PLS L with $|L| \leq n - 1$ is completable. It was finally settled in the affirmative by Smetaniuk ([Sme81]) in 1981. This gives us that $f(n) \geq n - 1$. That $f(n) < n$ is easily seen by the first two PLSs of Fig. 4-3 which cannot be completed. Hence $f(n) = n - 1$. □

Definition 10 *Let $g(n)$ be the largest number such that every $n \times n$ PLS L with $|L| \leq g(n)$ is extendible.*

Lemma 39 $g(n) = \lceil \frac{n^2}{2} \rceil - 1$.

Proof: We first show that $g(n) \geq \frac{n^2}{2} \geq \lceil \frac{n^2}{2} \rceil - 1$. Consider any $n \times n$ PLS L such that $|L| < \frac{n^2}{2}$. Let r_i (c_j)be the set of non-zero entries in row i (column j) of L. If we show that there exists an i, j such that $L_{ij} = 0$ and $|r_i| + |c_j| < n$, then we are done because it implies that L can extended by setting L_{ij} to a value in $\{1, 2, \ldots, n\} - r_i - c_j$. It remains to show that there exists an i, j such that $L_{ij} = 0$ and $|r_i| + |c_j| < n$. We do this by invoking the Cauchy-Schwartz inequality to show that the expectation $\mathrm{E}[n - |r_i| + n - |c_j| : L_{ij} = 0] = (\frac{1}{n^2 - |L|})(\sum_i (n - |r_i|)^2 + \sum_j (n - |c_j|)^2) \geq (\frac{1}{n^2 - |L|})(\frac{(n^2 - |L|)^2}{n} + \frac{(n^2 - |L|)^2}{n}) = 2(n - \frac{|L|}{n}) > n$. It is easy to see that $g(n) < \lceil \frac{n^2}{2} \rceil$ by considering the general versions of the last two examples in Fig. 4-4. □

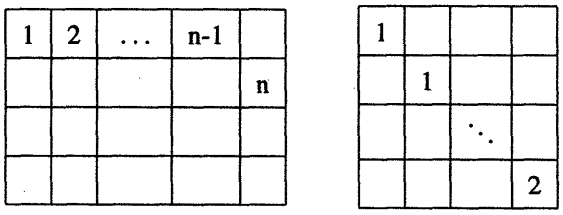

Figure 4-3: Blocked PLSs with n entries

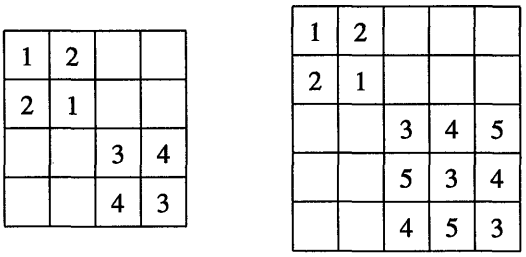

Figure 4-4: Blocked PLSs with $\lceil \frac{n^2}{2} \rceil$ entries

4.1.5 Greedy Algorithms

A Greedy Algorithm Based on Linear Programming

The problem of maximally extending a PLS L may be expressed as an integer program:

$$\max \textstyle\sum_{ijk} x_{ijk} \tag{4.1}$$

subject to

$$\forall j,k \textstyle\sum_i x_{ijk} \leq 1,\ \forall i,k \textstyle\sum_j x_{ijk} \leq 1,\ \forall i,j \textstyle\sum_k x_{ijk} \leq 1$$

$$\forall i,j,k \text{ such that } L_{ij} = k \neq 0, x_{ijk} = 1 \tag{4.2}$$

$$\forall i,j,k, x_{ijk} \in \{0,1\}. \tag{4.3}$$

The association of a feasible point m with the PLS $M_{ij} = k \iff m_{ijk} = 1$ is a natural correspondence between those LSs which extend L and the feasible points of the integer program. A variable which shares no constraint with any variable of (4.2) shall be called *free*.

Relaxing this integer program to a linear program yields a natural greedy algorithm:

GREEDY (LP):

1. Set $t = 0$. Set $A^0_{ij} = 0$.
2. If $L \oplus A^t$ is blocked, return A^t. Otherwise let x^* be an optimal solution to the linear program for $L \oplus A^t$ and $(\hat{i},\hat{j},\hat{k})$ be so that $x^*_{\hat{i}\hat{j}\hat{k}}$ is a maximum free variable. Set

$$A^{t+1}_{ij} = \begin{cases} \hat{k} & \text{if } (i,j) = (\hat{i},\hat{j}) \\ A^t_{ij} & \text{otherwise.} \end{cases}$$

Increment t and begin step 2 again.

If L is not blocked there exists a free variable so that step 2 may proceed. Furthermore, if x^*_{ijk} is free, k is a consistent assignment to cell L_{ij} so that each augmentation made by step 2 results in a (larger) PLS $L \oplus A^{t+1}$.

Lemma 40 *Let L be a PLS and t the number of iterations performed by* GREEDY (LP) *on L. Then $t \geq$*

$(\frac{1}{3-\frac{2}{n}})|L^{\perp}|$.

Proof: For a PLS L, let ϕ_L be the optimal value of the associated linear program. Notice that during each iteration of the algorithm, at least one constraint containing a free variable is tight, so that the $x_{\hat{i}\hat{j}\hat{k}}$ selected has value at least $1/n$. Examine iteration s of the algorithm. Let x^* be an optimal solution to the linear program for $L \oplus A^s$ having value ϕ_s and $(\hat{i}, \hat{j}, \hat{k})$ the triple selected in this iteration. Define

$$a_{ijk} = \begin{cases} 1 & \text{if } (i,j,k) = (\hat{i},\hat{j},\hat{k}) \\ 0 & \text{if } (i \neq \hat{i} \text{ and } (j,k) = (\hat{j},\hat{k})) \text{ or } (j \neq \hat{j} \text{ and } (i,k) = (\hat{i},\hat{k})) \text{ or} \\ & \qquad (k \neq \hat{k} \text{ and } (i,j) = (\hat{i},\hat{j})) \\ x^*_{ijk} & \text{otherwise.} \end{cases}$$

Notice that a is a feasible solution to the linear program for $L \oplus A^{s+1}$ with value at least $\phi_s - (2 - \frac{2}{n})$. Each iteration, then, depresses the objective function of the associated linear program by at most $(2 - \frac{2}{n})$ whence $t \geq \frac{\phi_L - |L|}{3+\frac{2}{n}} \geq \frac{|L^{\perp}|}{3+\frac{2}{n}}$. □

Recall that $|A^t| = t$ so that GREEDY (LP) achieves a $\frac{1}{3-\frac{2}{n}}$ approximation factor. This proves the following theorem.

Theorem 41 GREEDY (LP) *is a* $(\frac{1}{3} + \Omega(\frac{1}{n}))$*-approximation algorithm.*

The Naive Greedy Algorithm

Lemma 42 *Let L be a PLS and A, B two PLSs, each compatible with L, so that $L \oplus B$ is blocked. Then* $|B| \geq \frac{1}{3}|A|$.

Proof: For each pair (i,j) with $B_{ij} \neq 0$, let $S_{ij} = \{(i,j)\} \cup \{(i,j') \mid B_{ij} = A_{ij'}\} \cup \{(i',j) \mid B_{ij} = A_{i'j}\}$. Then $|S_{ij}| \leq 3$. If $|A| > \sum_{ij} |S_{ij}|$ then there is a pair (u,v), appearing in no S_{ij}, so that A_{uv} is non-empty. In this case, $(L \oplus B)_{uv}$ may be consistently set to A_{uv}, contradicting that $L \oplus B$ is blocked. Hence $|A| \leq \sum_{ij} |S_{ij}| \leq 3|B|$. □

Consider the greedy algorithm defined as follows:

GREEDY:

1. Set $t = 0$. Set $A^0_{ij} = 0$.
2. If $L \oplus A^t$ is blocked, return A^t. Otherwise, select a pair $(\hat{i}, \hat{j})$ with $(L \oplus A^t)_{\hat{i}\hat{j}} = 0$ and a color $\hat{k}$ so that

$$A^{t+1}_{ij} = \begin{cases} \hat{k} & \text{if } (i,j) = (\hat{i},\hat{j}) \\ A^t_{ij} & \text{otherwise.} \end{cases}$$

is compatible with L. Increment t and begin step 2 again.

Since GREEDY computes an extension A^k so that $L \oplus A^k$ is blocked, $|A^k| \geq \frac{1}{3}|L^\perp|$. This proves the following theorem.

Theorem 43 GREEDY *is a* $\frac{1}{3}$*-approximation algorithm.*

The example in Fig. 4-2 demonstrates that our analysis of the performance of the greedy algorithm is tight. This PLS can be filled to completion. However, an incorrect choice by GREEDY to fill 2 in $(1,1)$ blocks the LS. The greedy algorithm can be implemented in $O(n^3)$ time.

4.1.6 Approximation Algorithms Based on Matching

A Linear Programming Based Algorithm Using Matching

We again consider the linear program associated with a PLS L.

MATCHING (LP):

1. Set $A^0_{ij} = 0$. Carry out the following for each $k = 1, \ldots, n$. Let x^* be a solution to the linear program associated with $L \oplus A^{k-1}$. If $\forall i,j\ x^*_{ijk} = 0$, define $A^k = A^{k-1}$ and move on to the next k. Construct the weighted bipartite graph $G = (U, V, E, w : E \to \mathbb{Q}^+)$ with $U = V = \{1, \ldots, n\}$, $E = \{(u,v) \mid x^*_{uvk} \neq 0\}$, $w(u,v) = x^*_{uvk}$. Select a matching M which maximizes $\frac{|M|}{|M|+\|G\|-\|M\|}$ where $|M|$ is the cardinality of the matching, $\|M\|$ is the weight of the matching, and $\|G\| = \sum_{e \in E} w(e) = \sum_{ij} x^*_{ijk}$ is the total weight of G. Since M is a matching, the variables associated with the edges of M are independent (that is, none of these variables occur together in a constraint) and we may define the PLS

$$A^k_{ij} = \begin{cases} k & \text{if } (i,j) \in M \\ A^{k-1}_{ij} & \text{otherwise.} \end{cases}$$

   Furthermore, each edge of M corresponds to a non-zero variable so that L and A^k are compatible.

2. Return A^n.

Notice that a matching optimizing the quantity $\frac{|M|}{|M|+\|G\|-\|M\|}$ may be computed in polynomial time by computing a maximum weight matching of each cardinality $c \in \{1, \ldots, n\}$ for which a matching exists and selecting the optimum (see [Tar83], for example). Hence the algorithm runs in polynomial time. To show that this is a 1/2 approximation algorithm, we first prove the following lemma.

Lemma 44 *Let* $G = (U, V, E, w : E \to \mathbb{Q}^+)$ *be a weighted bipartite graph with* $\forall u, \sum_v w(u,v) \leq 1$. *Then for any maximum cardinality matching* M, $|M| \geq \|G\|$. *Hence* $\max_M \frac{|M|}{|M|+\|G\|-\|M\|} \geq \frac{1}{2}$.

Proof: To begin with, we show that for any maximum matching M in G, there is a subset of vertices W such that:

(i) W covers each edge in E, and

(ii) each edge in M is covered by exactly one vertex in W.

A vertex v covers an edge (x, y) if either $x = v$ or $y = v$. It is easy to see that picking either of the vertices of every edge in M always satisfies the second requirement trivially. Suppose the first requirement is not met. In other words, an edge $(u_0, v_1) \notin M$ is not covered by any vertex in the current W. By maximality of M, there is some $(u_1, v_1) \in M$ such that $u_1 \in W$, by condition (ii). Now, let $W = W \backslash \{u_1\} \cup \{v_1\}$. If the first condition is met, we are done. Otherwise, it implies there is an edge $(u_1, v_2) \notin M$ such that it is not covered by the current W. We repeat the same process now. It is clear that we cannot go indefinitely. When we terminate, we see that $(u_0, v_1), (v_1, u_1), \ldots, (u_{k-1}, v_k)$ is an augmenting path, contradicting the maximality of M.

Since for any vertex u, $\sum_v w(u, v) \leq 1$, the above shows that $|M| \geq \|G\|$. □

Theorem 45 MATCHING (LP) *is a* $\frac{1}{2}$*-approximation algorithm.*

Proof: We now consider the effect that stage t of the above algorithm has had on the optimal solution to the linear program. Let ϕ_{t-1} be the optimal value of the linear program associated with $L \oplus A^{t-1}$ and x^* a vector achieving this optimal value. Consider the vector

$$a_{ijk} = \begin{cases} 1 & \text{if } k = t \text{ and } (i,j) \in M \\ 0 & \text{if } (k = t \text{ and } (i,j) \notin M) \text{ or } (k \neq t \text{ and } (i,j) \in M) \\ x^*_{ijk} & \text{otherwise.} \end{cases}$$

a is a feasible solution to the linear program associated with $L \oplus A^t$ and $\sum_{ijk} a_{ijk} \geq \phi_{t-1} - \|G\| + \|M\|$. Hence $\phi_t \geq \phi_{t-1} - \|G\| + \|M\|$. In this case we have set $|M|$ variables and depressed the optimum value of the linear program by at most $\|G\| - \|M\|$. From the above lemma $\frac{|M|}{|M|+\|G\|-\|M\|} \geq \frac{1}{2}$, so that the above algorithm is a $\frac{1}{2}$-approximation algorithm. □

A Combinatorial Algorithm Using Matching

Consider a PLS L and define $L^t = \{(i,j) \mid \forall s\, L_{is} \neq k, \forall s\, L_{sj} \neq k\}$ to be the collection of cells which will admit a t. Consider the following algorithm:

MATCHING:

1. Set $A^0_{ij} = 0$.
2. For each $k = 1, \ldots, n$, consider the bipartite graph $G = (U, V, E)$ with $U = V = \{1, \ldots, n\}$ and $E = L^k$. Let M be maximum matching in G. Set

$$A^k_{ij} = \begin{cases} k & \text{if } (i,j) \in M \\ A^{k-1}_{ij} & \text{otherwise.} \end{cases}$$

3. Return A^n.

Consider stage k of the above algorithm. Define

$$P^k_{ij} = \begin{cases} 0 & \text{if } ((L \oplus A^{k-1})^{\perp}_{ij} = k) \text{ or } ((i,j) \in M) \\ (L \oplus A^{k-1})^{\perp}_{ij} & \text{otherwise.} \end{cases}$$

Notice that P^k is always compatible with $L \oplus A^k$ so that $|(L \oplus A^k)^{\perp}| \geq |P^k| \geq |(L \oplus A^{k-1})^{\perp}| - 2|M|$. (Since M is a maximum matching, $(L \oplus A^{k-1})^{\perp}$ can have no more than $|M|$ cells assigned to k.) This proves the following theorem.

Theorem 46 MATCHING *is a* $\frac{1}{2}$*-approximation algorithm.*

The following example (Fig. 4-5) demonstrates that our analysis of the performance of the MATCHING algorithm is in fact tight. The PLS (left) can in fact be filled to completion (right), but a bad choice of matching can block it (middle).

	2	3	
2			1
3			2
	1	2	

4	2	3	
2	4		1
3		4	2
	1	2	4

1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

Figure 4-5: A worst-case scenario for the matching algorithm

We repeat the matching step for each of the n colors. Each matching step can be performed in $O(n^{2.5})$ by the Hopcroft-Karp algorithm ([HK73]). Therefore, this algorithm runs in $O(n^{3.5})$ time.

4.1.7 Branch and Bound Algorithm

Backtracking is a general and proven method for exhaustively going through the space of all possible solutions. BRANCHBOUND does a search of the (potentially $(n+1)$-ary) tree corresponding to the choices of colors for each 0 entry in the PLS. But since the number of possible nodes in the tree is hugely exponential - $\Theta(n^{n^2})$ - BRANCHBOUND employs four techniques to cut down the running time.

Preprocessing. To prevent costly recomputation at each step BRANCHBOUND sets up an elaborate data structure ahead of time. This data structure permits quick table lookup of the set of possible colors that can be validly used in a 0 entry. The data structure also permits a quick lookup of the list of invalidations that would be caused by setting entry $L_{i,j}$ to color k, for all i, j, k.

Implicit Depth first search. The entire algorithm is coded in a conventional higher-level language. To save time spent processing function calls BRANCHBOUND is implemented as an iterative loop that implicitly traverses the tree in depth first fashion.

Degree of Freedom. To reduce the number of backtracking steps the degree of freedom concept is used in a fashion identical to that of Algorithm 1 of [CB95]. The order of 0 entries tried for backtracking is based on the degree of freedom of the entries.

Dynamic Pruning. At any point the algorithm maintains the best solution found so far. At each node it does a quick evaluation (using the data structure) and linear programming to figure out if the best potential solution in the subtree rooted at that node could better the best solution found so far. If so it continues the search else it aborts the search at that node. This is also known as alpha-beta pruning in the game theory literature.

It is clear that BRANCHBOUND obtains a solution that is as good as that of OPT. Since the problem is NP-complete, BRANCHBOUND could potentially take $\Omega(n^{n^2})$ time.

4.1.8 Experimental Results

To study the applicability of our approximation algorithms in practice, we performed simulation studies. We are mainly concerned with the quality of the output (i.e., the number of filled entries in the final PLS) and the time taken for execution. We then compare the performance against BRANCHBOUND (which is OPT, as far as the quality of the result is concerned). The simulations were conducted on 4×4 to 9×9 PLSs that were randomly filled with initial densities from 20% to 80%. The performance of various algorithms are shown in Figure 4-6. The table shows the output densities obtained and execution time (in ms) averaged over several trials (100 trials). The simulations were written in C++ and run on a Sparc-20. The results in the tables can be taken in at a glance by inspecting the graphs in Figure 4-7.

From our results, it is clear that the approximation algorithms with their heuristics perform very well in practice. As expected, BRANCHBOUND runs fast when the density is large (since there are less choices). This suggests a combined approach in practice – using the approximation algorithms at low densities and BRANCHBOUND at higher densities.

Algorithm		4 × 4 PLS				5 × 5 PLS				6 × 6 PLS			
	Initial%	20	40	60	80	20	40	60	80	20	40	60	80
GREEDY	Final%	96	87	85	83	91	85	86	86	90	87	85	85
	Time	0	0.08	0.08	0.12	0.08	0.08	0.32	0.12	0.24	0.08	0.12	0.12
MATCH	Final%	100	87	87	83	94	91	87	86	98	91	86	85
	Time	1	0.52	0.76	0.36	1.24	1.08	1	0.68	1.88	1.64	1.4	1.08
BRANCHBOUND	Final%	100	100	87	83	100	93	87	86	100	95	88	86
	Time	2.52	1.12	0.32	0.16	9.84	2.64	0.68	0.12	75.04	14.68	2.04	0.16

Algorithm		7 × 7 PLS				8 × 8 PLS				9 × 9 PLS			
	Initial%	20	40	60	80	20	40	60	80	20	40	60	80
GREEDY	Final%	87	87	85	85	89	88	86	88	91	89	86	85
	Time	0.24	0.28	0.08	0	0.25	0.32	0.32	0.2	0.6	0	0.32	0.12
MATCH	Final%	96	92	88	85	100	94	89	88	97	95	88	85
	Time	3	2.72	2.16	1.64	4	3.4	2.76	2.16	5.6	4.7	3.48	2.76
BRANCHBOUND	Final%	100	98	90	85	100	98	90	88	100	99	91	85
	Time	1227	153	7.04	0.4	589	12k	16.16	0.4	246k	401k	144.56	0.48

Figure 4-6: Experimental evaluation of the algorithms

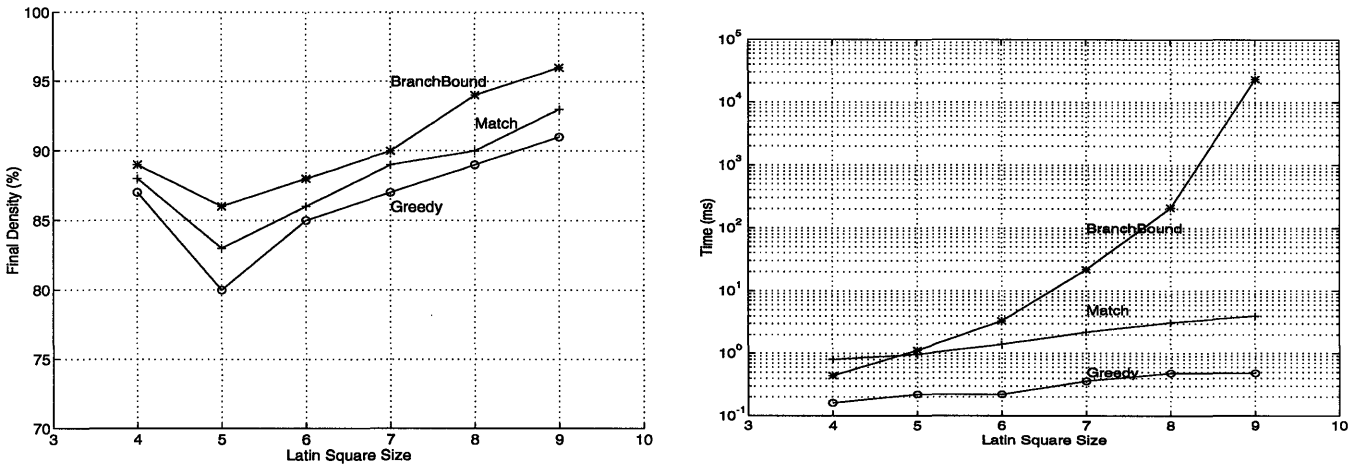

Figure 4-7: Performance of algorithms when initial density = 50%

4.1.9 Extending Blocked PLSs

In many applications, the problem of completing a blocked PLS with new available colors is significant. A natural question is this: given a blocked $n \times n$ PLS L, how many extra colors are necessary to complete it. This can be answered exactly (in polynomial time) by constructing the bipartite graph G_L on the $2n$ vertices, $R_i, C_j, 1 \leq i, j \leq n$, such that there is an edge between R_i and C_j iff $L_{ij} = 0$; and observing by Hall's theorem [Hal48] that the edge set of this bipartite graph can be partitioned into k^* disjoint matchings where k^* is the maximum degree of a vertex in the bipartite graph defined above. By coloring each such matching with a new color, we ensure that there are no conflicts generated. Thus, k^* new colors suffice. Notice that k^* colors are indeed necessary since some node of G_L has degree k^*.

In fact, one can see that $k^* \leq n/2$ by a proof similar to that of Lemma 39. And, in fact k^* can be equal to $n/2$, as can be seen from the two examples in Fig. 4-4.

A related question is this: given a blocked L, and k new colors, what is the maximum number of entries that can be filled using these new colors. For $k = 1$, it is equivalent to finding the maximum match-

ing in G_L (defined above) and hence can be exactly computed. For $k > 1$, this problem is equivalent to finding disjoint matchings $M_1, \ldots, M_k$ in G_L such that $\sum_{i=1}^{k} |M_i|$ is maximized. This number can be exactly computed by computing a maximum flow on the following graph G'_L. $V(G_{L'}) = V(G_L) \cup \{s,t\}$, $E(G_{L'}) = E(G_L) \bigcup \cup_{i=1}^{n}(s,R_i) \bigcup \cup_{j=1}^{n}(C_j,t)$, and $c(e) = 1$ if $e \in E(G_L)$ and $c(e) = k$ otherwise. It is easy to extract the actual color assignment to edges from the maximum flow graph.

4.1.10 Concluding Remarks

In this section we presented three algorithms to complete a Latin router in which some of the entries may have been preassigned. Two of the algorithms are fast polynomial time approximation algorithms with provable worst-case bounds. Unlike Algorithm 2 of [CB95], neither of these algorithms changes preassigned entries. GREEDY possesses the extra advantage of being usable in a dynamic situation where routing requests are online. The third is an exhaustive search-based algorithm that uses pruning to dynamically shrink the search-space. BRANCHBOUND is a more efficient version of Algorithm 1 of [CB95],

These algorithms have applications to the problem of lightpath assignment in wide area wavelength routing optical networks. We also present simulation results which show that on average the two approximation algorithms are near-optimal, in addition to being very fast.

We now present a conjecture we would like to see settled. Define the *Latin square polytope* to be $\mathfrak{L}_n = \{x \in (\mathbb{R}^n)^3 \mid \forall i,j,k\ x_{ijk} \geq 0, \forall j,k \sum_i x_{ijk} \leq 1, \forall i,k \sum_j x_{ijk} \leq 1, \forall i,j \sum_k x_{ijk} \leq 1\}$. We conjecture the following:

Conjecture 1 *For every vertex* $v \in \mathfrak{L}_n$, $\forall i,j,k, v_{ijk} = 0$ *or* $v_{ijk} \geq \frac{1}{n}$.

This would show that MATCHING(LP) achieves a factor of $\frac{1}{2} + \Omega(\frac{1}{n})$.

4.2 Network: Global Bandwidth Conservation

4.2.1 Motivation

Though the number of wavelengths at a given node is limited by the size of the switch, nevertheless, it is possible to build a transparent wide-area optical network by *spatial reuse* of wavelengths. Figure 1-1 shows an all-optical wavelength routing network with two 3 × 3 switches. Though each switch has only 3 input and output ports, it is still possible to set up all the shown interconnections by (re)using wavelength **1** both for the $A - B$ and $C - D$ routes.

4.2.2 Summary of Results

Bandwidth is a very valuable resource in wavelength division multiplexed optical networks. The fundamental problem in bandwidth utilization in optical networks can be modeled as follows:

Definition 11 (OPTICAL ROUTING:) *Given a directed (undirected) graph G and a set P of n requests, each of which is a pair of nodes to be connected by a directed (undirected) path, find a set of paths corresponding to these requests and an assignment of wavelengths to the paths so as to minimize the total number of wavelengths used. The assignment should ensure that different paths allotted the same wavelength must be edge-disjoint.*

We present a polynomial-time algorithm for this problem on fixed constant-size topologies. We combine this algorithm with ideas from Raghavan and Upfal [RU94] to obtain an optimal assignment of wavelengths on constant degree *undirected* trees. Mihail, Kaklamanis, and Rao [MKR95] posed the following open question: what is the complexity of this problem on directed trees? We show that it is **NP**-complete even on binary and constant depth directed trees.

4.2.3 Previous Work

OPTICAL ROUTING has been studied in great detail on a number of different and fundamental topologies by Raghavan and Upfal [RU94]. This problem is **NP**-complete for undirected trees by reduction from edge coloring. They gave a $9/8$-approximation algorithms for undirected trees and a 2-approximation algorithm for rings. They also present approximation algorithms for expanders, meshes, and bounded-degree graphs. Subsequently, Mihail, Kaklamanis, and Rao [MKR95] consider the problem on directed trees (these are graphs obtained by replacing each edge of an undirected tree by two directed edges in opposite directions). They obtain a $15/8$-approximation algorithm using potential function arguments. Aumann and Rabani [AR95] consider routing permutations on arrays, hypercubes, and arbitrary bounded degree networks, while Bermond et. al. [JCGP+96] study the problems of broadcasting and gossiping in optical networks.

Section 4.2.4 contains some notation; Section 4.2.5 presents a polynomial time algorithm for the problem of OPTICAL ROUTING on constant-size topologies. Section 4.2.6 deals with hardness of routing on constant-degree trees. Section 4.3 lists some interesting observations on OPTICAL ROUTING on trees.

4.2.4 Preliminaries

For an undirected graph G, let $\chi'(G)$ denote its *chromatic index* (i.e., the edge color number). ($\chi(G)$ denotes the vertex color number, as usual. We use $\chi'(G)$ both to denote an optimal edge coloring as well as the number of colors used, i.e. the chromatic index. For edge e, $\chi'(e)$ denotes the color of the edge. When G is a tree, for any given $u, v \in V$, the path from u to v is unique and hence OPTICAL ROUTING is equivalent to path coloring on trees. Given an instance of OPTICAL ROUTING with a graph G and a (multi)set $P(|P| = n)$ of paths, we use $\Lambda(G, P)$ to denote both an assignment of wavelengths and the total number of wavelengths used; we use $\lambda(p)$ to denote the wavelength of path p. $\Lambda^*(G, P)$ denotes the optimal number of wavelengths for the set of requests P. Note that wavelengths can also be interpreted as colors.

4.2.5 Constant Size Graphs

We now present an exact polynomial-time algorithm for the problem of OPTICAL ROUTING on a fixed constant-size topology, G (i.e., k is constant). This problem is non-trivial – a combinatorial approach seems difficult. We present the algorithm only for undirected graphs. The algorithm for directed graphs is a straightforward modification.

Theorem 47 OPTICAL ROUTING *on constant-size topologies is in* **P**.

Proof: Let r_{ij} denote the number of paths from node i to j. Note that r_{ij} could potentially be as large as n.

Let a path-matching be a collection of edge-disjoint simple paths. Let $\mathcal{M}_G = \{M \mid M \text{ is a path-matching in } G\}$ be the set of path-matchings in G. Note that since G is constant-size, so is $\mathcal{M}_G$.

Consider the following integer program (IP):

The optimization function:

$$\min \sum_{M \in \mathcal{M}_G} x_M$$

subject to the constraints:

$$\forall i, j, \sum_{i \sim j \in M} x_M \geq r_{ij}.$$

It is easy to see that the above IP models the problem of OPTICAL ROUTING exactly. A solution to the IP yields $\Lambda^*(G, P)$, the optimum number of wavelengths needed. Though the IP has terms in the constraints which are linear in n, the optimizing function has only a constant number of variables. This means we can solve the IP exactly using Lenstra's polynomial-time algorithm for integer programs in fixed dimension [Len83]. □

Thus we have an exact polynomial-time algorithm for the problem of OPTICAL ROUTING on a fixed constant-size topology.

4.2.6 Constant Degree Trees

First, we show that for undirected trees of constant degree, the problem of coloring paths is in **P**. Then, we show that for directed trees, this problem is **NP**-complete even for binary trees.

Theorem 48 *Coloring paths on undirected bounded-degree trees is in* **P**.

Proof: Let T be an undirected tree with vertex set V and edge set E. The problem now is to assign colors in an optimal fashion to paths in P. Observe that the following nice decomposition result holds: let the removal of edge $e \in E$ result in two trees $T_1(V_1, E_1)$ and $T_2(V_2, E_2)$. P is partitioned into P_1 (the paths entirely in T_1), P_2 (the paths entirely in T_2), and P_{12} (the paths that go through e). Consider the trees $(V_1, E_1 \cup \{e\})$ with paths $P_1 \cup P_{12}$ and $(V_2, E_2 \cup \{e\})$ with paths $P_2 \cup P_{12}$. If we can color both the above instances optimally then

we can combine the colors of the two to get an optimal coloring of P on T. Thus we need only to solve the problem on stars (trees with one central vertex and many leaves) and then we can put the solutions together to get a solution for the whole. But since T is constant degree, the stars we get from breaking up the tree are constant-sized and so we can use the results from Theorem 47 to get optimal solutions for these stars.

Thus we have an exact polynomial-time algorithm for the problem of OPTICAL ROUTING on a constant-degree undirected trees. □

To prove the next theorem, we need some basic results from vertex coloring interval and circular-arc graphs. Recall that an interval (circular-arc) graph is the intersection graph of intervals(arcs) on a line (circle). Intervals graphs can be vertex colored in linear time. Vertex coloring circular-arc graphs is **NP**-hard [GJ79b].

Theorem 49 *Coloring paths on directed binary trees is* **NP***-complete.*

Proof: We reduce the problem of vertex coloring circular-arc graphs to the problem of coloring paths on a directed binary tree. As an instance of the vertex-coloring problem we are given a circular-arc graph G and an integer k, and asked whether the chromatic number $\chi(G) = k$? Our idea is to embed a circle on the tree. We consider the circular-arc representation of G, which is computable in polynomial-time [Gol80]. We "cut" G at two distinct points p_1, p_2 on the circle to partition G into two "pieces" G_1, G_2. Note that G_1 and G_2 are collections of intervals on a line. Note also that there are intervals in both G_1 and G_2 that correspond to the same arc in G. We construct a directed tree as follows: we take a suitably long directed path and pick a vertex in the middle to be the root. This gives us a tree with two leaves. We embed the interval graph G_1 on this directed tree as a set of directed paths going from left to right and G_2 going from right to left. We now need to ensure that the same arc in G_1 and G_2 gets assigned the same color. We do this by attaching a binary tree to each end of the tree. The paths corresponding to the same arc in G_1 and G_2 are then extended into these trees until they can be split down two adjacent leaves. We then add $k-1$ directed paths across these two leaves thus ensuring that the two extended paths get the same color in any k-coloring. See Fig. 4-8 for an illustration of this reduction for a 3-node circular-arc graph. The resulting tree and set of directed paths constitute the instance of the OPTICAL ROUTING problem on directed trees.

It is easy to see that the resulting instance of OPTICAL ROUTING is k-colorable if and only if the original circular-arc graph can be vertex colored with k colors.

□

4.3 Concluding Remarks

An interesting parameter is the thickness denoted $L_{\max}$. Given a tree T and set of paths P, $L_{\max}$ is defined to be the maximum thickness of of the paths at any point in T. When $L_{\max} = 1$, one color suffices since there is no overlap of paths. When $L_{\max} = 2$, we can show that the set of paths can be decomposed into

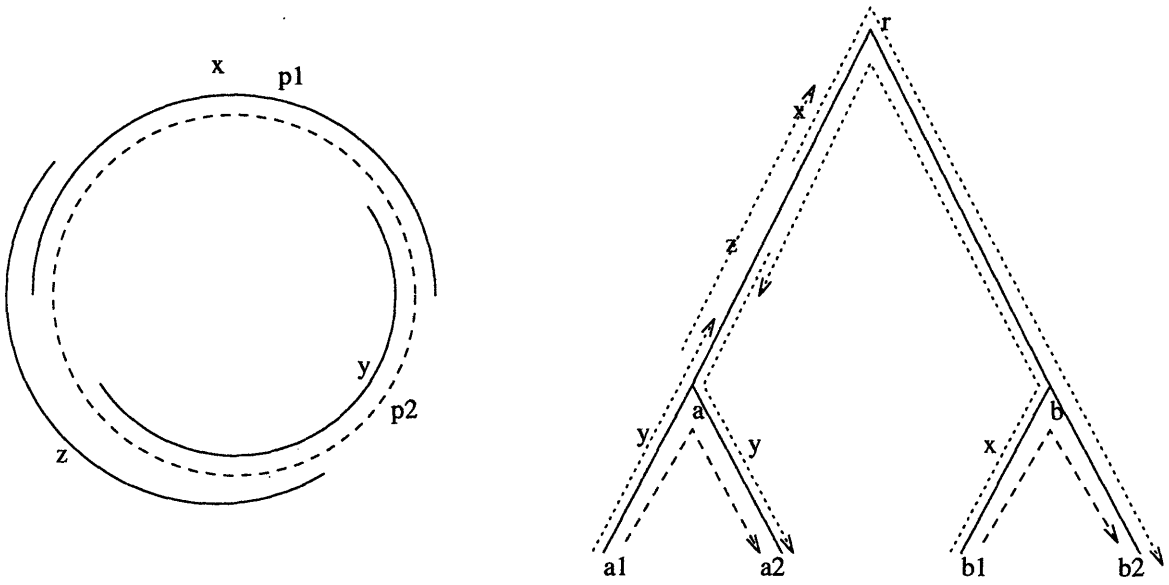

Figure 4-8: Reduction to a binary directed tree

a tree of cycles. Hence, either 2 or 3 colors suffice and the problem is in **P**. We can reduce the problem of edge-coloring cubic graph to show that the problem is **NP**-complete when $L_{\max} = 3$.

The problem of obtaining a better than $7L_{\max}/4$ algorithm for binary trees or better than $15L_{\max}/8$ for general trees ([MKR95]) is still tantalizingly open.

Chapter 5

Asymmetric Alternation in Interaction

5.1 Motivation

The framework of cooperating provers has received much attention. A natural question arises: what happens in a scenario in which provers *compete* with each other? In this chapter we explore a specific model of interactive proof systems – namely, alternating prover proof systems . The aim of this line of investigation was to obtain improved hardness of approximation results.

5.1.1 Summary of Results

We consider the scenario of *competing-prover proof systems*. In a *competing-prover proof system* two teams of competing provers ($P_i : i \in I$) and ($P_i : i \in \overline{I}$), (where $I \subseteq \{0, \ldots, k-1\}$, and $\overline{I} = \{0, \ldots, k-1\} \setminus I$), interact with a probabilistic polynomial-time verifier. The first team of provers tries to convince the verifier that ω is in L, for some prespecified-specified word ω and language L. The second team has the opposite objective, but the verifier does not know which team to trust. Before the protocol begins, the provers fix their strategies in the order specified by their subindices. These strategies are deterministic.[1] To model the situation of k competing provers, we propose and study the class **k-APP** of languages that have k-alternating-prover one-round proof systems.

Definition 12 *(k-Alternating-prover proof systems) A language L is said to have a* **k-APP** *system with parameters ($r(n), q(n), [Q_0, \ldots, Q_{k-1}]$), $Q_i \in \{\exists, \forall\}$, and error ($\epsilon_{acc}, \epsilon_{rej}$) if there is a probabilistic non-adaptive verifier V which runs in time polynomial in the size of the input, n, interacts with two teams of competing provers, and the following hold:*

[1] We will later see that it is important to distinguish between the cases where the provers can have randomized as opposed to deterministic strategies.

if $\omega \in L$, *then* $Q_0 P_0, \ldots, Q_{k-1} P_{k-1}$ *such that*

$$\Pr\left[\left(V \leftrightarrow P_0, \ldots, P_{k-1}\right)(\omega, r) \text{ accepts}\right] \geq 1 - \epsilon_{acc}$$

if $\omega \notin L$, *then* $\overline{Q}_0 P_0, \ldots, \overline{Q}_{k-1} P_{k-1}$ *such that*[2]

$$\Pr\left[\left(V \leftrightarrow P_0, \ldots, P_{k-1}\right)(\omega, r) \text{ accepts}\right] \leq \epsilon_{rej},$$

where the verifier is allowed one round of communication with each prover, the probabilities are taken over the random coin tosses of V, *and the quantifiers range over the set of deterministic provers' strategies. Furthermore,* V *uses* $O(r(n))$ *coin flips and the provers' responses are of size* $O(q(n))$.

As in multi-prover proof systems no prover has access to the communication generated by or directed to any other prover. The hierarchical order among the provers of an alternating-prover proof system is a key feature of the proof systems we consider.

We also consider systems of k-competing oracles.

Definition 13 *(k-Alternating-oracle proof systems) A language* L *is said to have a* **k-AOP** *system with parameters* $(r(n), q(n), [Q_0, \ldots, Q_{k-1}])$, $Q_i \in \{\exists, \forall\}$, *and error* $(\epsilon_{acc}, \epsilon_{rej})$ *if there is a probabilistic non-adaptive verifier* V *which runs in time polynomial in the size of the input,* n, *queries two teams of competing oracles, and the following hold:*

if $\omega \in L$ *then* $Q_0 O_0, \ldots, Q_{k-1} O_{k-1}$ *such that*

$$\Pr\left[\left(V \leftrightarrow O_0, \ldots, O_{k-1}\right)(\omega, r) \text{ accepts}\right] \geq 1 - \epsilon_{acc},$$

if $\omega \notin L$ *then* $\overline{Q}_0 O_0, \ldots, \overline{Q}_{k-1} O_{k-1}$ *such that*

$$\Pr\left[\left(V \leftrightarrow O_0, \ldots, O_{k-1}\right)(\omega, r) \text{ accepts}\right] \leq \epsilon_{rej},$$

where the probabilities are taken over the random coin tosses of V, *and the quantifiers range over the set of possible oracles. Furthermore,* V *uses* $O(r(n))$ *coin flips and queries* $O(q(n))$ *bits.*

There is a fundamental difference between alternating-prover proof systems and alternating-oracle proof systems. Provers may be asked only one question, whereas oracles may be asked multiple questions but their answers may not depend on the other questions. (This requirement was labeled in [FRS88] as the oracle requirement.)

[2] *For* $Q \in \{\exists, \forall\}$, $\overline{Q}$ *denotes* $\forall$ *if* $Q = \exists$, *and* $\exists$ *otherwise.*

We now establish some conventions that we will need for the rest of the chapter. Consider a k-APP or k-AOP systems with parameters $(r(n), q(n), [Q_0, \ldots, Q_{k-1}])$, $Q_i \in \{\exists, \forall\}$, and error $(\epsilon_{acc}, \epsilon_{rej})$. When we omit $r(n)$ and $q(n)$ they are assumed to be $\log n$ and 1 respectively; when the quantifiers do not appear, they are assumed to alternate beginning with $\exists$; when $\epsilon_{acc} = \epsilon_{rej} = \epsilon$, we say the system has error ϵ. We abbreviate the sequence of $2t$ alternating quantifiers beginning with $\exists$ by $(\exists, \forall)^t$. We define the class $\mathbf{k\text{-}APP}$ to be the set of languages that have a k-alternating-prover proof system with error $1/3$.[3] We follow the standard custom of denoting the length of the string ω by n. Finally, we denote provers by boldface letters, and their strategies by italic letters.

We characterize $\mathbf{NEXPTIME}$ and $\mathbf{NP}$ in terms of two-alternating-prover proof systems. We establish conditions under which alternating-prover proof systems are equivalent in power to alternating-oracle proof systems. We show that alternating-prover proof systems are weakened if we allow the provers to have access to secret sources of unbiased random bits. (In contrast the power of cooperating-prover proof systems, remains unchanged if the provers are allowed to use private coins.) We characterize the k-th level of the polynomial-time hierarchy in terms of k-alternating-oracle proof systems. This result is an extension of the $\mathbf{NP} = \mathbf{PCP}(\log n, 1)$ of [ALM+92b]. Finally, we give a precise characterization of each level of the polynomial-time hierarchy in terms of alternating-prover proof systems. The main result of this work is

Theorem 50 *For every constant ϵ, $0 < \epsilon < 1/2$, a language L is in $\Sigma_k^{\mathbf{P}}$, if and only if, it has a $(k+1)$-alternating-prover proof system with error ϵ.*

5.2 Previous Work

Interactive proof systems and the class $\mathbf{IP}$ were independently proposed by Babai [Bab85] and Goldwasser, Micali, and Rackoff [GMR85]. The major question left open by these papers was to determine the precise relation between $\mathbf{IP}$ and classical complexity classes. This was finally settled by Shamir *et al.* [LFKN90, Sha90] who showed that $\mathbf{IP} = \mathbf{PSPACE}$.

In order to allow for perfect zero-knowledge proofs without cryptographic assumptions, Ben-Or, Goldwasser, Kilian, and Wigderson [BOGKW88] introduced the concept of a multi-prover interactive proof. Here, the scenario is similar to the one of *interactive proof systems*, now, however, there are k powerful provers with whom the verifier interacts. Moreover, there is a crucial restriction on the provers, they cannot communicate with each other. The set of languages recognizable by multi-prover interactive proof systems is denoted $\mathbf{MIP}$. Clearly, $\mathbf{IP} \subseteq \mathbf{MIP}$. Babai, Fortnow, and Lund [BFL90] showed that $\mathbf{NEXPTIME} = \mathbf{MIP}$. This characterization was strengthened when [LS91, FL92] showed that $\mathbf{NEXPTIME}$ languages have two-prover one-round proof systems.

For a more thorough account of the work in interactive proof systems we refer the reader to the surveys of Johnson [Joh88, Joh92].

[3] Note that we are already employing the previously stated conventions.

Another line of research deriving from the $\mathbf{MIP} = \mathbf{NEXPTIME}$ result concerns the alternative characterization of **MIP** proposed by Fortnow, Rompel, and Siper [FRS88]. This alternative characterization is what more modern terminology refers to as *probabilistically checkable proofs* (PCPs). Here, the scenario is as before, but instead of the verifier having access to a prover he has random access to a proof tape (formally an oracle).

In essence, [FRS88] shows that the class of languages having probabilistically checkable proofs is precisely **MIP**, hence, $\mathbf{NEXPTIME} = \mathbf{PCP}(n^{O(1)}, n^{O(1)})$ [BFL90]. This result was followed by a sequence of highly technical developments that led to the celebrated $\mathbf{NP} = \mathbf{PCP}(\log n, 1)$ result of Arora, Lund, Motwani, Sudan, and Szegedy [ALM+92b]. Key among this sequence of developments were the papers of Babai, Fortnow, Levin, and Szegedy [BFLS91], Feige, Goldwasser, Lovász, Safra, and Szegedy [FGL+91b], and Arora and Safra [AS92b].

Informally, the $\mathbf{NP} = \mathbf{PCP}(\log n, 1)$ result says that languages in **NP** have proofs of membership that can be verified probabilistically by a small number of spot-checks. In this short discussion we cannot do true justice to the many researchers that contributed to the work leading to [ALM+92b] and the derivation of its many consequences. Thus, we refer the reader interested in a more accurate account of the contributions related to PCPs to the surveys of Babai [Bab92], Goldreich [Gol94], Johnson [Joh92], and to the thesis of Arora [Aro94] and Sudan [Sud92].

More recently, Feige and Kilian [FK94], building on top of the $\mathbf{NP} = \mathbf{PCP}(\log n, 1)$ result of [ALM+92b], show that **NP** is characterized by two-prover one-round proof systems in which the verifier has access to only $O(\log n)$ random bits, the provers' responses are constant size, and the probability that the verifier incorrectly accepts a string not in the language is at most ϵ, where ϵ can be an arbitrarily small positive constant.

In all the prover proof systems that we have discussed so far, *all* the provers are trying to convince the verifier of the veracity of a given statement. In other words, the provers always *cooperate* with each other.

Proof systems related to the one given in Definition 12, but where the provers do not have access to each others' strategies, have been studied in Feige, Shamir, and Tennenholtz [FST87]. Yet, the use, in a complexity theoretic setting, of the notion of competition among independent decision makers is older. Stockmeyer [Sto76a] used games between competing players to characterize languages in the polynomial-time hierarchy. Other uses of competing players to study complexity classes include Condon [Con88], Feigenbaum, Koller and Shor [FKS95], Reif [Rei79], Russell and Sundaram [RS95], and Peterson and Reif [PR79].

More recently, Condon, Feigenbaum, Lund, and Shor [CFLS93, CFLS94] characterized **PSPACE** by systems in which a verifier with $O(\log n)$ random bits can read only a constant number of bits of a polynomial-round debate between two players. These systems are the motivation four our definitions of k-competing oracles.

The rest of this chapter is organized as follows: Section 5.3 contains a characterization of **NP** in terms of two-alternating-prover proof systems; Section 5.4 studies the effect of allowing the provers of an alternating-prover proof system to have access to private coins; Section 5.5 contains the characterization of the levels

of the polynomial-time hierarchy in terms of alternating-oracle proof systems; Section 5.6 contains a review of some results of Feige and Kilian [FK94]; Section 5.7 contains a proof of Theorem 50; and, Section 5.8 contains some closing remarks.

5.3 Two-Alternating-Prover Proof Systems for NP

Suppose that a language L can be recognized by a two-cooperating-prover one-round proof system with verifier $\widehat{V}$ and provers $\widehat{\mathbf{P}}_0$ and $\widehat{\mathbf{P}}_1$ quantified by $([\exists, \exists])$. The verifier $\widehat{V}$ on random string r, asks prover $\widehat{\mathbf{P}}_0$ question $q^{(0)}(r)$, asks prover $\widehat{\mathbf{P}}_1$ question $q^{(1)}(r)$, and uses the answers to the questions to decide whether or not to accept. We will transform this system into a two-alternating-prover proof system with verifier V and provers $\mathbf{P}_0$ and $\mathbf{P}_1$ quantified by $([\exists, \forall])$. In this latter system, prover $\mathbf{P}_0$ claims that there exist provers $\widehat{\mathbf{P}}_0$ and $\widehat{\mathbf{P}}_1$ that would convince $\widehat{V}$ to accept. Prover $\mathbf{P}_1$ is trying to show that $\mathbf{P}_0$ is wrong. The verifier V will simulate the verifier $\widehat{V}$ from the original system to generate the questions $q^{(0)}(r)$ and $q^{(1)}(r)$ that the verifier $\widehat{V}$ would have asked the cooperating provers. To justify his claim, $\mathbf{P}_0$ will tell V what $\widehat{\mathbf{P}}_0$ or $\widehat{\mathbf{P}}_1$ would have said in answer to either question. To test $\mathbf{P}_0$'s claim, V will pick one of the two questions $q^{(0)}(r)$ and $q^{(1)}(r)$ at random and ask $\mathbf{P}_0$ to respond with what the corresponding prover would have said in answer to the question. Unfortunately, this does not enable V to get the answer to both questions. To solve this problem, we recall that $\mathbf{P}_1$ knows what $\mathbf{P}_0$ would answer to any question. Thus, V will send both questions to $\mathbf{P}_1$, and request that $\mathbf{P}_1$ respond by saying how $\mathbf{P}_0$ would have answered the questions.

If $\omega \in L$, then $\mathbf{P}_0$ is honest, and $\mathbf{P}_1$ has to lie about what $\mathbf{P}_0$ would say in order to get the verifier to reject. If $\mathbf{P}_1$ lies about what $\mathbf{P}_0$ said, he will be caught with probability at least $1/2$, because $\mathbf{P}_1$ does not know which question V sent to $\mathbf{P}_0$. Thus, the verifier will accept with probability at least $1/2$.

On the other hand, if $\omega \notin L$, then $\mathbf{P}_1$ will honestly answer by telling V what $\mathbf{P}_0$ would have answered to both questions. In this case, V will accept only if the provers that $\mathbf{P}_0$ represent would have caused $\widehat{V}$ to accept. Thus, we obtain a two-alternating-prover proof system with error $(1/2, \epsilon)$.

We now show how we can balance these error probabilities by a 'parallelization' of the above protocol with an unusual acceptance criteria.

Lemma 51 *A two-cooperating-prover one-round proof system with error $(0, \epsilon)$ can be simulated by a two-alternating-prover one-round proof system with error $(2^{-m}, m2^{m-1}\epsilon)$, an m-fold increase in the verifier's randomness and an $m2^m$-fold increase in the length of the answers returned by the provers.*

Proof: Let $\widehat{V}$ denote the verifier and $\widehat{\mathbf{P}}_0, \widehat{\mathbf{P}}_1$ the provers of the two-cooperating-prover proof system. Now, consider the two-alternating-prover proof system with verifier V and provers $\mathbf{P}_0, \mathbf{P}_1$ that execute the following protocol, henceforth referred to as the BASIC SIMULATION PROTOCOL:

THE QUERIES

— Verifier V generates m pairs of questions as the old verifier $\widehat{V}$ would have, i.e. V selects random strings $r_1, \ldots, r_m$ and generates $\left(\left(q^{(0)}(r_s), q^{(1)}(r_s)\right) : s = 1, \ldots, m \right)$.

— V then picks one question from each pair, i.e. randomly chooses $\vec{j} = (j_1, \ldots, j_m)$ in $\{0,1\}^m$, and sends to $\mathbf{P}_0$ the questions $\left(q^{(j_s)}(r_s) : s = 1, \ldots, m\right)$ and $\vec{j}$.

— V sends to $\mathbf{P}_1$ all the tuple of questions $\left(\left(q^{(0)}(r_s), q^{(1)}(r_s)\right) : s = 1, \ldots, m \right)$.

THE RESPONSES

— $\mathbf{P}_0$ is supposed to respond with $\left(\widehat{P}_{j_s}(q^{(j_s)}(r_s)) : s = 1, \ldots, m \right)$, which is how $\widehat{\mathbf{P}}_0$ and $\widehat{\mathbf{P}}_1$ would have answered the questions.

— $\mathbf{P}_1$ is supposed to respond with what $\mathbf{P}_0$ would have said to $\vec{i} = (i_1, \ldots, i_m)$ and $\left(q^{(i_s)}(r_s) : s = 1, \ldots, m\right)$, henceforth denoted $\left(a_s^{\vec{i}} : s = 1, \ldots, m\right)$, for every one of the 2^m possible assignments to $\vec{i}$.[4]

ACCEPTANCE CRITERIA

— The verifier V accepts if $\mathbf{P}_1$ did not correctly represent what $\mathbf{P}_0$ said on the tuple of questions that $\mathbf{P}_0$ was asked.

— Otherwise, V accepts if there exists at least one $s \in \{1, \ldots, m\}$ and index vector $\vec{i}$ such that $(\vec{i})_s \neq (\vec{j})_s$ and $\widehat{V}$ on random string r_s would accept if given answers $a_s^{\vec{i}}$ from $\widehat{\mathbf{P}}_{(\vec{i})_s}$ and $a_s^{\vec{j}}$ from $\widehat{\mathbf{P}}_{(\vec{j})_s}$.

Assume that the two-cooperating-prover proof system accepts. Then, $\mathbf{P}_0$ is honest. Hence, $\mathbf{P}_0$ truthfully answers each question he is asked as $\widehat{\mathbf{P}}_0$ would have. We will see that this forces $\mathbf{P}_1$ to lie in response to all but the set of indices $\vec{j}$. Indeed, suppose there is an m-tuple of questions indexed by $\vec{i}$, $\vec{i} \neq \vec{j}$, for which $\mathbf{P}_1$ honestly represents what $\mathbf{P}_0$ would say. Then there must be an s such that $(\vec{i})_s \neq (\vec{j})_s$. If $\mathbf{P}_1$ honestly represented $\mathbf{P}_0$'s answers on tuple $\vec{i}$, the verifier V would accept, since $\widehat{V}$ on random string r_s would accept if given answers $a_s^{\vec{i}}$ from $\widehat{\mathbf{P}}_{(\vec{i})_s}$ and $a_s^{\vec{j}}$ from $\widehat{\mathbf{P}}_{(\vec{j})_s}$. Since $\mathbf{P}_1$ does not see the randomly chosen $\vec{j}$, he decides to honestly answer the questions sent to prover $\mathbf{P}_0$ with probability at least $1 - 2^{-m}$. Hence, the probability that V accepts is at most 2^{-m}.

On the other hand, assume that the two-cooperating-prover proof system rejects. Then the probability that V accepts is at most ϵ for each s, and $\vec{i}$ such that $(\vec{i})_s \neq (\vec{j})_s$, for a total error of $m2^{m-1}\epsilon$. □

Lemma 51 yields an exponential-in-m blowup in the size of the provers' responses. This may seem to be a fatal drawback. But, we will apply this lemma in situations where m is not too large and ϵ is very small.

[4] We have to allow $\mathbf{P}_1$ to answer this way. This is because a cheating prover $\mathbf{P}_0$ may vary its strategy according to the value of $\vec{j}$ that it receives, but which $\mathbf{P}_1$ does not see.

We will combine the preceding lemma with the theorem of Feige and Kilian [FK94], which we state below in our terminology:

Theorem 52 *(Feige-Kilian [FK94]) For any constant ϵ, $0 < \epsilon < 1/2$, a language L is in* **NP**, *if and only if, L has a two-alternating-prover proof system with parameters $([\exists, \exists])$ and error $(0, \epsilon)$.*

Remark 1 *The proof of Theorem 52 given in [FK94] is based on a proof of a weak version of the parallel repetition theorem. Shortly after [FK94] appeared, Raz [Raz95] succeeded in proving the parallel repetition theorem. Thus, the above claim can now be proved by combining the* $\mathbf{NP} = \mathbf{PCP}(\log n, 1)$ *result of [ALM+92b] with the techniques from [FRS88] for simulating an oracle by a pair of provers, and then reducing the error of the proof system thus derived by parallel repetition. This yields an alternative proof of Theorem 52 where the hidden constants are better (smaller).*

Corollary 53 *For any constant ϵ, $0 < \epsilon < 1/2$. A language L is in* **NP**, *if and only if, L has a two-alternating-prover proof system with error ϵ.*

Proof: The forward direction follows immediately from Lemma 51 and Theorem 52 by choosing an appropriate integer m and a positive constant ϵ such that $2^{-m} \leq 1/3$ and $m2^{m-1}\epsilon \leq 1/3$. For the reverse direction, assume L has a two-alternating-prover proof system with error ϵ. Observe that the strategy of the first prover is of length polynomial in the size of the input ω. Thus, given ω we can non-deterministically guess the optimal strategy of the first prover. We then compute the optimal strategy for the second prover. This can be done in polynomial time. It is now possible to compute the acceptance probability efficiently by simulating the verifier's behavior on each of the $2^{O(\log(|\omega|))}$ random strings. The probability thus computed is at least $1 - \epsilon$, if and only if, $\omega \in L$. □

Corollary 54 *A language L is in* **NEXPTIME**, *if and only if, L has a two-alternating-prover proof system with parameters $(poly(n), poly(n), [\exists, \forall])$ and error $(1/poly(n), 1/exp(n))$.*[5]

Proof: We first prove the forward direction. Observe that the **NEXPTIME** characterization of [FL92] in terms of multi-prover interactive proof systems, can be rephrased (in our terminology) as: a language is in **NEXPTIME**, if and only if, it can be recognized by a two-alternating-prover proof system with parameters $(poly(n), poly(n), [\exists, \exists])$ and error $(0, 1/exp(n))$. Hence the corollary follows again from Lemma 51 and by choosing $2^{-m} \leq 1/poly(n)$ and $m2^{m-1}\epsilon \leq 1/exp(n)$. The reverse implication follows by properly scaling the proof of the reverse direction of Corollary 53. □

5.3.1 The Robust Simulation Protocol

In Section 5.7, we will need a stronger version of Lemma 51. This section describes extensions to the BASIC SIMULATION PROTOCOL that yield such strengthening.

[5] *The terms $poly(n)$ and $exp(n)$ refer to $O(n^c)$ and $O\left(2^{n^c}\right)$ respectively, where c is some positive constant.*

In the previous section we saw how two-cooperating-prover proof systems with one sided error could be simulated by two-alternating-prover proof systems. We now study the possibility of carrying out a similar simulation of k-cooperating-prover proof systems, even in the case that the error is two sided. In addition, the simulation is carried out in a more demanding scenario.

As in the BASIC SIMULATION PROTOCOL we consider an alternating-prover proof system verifier V who *concurrently* simulates m rounds of the original cooperating-prover proof system verifier $\widehat{V}$. We refer to each one of these simultaneously performed rounds as a *run*. For each one of these m runs V assigns a label of accept/reject based on computations and simulations of $\widehat{V}$. We model the situation where some of the provers' responses are maliciously corrupted. We do this by allowing up to a ξ-fraction of the labels to be arbitrarily flipped. We refer to these flipped labels as *wild-cards*. The verifier V now decides to accept or reject based on the total number of accept labels. We refer to these systems as *alternating-prover proof systems with a ξ-fraction of wild-cards.*

Lemma 55 *Let $\beta > 1$ be such that $e^{\beta}\beta^{-\beta} = 1/e$. A k-cooperating-prover one-round proof system with error $(\epsilon_{acc}, \epsilon_{rej})$ can be simulated by a two-alternating-prover one-round proof system with a ξ-fraction of wild-cards, with error $\left(e^{-m\epsilon_{acc}} + (1-1/k)^{(1-2\xi-\beta\epsilon_{acc})m}, mk^{(m-1)(k-1)}\epsilon_{rej}\right)$, an m-fold increase in the verifier's randomness and an mk^m-fold increase in the length of the answers returned by the provers.*

Proof: Let $\widehat{V}$ denote the k-cooperating-prover proof systems' verifier, and $\widehat{\mathbf{P}}_0, \ldots, \widehat{\mathbf{P}}_{k-1}$ its provers. Moreover, denote by V the verifier and by $\mathbf{P}_0, \mathbf{P}_1$ the provers of the two-alternating-prover proof system that executes the following protocol, henceforth referred to as the ROBUST SIMULATION PROTOCOL WITH A ξ-FRACTION OF WILD-CARDS:

THE QUERIES

— V generates m k-tuples of questions as $\widehat{V}$ would have, i.e. V selects random strings $r_1, \ldots, r_m$ and generates $\left(\left(q^{(0)}(r_s), \ldots, q^{(k-1)}(r_s)\right) : s = 1, \ldots, m \right)$.

— V then chooses one question from each k-tuple, i.e. randomly selects $\vec{j} = (j_1, \ldots, j_m)$ in $\{0, \ldots, k-1\}^m$, and sends to $\mathbf{P}_0$ both $\vec{j}$ and the questions $\left(q^{(j_s)}(r_s) : s = 1, \ldots, m\right)$.

— V sends to $\mathbf{P}_1$ the tuple of questions $\left(\left(q^{(0)}(r_s), \ldots, q^{(k-1)}(r_s)\right) : s = 1, \ldots, m \right)$.

THE RESPONSES

— $\mathbf{P}_0$ is supposed to respond with $\left(\widehat{P}_{j_s}(q^{(j_s)}(r_s)) : s = 1, \ldots, m \right)$, which is how $\widehat{\mathbf{P}}_0, \ldots, \widehat{\mathbf{P}}_{k-1}$ would have answered the questions.

— Prover $\mathbf{P}_1$ is supposed to respond with what prover $\mathbf{P}_0$ would have answered to $\left(q^{(i_s)}(r_s) : s = 1, \ldots, m\right)$ and $\vec{i} = (i_1, \ldots, i_m)$, say $\left(a_s^{\vec{i}} : s = 1, \ldots, m\right)$, for every one of the k^m possible assignments to $\vec{i}$.[6]

[6] We have to allow $\mathbf{P}_1$ to answer this way. This is because a cheating prover $\mathbf{P}_0$ may vary its strategy according to the value of $\vec{j}$ that it receives, but which $\mathbf{P}_1$ does not see.

ACCEPTANCE CRITERIA

— V accepts if $\mathbf{P}_1$ did not correctly represent what $\mathbf{P}_0$ said on the tuple of questions that $\mathbf{P}_0$ was asked.

— Otherwise, the verifier V labels a run s accept if there are index vectors $\vec{i}_1, \ldots, \vec{i}_{k-1}$ such that $(\vec{j})_s$ and $(\vec{i}_1)_s, \ldots, (\vec{i}_{k-1})_s$ are all distinct, and $\widehat{V}$ on random string r_s accepts given answers $a_s^{\vec{j}}$ from $\widehat{\mathbf{P}}_{(\vec{j})_s}$ and $a_s^{\vec{i}_1}, \ldots, a_s^{\vec{i}_{k-1}}$ from $\widehat{\mathbf{P}}_{(\vec{i}_1)_s}, \ldots, \widehat{\mathbf{P}}_{(\vec{i}_{k-1})_s}$ respectively. Up to a ξ-fraction of these labels could be wild-cards. The verifier V accepts if there are at least $\xi m + 1$ runs labeled accept.

Assume the cooperating-prover proof system accepts. Then $\mathbf{P}_0$ honestly answers the question he is asked in each run. Let p be the fraction of the random strings which V selects that would cause $\widehat{V}$ to reject on input ω. If $\mathbf{P}_1$ answers honestly in more than a $(1-1/k)^{(1-p-2\xi)m}$ fraction of the $\vec{i}$'s, then V labels at least $\xi m+1$ of the runs with accept. A multiplicative Chernoff bound (see e.g. [AS92a, Theorem A.12]) shows that the probability that p is greater than $\beta\epsilon_{acc}$ is at most $(e^{\beta}\beta^{-\beta})^{m\epsilon_{acc}}$, which by our choice of β equals $e^{-m\epsilon_{acc}}$. Hence, the verifier V rejects with probability at most $e^{-m\epsilon_{acc}} + (1-1/k)^{(1-2\xi-\beta\epsilon_{acc})m}$.

Assume now that the cooperating-prover proof system rejects. The verifier V accepts if at least one more than ξm of the runs get labeled accept. A fraction ξ of the runs could be wild-cards. The probability that there exists one more run labeled accept is at most ϵ_{rej} for each s, and $\vec{i}_1, \ldots, \vec{i}_{k-1}$ such that $(\vec{j})_s$ and $(\vec{i}_1)_s, \ldots, (\vec{i}_{k-1})_s$ are all distinct, for a total error of $mk^{(m-1)(k-1)}\epsilon_{rej}$. □

5.4 Two-Alternating Randomized Prover Proof Systems

The power of *cooperating*-prover proof systems, is unchanged if the provers are allowed to choose randomized strategies. In this section we show that the situation is different for competing-prover proof systems.

A *two-alternating randomized-prover proof system*, denoted **2-ARP**, is one where the provers have access to private coins. The same conventions that apply to two-alternating-prover proof system will be adopted for two-alternating randomized-prover proof systems.

The main result of this section shows that unless $\mathbf{P} = \mathbf{NP}$, a two-alternating randomized-prover proof system with constant error is strictly weaker than a two-alternating-prover proof system with constant error. Intuitively, in a two-alternating randomized-prover proof system the first prover can use his private coins to 'hide' his strategy from the second prover. The second prover cannot then figure out the exact intentions of the first prover. He cannot devise as good a strategy against this type of adversary as opposed to when the first prover acts according to a deterministic strategy. Thus, the second prover is of less use to the verifier, and consequently the whole proof system is weakened.

Lemma 56 *For every constant ϵ, $0 < \epsilon < 1/2$, a language L is in $\mathbf{P}$, if and only if, L has a two-alternating randomized-prover proof system with parameters $(O(\log(n)), O(\log(n)), [\exists, \forall])$ and error ϵ.*

Proof: The forward direction is trivial, since the verifier alone, without the help of the provers, can decide languages in $\mathbf{P}$.

We now prove the converse. Let

— $D_i(q,a)$ be the probability that prover $\mathbf{P}_i$ responds to q with answer a,

— Q_i be the set of possible questions to prover $\mathbf{P}_i$, and A_i be the set of possible responses of prover $\mathbf{P}_i$,

— $D_i = (\, D_i(q,a) : q \in Q_i, a \in A_i \,)$,

— R be the set of random strings that the verifier may generate on input ω,

— π_r be the probability that the verifier generates random string r,

— $q^{(i)}(r)$ be the question that the verifier sends to prover $\mathbf{P}_i$ on random string r,

— $V(r, a_0, a_1)$ be 1 if the verifier V accepts on input ω, random string r and given responses a_0 from $\mathbf{P}_0$ and a_1 from $\mathbf{P}_1$, Otherwise, let $V(r, a_0, a_1)$ be 0.

Feige and Lovász [FL92] have shown that the probability of acceptance of a two-cooperating-prover proof system can be computed by solving a quadratic optimization problem in max form. Applying their technique we get that the probability that V accepts on input ω is

$$\begin{aligned} &\max_{D_0} \; C\,(D_0) \\ &\text{subject to,} \quad \forall q \in Q_0, \; \sum_{a \in A_0} D_0(q,a) = 1 \\ &\qquad\qquad \forall q \in Q_0, \forall a \in A_0, \;\; D_0(q,a) \geq 0\,, \end{aligned}$$

where

$$\begin{aligned} C(D_0) \stackrel{\text{def}}{=} \; &\min_{D_1} \sum_{r \in R} \sum_{a_0 \in A_0, a_1 \in A_1} \pi_r V(r, a_0, a_1)\, D_0(q^{(0)}(r), a_0)\, D_1(q^{(1)}(r), a_1) \\ &\text{subject to,} \quad \forall q \in Q_1, \; \sum_{a \in A_1} D_0(q,a) = 1 \\ &\qquad\qquad \forall q \in Q_1, \forall a \in A_1, \;\; D_1(q,a) \geq 0\,. \end{aligned}$$

By the duality theorem of linear programming (see e.g. [Sch86, Ch. 7.§4]), $C(D_0)$ can be expressed as the optimum of a linear program in max form. Thus, to compute the probability that V accepts ω it is enough to solve a linear program of $poly(|\omega|, |A_0|, |A_1|, |R|)$ size. Since in our case $|A_0|, |A_1|, |R| = 2^{O(\log |\omega|)}$, and linear programming is polynomial-time solvable, the lemma follows. □

An analogous result for $\mathbf{EXP}$, for the only if part, was independently obtained in [FKS95].

Corollary 57 *For every constant ϵ, $0 < \epsilon < 1/2$, a language L is in $\mathbf{P}$, if and only if, L has a two-alternating randomized-prover proof system with error ϵ.*

But, in Corollary 53 we showed that languages in $\mathbf{NP}$ have two-alternating-prover proof systems with constant error. Hence, $\mathbf{2\text{-}APP} \neq \mathbf{2\text{-}ARP}$, unless $\mathbf{P} = \mathbf{NP}$.

5.5 Alternating-Oracle Proof Systems for Σ_k^P

Feige and Kilian's proof of Theorem 52 uses an amplification of the main result of [AS92b, ALM+92b], which states that $\mathbf{NP} = \mathbf{PCP}(\log n, 1)$. In our terminology, this says that **NP** languages have one-alternating-oracle proof systems with error $(0, \epsilon)$, where ϵ can be an arbitrarily small positive constant. In order to extend our results beyond **NP**, we will need analogous tools that we can apply to languages in Σ_k^P. Thus, we begin by showing that languages in Σ_k^P have k-alternating-oracle proof systems.

We consider from now on only the case in which k is odd. Our results have analogous statements for even k.

The next theorem is implicit in [CFLS94].

Theorem 58 *(k odd) For any constant $\epsilon > 0$. Every language L in Σ_k^P has a k-alternating-oracle proof system with error $(0, \epsilon)$.*

In the proof of this theorem, we will make use of the fact that there are *encodings*, $E(\cdot)$, that associate to a string x, a string $E(x)$ such that the following holds:

— the length of $E(x)$ is linear (for our purposes, polynomial would suffice) in the length of x.

— there is a polynomial-time algorithm that on input x returns $E(x)$.

— there is a constant ϵ_E and a polynomial-time algorithm D_E such that if y and $E(x)$ differ in at most an ϵ_E fraction of their bits, then $D_E(y) = x$ (in which case we say that x and y are *closer than* ϵ_E). Otherwise, $D_E(y)$ outputs FAILURE (in which case, we say that y is *farther than* ϵ_E from any codeword).

Encodings with this property exist (e.g. Justesen encodings [MS77, Ch. 10.§11]).

Proof of Theorem 58: Let L be a language in Σ_k^P. That is, there exists a polynomial-time Turing machine M such that $\omega \in L$ if and only if $\exists X_1, \forall X_2, \ldots, \exists X_k \; M(\omega, X_1, \ldots, X_k)$ accepts [CKS81]. As in [CFLS94], we will view the acceptance condition as a game between an $\exists$ player and a $\forall$ player who take turns writing down polynomial-length strings X_i, with the $\exists$ player writing on the odd rounds.

In our k-alternating-oracle proof system, the player who writes in round i purports to write down an encoding $E(X_i)$ of X_i. In addition, in the k-th round, the $\exists$ player is to write down encodings of everything the $\forall$ player said and a **PCP** proof that $M(\omega, X_1, \ldots, X_k)$ accepts. If each player actually wrote down codewords, then, using the techniques from [ALM+92b], the verifier would read a constant number of random bits from each oracle and a constant number of bits to check the **PCP** proof and accept if M on input $(\omega, X_1, \ldots, X_k)$ would have accepted, or reject with high probability if M would have rejected.

In reality, one of the players will be trying to cheat and will have little incentive to write codewords. Let Y_i denote the oracle that is written in the i-th round. If a player writes an oracle Y_i that is within ϵ_E of a codeword, then the other player will proceed as if Y_i was that codeword. On the other hand, if a player writes an oracle Y_i that is farther than ϵ_E from a codeword, then with high probability the verifier will detect that player's perfidy.

In the last round, the $\exists$ player writes down strings Z_i which he claims are encodings of Y_i, for each even i. In addition, the $\exists$ player will, for each bit of each oracle Y_i, provide a **PCP** proof that, when decoded, Z_i agrees with Y_i on that bit. For each even i, the verifier will test these **PCP** proofs to check that $D_E(Z_i)$ agrees with Y_i on some constant number of randomly chosen bits. If any of these tests fails, then the verifier will know that the $\exists$ player has cheated and will reject accordingly. If the Z_i's pass all the tests, then the verifier will be confident that $D_E(Z_i)$ is close to Y_i, for all even i. The verifier then accepts if the last player's proof indicates either that,

— $D_E(Y_i) =$ FAILURE for some even i, or

— $M(\omega, X_1, D_E(Y_2), \ldots, D_E(Y_{k-1}), X_k)$ accepts (note that the **PCP** proof refers to the X_i's for odd i through their encoding as Y_i, and to the Y_i's for i even through their encoding as Z_i).

To see why this protocol works, assume that $\omega \in L$. In this case, the $\exists$ player will always write codewords. Moreover, regardless of what oracle Y_i the $\forall$ player writes in turn i, the exist player will write $Z_i = E(Y_i)$. Thus, the Z_i's will always pass the consistency tests. If one of the $\forall$ player's oracles is farther than ϵ_E from a codeword, then the last player will include this fact in his proof, and the verifier will reject. On the other hand, if for each even i, Y_i is close to some codeword X_i, then the application of D_E to Y_i will result in Y_i being treated as the encoding of X_i in the $\exists$ player's **PCP** proof.

If $\omega \notin L$, then the $\exists$ player will have to cheat in order to win. If the $\exists$ player writes a message Y_i that is not close to a unique codeword, then this will be detected with high probability when the verifier checks the validity of the **PCP** proof supplied in the last round. If one of the Z_i's misrepresents the oracle, $Y_i = E(X_i)$, of a $\forall$ player, then either $D_E(Z_i)$ and Y_i will have to differ in at least an ϵ_E fraction of their bits, or the $\exists$ player will have to falsify the certification of the computation of D_E on Z_i so that it does not decode to X_i. In either case, the $\exists$ player will be caught with high probability. □

We note that Theorem 58 exactly characterizes Σ_k^P because an alternating-Turing machine with k alternations can guess the k oracles and then compute the acceptance probability of the k-alternating-oracle proof system.

Using Theorem 58 and the standard technique of [FRS88] for simulating an oracle by a pair of provers, we can transform a k-alternating-oracle proof system into a $(k+1)$-alternating-prover proof system.

Corollary 59 *(k odd) Let $k = 2t+1$. Then, $L \in \Sigma_k^P$, if and only if, L has a $(k+1)$-alternating-prover proof system with parameters $([(\exists, \forall)^t, \exists, \exists])$ and error $\left(0, 1 - \frac{1}{N}\right)$, where N is a large constant depending on k.*

In order to prove Theorem 50, we have to reduce the error in Corollary 59 and show how to change the quantifiers from $([(\exists, \forall)^t, \exists, \exists])$ to $([(\exists, \forall)^t, \exists, \forall])$. In other words, we want an analogue of the parallel repetition theorem and of Lemma 51 which apply to alternating-prover proof systems. In the next two sections we describe how these goals can be achieved.

5.6 Feige and Kilian lemmas

In Section 5.7, we will prove an analogue of the weak version of the parallel repetition theorem of Feige and Kilian [FK94] that applies to k-alternating-prover proof systems. The techniques used in [FK94] provide a deep insight into how a few random variables influence the value of a multi-variate function. In order to prove our analogue of their theorem, we will need a better understanding of some of their results, which we will summarize in this section.

Consider a prover $\mathbf{P}$ to which we send m randomly chosen questions $(q(r_1), \ldots, q(r_m))$ and to which $\mathbf{P}$ answers $(a_1, \ldots, a_m)$ according to a fixed strategy $f = (f_1, \ldots, f_m)$, a function from m-tuples (the m-tuple of questions that $\mathbf{P}$ receives), to m-tuples (the m-tuple of answers that $\mathbf{P}$ responds with). We would like $\mathbf{P}$ to use a global strategy, f, in which each f_i is only a function of the question on round i, i.e. a function of $q(r_i)$. We say that such a prover behaves *functionally*. We now discuss the consequences of a prover's failure to behave in this way.

Let $\widetilde{I}$ be a subset of $\{1, \ldots, m\}$ and $\widetilde{Q} = (q(r_i) : i \in \widetilde{I})$ be the questions indexed by $\widetilde{I}$. Now, suppose we knew both $\widetilde{I}$ and $\widetilde{Q}$, then, if the prover behaves functionally we can determine $(a_i : i \in \widetilde{I})$. But what if the prover does not behave functionally, but 'almost functionally'? To answer this we need to formalize what 'almost functionally' means: Say that an *f-challenge*[7] $(\widetilde{I}, \widetilde{Q})$ is *live* if there is an $\widetilde{A} = (a_i : i \in \widetilde{I})$ such that

$$\Pr\left[f_i(q(r_1), \ldots, q(r_m)) = a_i, \text{ for all } i \in \widetilde{I}\right] \geq \zeta,$$

where the probability is taken over the choices of $(r_i : i \notin \widetilde{I})$, and ζ is a parameter to be fixed later. If the above inequality holds, say that $\widetilde{A}$ is a *live answer* for the challenge $(\widetilde{I}, \widetilde{Q})$. Intuitively, if we know the questions $\widetilde{Q}$ that $\mathbf{P}$ receives on rounds $\widetilde{I}$ and that $\widetilde{A}$ is not a live answer for the challenge $(\widetilde{I}, \widetilde{Q})$, we should not be willing to bet that $\mathbf{P}$ will answer $\widetilde{A}$ on rounds $\widetilde{I}$, because, $\widetilde{A}$ has too small a chance of occurring. If $\mathbf{P}$ acts functionally in each round, then every challenge $(\widetilde{I}, \widetilde{Q})$ is live, since the questions that $\mathbf{P}$ is sent on a round completely determines his answer on that round.

For any choice of parameters $\epsilon, \zeta, \gamma, n, m$, and M such that $8\epsilon < \zeta^5$, $\zeta^{-5} < \gamma n$, $0 < \gamma < 1$, $M = m - n$, $\epsilon \geq \max\left\{\frac{256 \ln M}{M}, \left(\frac{8 \ln M}{M}\right)^{1/3}\right\}$ (in particular note that $m \gg n$), the following lemmas are implicit in the work of Feige and Kilian:

Lemma 60 *(Feige-Kilian) Let $j \in \{0, \ldots, n\}$. If $\widetilde{A} = (a_i : i \in \widetilde{I})$ is not a live answer for the challenge $(\widetilde{I}, \widetilde{Q})$, where $\widetilde{Q} = (q(r_i) : i \in \widetilde{I})$ and $\widetilde{I} = \{i_1, \ldots, i_j\}$, then, with probability at least $1 - n\epsilon$ (over the choices of $I \setminus \widetilde{I} = \{i_{j+1}, \ldots, i_n\}$ and $\{r_i \mid i \in I \setminus \widetilde{I}\}$, where $I = \{i_1, \ldots, i_n\} \subset \{1, \ldots, m\}$) it holds that*

$$\Pr\left[f_i(q(r_1), \ldots, q(r_m)) = a_i, \textit{ for all } i \in \widetilde{I}\right] < \zeta + (n - j)\epsilon,$$

[7] We omit f whenever it is clear from context.

where the probability is taken over the choices of $(r_i : i \notin I)$.

Intuitively, the preceding lemma says that if $\widetilde{A}$ is not a live answer set for the challenge $(\widetilde{I}, \widetilde{Q})$, and in addition to knowing the questions that $\mathbf{P}$ receives in rounds $\widetilde{I}$ we know the questions sent to $\mathbf{P}$ in rounds $I \supseteq \widetilde{I}$, we usually should still not be willing to bet that $\mathbf{P}$ will answer $\widetilde{A}$ in rounds $\widetilde{I}$. Note however, that for this to be true we assume that $|I|$ is a small fraction of all the rounds.

Lemma 61 *(Feige-Kilian) There is a good j for $f = (f_1, \ldots, f_m)$, $j < \gamma n$, such that if more than a ζ fraction of the challenges $(\widetilde{I}, \widetilde{Q})$ (over the choices of $\widetilde{I} = \{i_1, \ldots, i_j\}$ and $(r_i : i \in \widetilde{I})$) are live, then for a $(1-\zeta)$ fraction of the live challenges $(\widetilde{I}, \widetilde{Q})$, for every $i \notin \widetilde{I}$, and for each live answer set $\widetilde{A} = (a_i : i \in \widetilde{I})$ for the challenge $(\widetilde{I}, \widetilde{Q})$, it holds that there is a function $F_{f,i,\widetilde{Q},\widetilde{A}}$ such that*

$$\Pr\left[\,|\{\, i \in I \setminus \widetilde{I} \mid f_i(q(r_1), \ldots, q(r_m)) \neq F_{f,i,\widetilde{Q},\widetilde{A}}(q(r_i))\,\}| \leq \frac{2\zeta}{\delta}|I \setminus \widetilde{I}|\right] \;\geq\; 1-\delta\,,$$

where the probability is taken over the choices of $(r_i : i \notin \widetilde{I})$ and $I \setminus \widetilde{I} = \{i_{j+1}, \ldots, i_n\}$.

Informally stated, the previous lemma says that to every strategy of prover $\mathbf{P}$ we can associate a nonnegative integer j such that if a non-negligible fraction of the challenges $(\widetilde{I}, \widetilde{Q})$ are live, where $|\widetilde{I}| = j$, and if $\mathbf{P}$ answers the challenge $(\widetilde{I}, \widetilde{Q})$ with live answer $\widetilde{A}$, then, $\mathbf{P}$ behaves 'almost functionally' in a significant fraction of the rounds $I \setminus \widetilde{I}$, for most choices of the live challenges $(\widetilde{I}, \widetilde{Q})$, and of the sets $I \setminus \widetilde{I}$.

5.7 Alternating-Prover Proof Systems for $\Sigma_{\mathbf{k}}^{\mathbf{P}}$

In Section 5.5 we characterized $\Sigma_{\mathbf{k}}^{\mathbf{P}}$ in terms of alternating-prover proof systems with one sided but large rejection error and where the quantifiers associated to the provers, *except* the last one, alternated beginning with the existential quantifier. The goal of this section is to provide alternating-prover proof systems for $\Sigma_{\mathbf{k}}^{\mathbf{P}}$ languages which in addition achieve:

— Low error rates, and

— The quantifiers associated to *all* the provers alternate.

To achieve these goals we follow an approach similar to the one taken in the obtaining of two-alternating-prover proof systems for $\mathbf{NP}$ (see Section 5.3). We again restrict our discussion to the case $k = 2t+1$, i.e. k is odd. Our results have analogue statements for even k.

First, we prove that languages that have a $(k+1)$-alternating-prover proof system with parameters $([(\exists, \forall)^t, \exists, \exists])$ and one sided but large rejection error can be recognized by a $(k+3)$-alternating-prover proof system with parameters $([(\exists, \forall)^t, \forall, \exists, \exists, \exists])$ and two sided but arbitrarily small constant error.[8] Notice that we achieve

[8] In fact, a $(\mathbf{k+2})$-$\mathbf{APP}$ system with parameters $((\exists, \forall)^t, \forall, \exists, \exists))$ and similar error rates suffices, but proving this would unnecessarily complicate our exposition.

this without increasing the number of alternations of the quantifiers associated to the provers. This step can be viewed as an analogue of the weak version of the parallel repetition theorem of Feige and Kilian [FK94] that applies to competing-prover proof systems.

We then describe how to convert the proof systems obtained in the first step into $(k+1)$-alternating-prover proof systems in which all the quantifiers associated to the provers alternate, and where the error is small. This latter step is an analogue of Lemma 55.

5.7.1 Achieving Low Error Rates

We now informally describe a simulation protocol that achieves the first goal of this section. We consider a $(k+1)$-alternating-prover proof system with parameters $([(\exists, \forall)^t, \exists, \exists])$ and one sided but large rejection error, with verifier $\widehat{V}$ and provers $\widehat{\mathbf{P}}_0, \ldots, \widehat{\mathbf{P}}_k$. We simulate this proof system with a $(k+3)$-alternating-prover proof system where the error is low. The verifier of this latter proof system is denoted by V, and its provers by $\mathbf{P}_0, \ldots, \mathbf{P}_k$, and $\mathbf{P}_\exists$, $\mathbf{P}_\forall$. Prover $\mathbf{P}_i$ is quantified as $\widehat{\mathbf{P}}_i$ is. Moreover, $\mathbf{P}_\exists$ and $\mathbf{P}_\forall$ are quantified by $\exists$ and $\forall$ respectively. The ordering of the provers is such that $\mathbf{P}_\exists$ and $\mathbf{P}_\forall$ follow immediately after the last prover among $\mathbf{P}_0, \ldots, \mathbf{P}_k$ which is quantified by $\exists$ and $\forall$ respectively. Below we illustrate the relation among the old and new provers.

$$\begin{array}{ccccccc}
\overbrace{\phantom{\widehat{\mathbf{P}}_0}}^{\exists} & & & \overbrace{\phantom{\widehat{\mathbf{P}}_{k-3}}}^{\exists} & \overbrace{\phantom{\widehat{\mathbf{P}}_{k-2}}}^{\forall} & \overbrace{\phantom{\widehat{\mathbf{P}}_{k-1}\widehat{\mathbf{P}}_k}}^{\exists} & \\
\underbrace{\widehat{\mathbf{P}}_0} & \cdots & \underbrace{\widehat{\mathbf{P}}_{k-3}} & \underbrace{\widehat{\mathbf{P}}_{k-2}} & \underbrace{\widehat{\mathbf{P}}_{k-1}\widehat{\mathbf{P}}_k} & & \\
\downarrow & \cdots & \downarrow & \downarrow & \downarrow & & \\
\mathbf{P}_0 & \cdots & \mathbf{P}_{k-3} & \mathbf{P}_{k-2}\mathbf{P}_\forall & \mathbf{P}_{k-1}\mathbf{P}_k\mathbf{P}_\exists & &
\end{array}$$

The verifier V concurrently simulates many rounds of $\widehat{V}$'s protocol. We refer to each of these rounds as a *run*. The verifier V expects provers to behave functionally, i.e. to answer each of their questions according to a function that does not depend on the other questions they receive. Cheating provers have no incentive to comply with V's expectations. In order to penalize them for not doing so V implements the CONSISTENCY TEST. This test requires two (one for each competing team of provers) additional provers ($\mathbf{P}_\exists$ and $\mathbf{P}_\forall$). If a team of provers fails this test, their claim is rejected. Honest provers behave functionally. Thus, they always pass the CONSISTENCY TEST.

Cheating provers pass the CONSISTENCY TEST with a significant probability, only if they behave 'almost functionally'. But, this is not a guarantee that honest provers can determine these functional strategies. It is important to address this issue when designing protocols for alternating-prover proof systems as opposed to cooperating-prover proof systems.

The protocol implemented by V allows honest provers to make a set of 'educated' guesses regarding the strategies that cheating provers use. In fact, V allows $\mathbf{P}_1$ to make constantly many different guesses regarding the strategy that $\mathbf{P}_0$ uses to answer his questions. Either $\mathbf{P}_0$ has a significant probability of failing the CONSISTENCY TEST, or $\mathbf{P}_1$ has a significant probability of making the right guess among its many guesses. Prover $\mathbf{P}_1$ fixes, for each guess, a strategy for answering the questions he receives. The verifier V expects that

each strategy that $\mathbf{P}_1$ fixes corresponds to a functional behavior. But, if $\mathbf{P}_1$ is dishonest, he has no incentive in doing so. Hence, the verifier also allows $\mathbf{P}_2$ to make constantly many 'educated' guesses for each of the guesses that $\mathbf{P}_1$ makes, and so on and so forth.

At the end of the protocol, V can determine which of the provers' guesses are correct. With high probability, either cheating provers fail the CONSISTENCY TEST, or V can determine, among the correct guesses, a set of runs most of which can be treated as independent executions of $\widehat{V}$'s protocol. This suffices for achieving our error reduction goals.

Below we describe the precise protocol implemented by the verifier V. We then informally discuss the motivation behind the main components of the protocol.

THE QUERIES

— V chooses $I \subset \{1,\ldots,m\}$, $|I| = n$,[9] at random, and selects for every $i \in I$ a random string r_i as the old verifier $\widehat{V}$ would have. The runs indexed by I will be called the *compare* runs, the other runs will be called *confuse* runs.

— For every confuse run i and every prover $\mathbf{P}_l$ the verifier V selects random string $r_i^{(l)}$ as the old verifier $\widehat{V}$ would have.

For $i \in \{1,\ldots,m\}$, $l \in \{0,\ldots,k\}$ let $\rho_i^{(l)} = r_i$ if i is a compare run, $r_i^{(l)}$ otherwise.

— For every prover $\mathbf{P}_l$ the verifier V chooses $I_l \subset I \setminus \bigcup_{j<l} I_j$, $|I_l| = \gamma n$, at random together with a random ordering π_l of I_l for $l \in \{0,\ldots,k\}$.[10]

— V sends to prover $\mathbf{P}_l$, $l \in \{0,\ldots,k\}$, $\left(q^{(l)}(\rho_i^{(l)}) : i \in \{1,\ldots,m\} \setminus \bigcup_{j<l} I_j\right)$.[11]

— V sends to prover $\mathbf{P}_{l+1}$, $l \in \{0,\ldots,k-1\}$ all questions $\left(q^{(j)}(r_i) : i \in I_j\right)$, the set of indices I_j, and the orderings π_j, for all $j \in \{0,\ldots,l\}$. (We will see that this helps honest provers to make educated guesses about the strategies used by cheating provers that pass the CONSISTENCY TEST with a significant probability.)

— In order to implement the CONSISTENCY TEST the verifier V sends to provers $\mathbf{P}_\exists$ and $\mathbf{P}_\forall$ the questions $\left(q^{(l)}(r_i) : i \in I_l\right)$, the set of indices I_l and the orderings π_l, for all $l \in \{0,\ldots,k\}$.

The format of each prover's answer is a graph, more precisely a rooted tree.

For a rooted tree T, with node set $V(T)$ and edge set $E(T)$, we say that a node $v \in V(T)$ is at level i if its distance to the root of T is i. We say that an edge $e \in E(T)$ is at level i if it is rooted at a node of level i.

Definition 14 *Let J be a set of indices. We say that a tree T is a J-*tree *if nodes at level $j \in J$ have only one child.*

[9] We later set $m \gg n$.

[10] We later set $\gamma = \frac{1}{2(k+1)}$.

[11] When a question is sent to a prover the verifier indicates the run $i \in \{1,\ldots,m\}$ to which the question corresponds.

We consider trees whose nodes and edges are labeled. Internal nodes will be labeled by sets of indices. Edges rooted at a node labeled J will be labeled by a tuple of strings $(a_j : j \in J)$. Leaves will also be labeled by a tuple of strings. We denote the label of a node v by $\mathcal{L}_T(v)$, and the label of the edge e by $\mathcal{L}_T(e)$.

Definition 15 *Let v be a node of level l of tree T with root v_0. Let $v_0, \ldots, v_{l-1}$ be the sequence of nodes along the path from the root of T to v. Define the history of v as follows:*

$$\mathcal{H}_T(v) \;=\; [\mathcal{L}_T(v_0), \mathcal{L}_T(v_0, v_1)] \cdots [\mathcal{L}_T(v_{l-1}), \mathcal{L}_T(v_{l-1}, v)] \,.$$

If v is the root of T we say that its history is empty and we denote it by $\emptyset$.

THE RESPONSES

— Prover $\mathbf{P}_l$ responds with a tree T_l, $\mathbf{P}_\exists$ with a tree $T_\exists$, and $\mathbf{P}_\forall$ with a tree $T_\forall$.

— T_l, for $l \in \{0, \ldots, k\}$ is a tree of depth l labeled as follows:

- An internal node v at level $j \in \{0, \ldots, l-1\}$ is labeled by I_j. Every internal node v at level j has an edge rooted at v for every possible tuple of answers of $\mathbf{P}_j$ to questions $\left(q^{(j)}(r_i) : i \in I_j\right)$, and labeled by this tuple of answers.
- A leaf is labeled either FAILURE, SUCCESS, or by a tuple of answers $\left(a_i^{(l)} : i \in \{1, \ldots, m\} \setminus \bigcup_{j<l} I_j\right)$. If $\mathbf{P}_l$ is honest, he labels the leaves of T_l as shown later under HONEST PROVERS PROCEDURE FOR LABELING LEAVES.

— $T = T_\exists$ (respectively $T = T_\forall$) is a K-tree of depth $k+1$, where $K = \{0, 2, \ldots, k-1, k\}$ (respectively $K = \{1, 3, \ldots, k-2\}$), labeled as follows:

- Nodes are labeled as the nodes of the trees described above. Edges at levels $l \notin K$ are also labeled as in the trees described above.
- The only edge of level $l \in K$ rooted at node v of T is labeled by a tuple of answers $\left(a'^{(l)}_i : i \in I_l\right)$. An honest prover sets $a'^{(l)}_i = a_i^{(l)}$ for all $i \in I_l$, where $\left(a_i^{(l)} : i \in \{1, \ldots, m\} \setminus \bigcup_{j<l} I_j\right)$ is the label of the leaf of T_l with history $\mathcal{H}_T(v)$. (Note that T belongs to T_l's team.)

HONEST PROVERS PROCEDURE FOR LABELING LEAVES

Assume $\mathbf{P}_l$ is honest, $l \in \{0,\ldots,k\}$, then $\mathbf{P}_l$ will generate a tree T_l of the proper format. Consider the leaf u_l of T_l. Let $u_0,\ldots,u_l$ be a path in T_l from the root u_0 to the leaf u_l. In labeling u_l, $\mathbf{P}_l$ considers the strategies $f_0,\ldots,f_{l-1}$ that $\mathbf{P}_0,\ldots,\mathbf{P}_{l-1}$ use in labeling the leaves (abusing notation) $u_0,\ldots,u_{l-1}$ of $T_0,\ldots,T_{l-1}$ with history $\mathcal{H}_{T_l}(u_0),\ldots,\mathcal{H}_{T_l}(u_{l-1})$ respectively. For $s \in \{0,\ldots,l-1\}$, let j_s be the smallest good j, as defined by Lemma 61 (hence $j < \gamma n$), for f_s. Let $\widetilde{I}_s \subseteq I_s$ be the set of the first j_s indices of I_s as determined by π_s. Let $\widetilde{Q}_s = (q^{(s)}(r_i) : i \in \widetilde{I}_s)$ be the set of questions on runs $\widetilde{I}_s$, and let $\widetilde{A}_s = (\widetilde{a}_i^{(s)} : i \in \widetilde{I}_s)$ be the set of answers in runs $\widetilde{I}_s$ induced by the label of the leaf u_s. Define the following events:

Event E_1 : $\forall s \in \{0,\ldots,l-1\}$, the answer sets $\widetilde{A}_s$ are live for the f_s-challenge $(\widetilde{I}_s, \widetilde{Q}_s)$.

Event E_2 : $\forall s \in \{0,\ldots,l-1\}$, more than a ζ fraction of the f_s-challenges $(\widetilde{I}_s, \widetilde{Q}_s)$ (over the choices of $\widetilde{I}_s = \{i_1,\ldots,i_{j_s}\}$ and $(r_i : i \in \widetilde{I}_s)$) are live.

For an appropriate choice of parameters we can distinguish two cases,

Events E_1 and E_2 occur: $\forall i \in \{1,\ldots,m\} \setminus \bigcup_{s<l} I_s$, $\forall s \in \{0,\ldots,l-1\}$, $\mathbf{P}_l$ determines (if possible) the functions $F_{f_s,i,\widetilde{Q}_s,\widetilde{A}_s}$ (as in Lemma 61), and the optimal strategy for answering run i, say $P_i^{u_l}$, of the $(l+1)$-th prover $\widehat{\mathbf{P}}_l$ assuming that the first l provers strategies are $F_{f_0,i,\widetilde{Q}_0,\widetilde{A}_0},\cdots,F_{f_{l-1},i,\widetilde{Q}_{l-1},\widetilde{A}_{l-1}}$. If $\mathbf{P}_l$ is unable to determine some $F_{f_s,i,\widetilde{Q}_s,\widetilde{A}_s}$ he labels u_l with FAILURE, that is, $\mathbf{P}_l$ recognizes that cheating provers have outsmarted him. Otherwise, $\mathbf{P}_l$ answers run i of u_l according to $P_i^{u_l}$.

Otherwise: $\mathbf{P}_l$ labels u_l with SUCCESS, that is, $\mathbf{P}_l$ expresses its confidence that the cheating provers will not pass the CONSISTENCY TEST.

The verifier will reject the claim of any team of provers that fails to respond in the proper format. Thus, we only focus in the case that $T_0,\ldots,T_k$, and $T_\exists, T_\forall$ are in the proper format.

In particular we are interested on a specific quantity associated to tree T_l, for every $l \in \{0,\ldots,k\}$. This quantity will be denoted by $\mathcal{H}(T_0,\ldots,T_l)$, and is recursively defined as follows: $\mathcal{H}(T_0) = \left[I_0, \left(a_i^{(0)} : i \in I_0\right)\right]$. If v_l is the leaf of T_l with history $\mathcal{H}(T_0,\ldots,T_{l-1})$ and label $\left(a_i^{(l)} : i \in \{1,\ldots,m\} \setminus \bigcup_{j<l} I_j\right)$ then $\mathcal{H}(T_0,\ldots,T_l) \stackrel{\text{def}}{=}$

$$\mathcal{H}(T_0,\ldots,T_{l-1})\left[I_l, \left(a_i^{(l)} : i \in I_l\right)\right] = \left[I_0, \left(a_i^{(0)} : i \in I_0\right)\right]\cdots\left[I_l, \left(a_i^{(l)} : i \in I_l\right)\right].$$

Note that there is only one leaf in $T_\exists$ and another in $T_\forall$ both of which have the same history. This history determines a common branch of $T_\exists$ and $T_\forall$ which we denote by,

$$\mathcal{H}'(T_\exists, T_\forall) \stackrel{\text{def}}{=} \left[I_0, \left(a_i'^{(0)} : i \in I_0\right)\right]\cdots\left[I_k, \left(a_i'^{(k)} : i \in I_k\right)\right].$$

ACCEPTANCE CRITERIA

— If a prover does not comply with the proper format of the responses, then, the verifier V rejects the claim of the team to which he belongs. (Honest provers will always answer in the proper format.)

— The verifier V implements the following CONSISTENCY TEST: V determines the largest $l \in \{0, \ldots, k+1\}$ such that $\mathcal{H}(T_0), \ldots, \mathcal{H}(T_0, \ldots, T_{l-1})$ are prefixes of $\mathcal{H}'(T_\exists, T_\forall)$. Two different situations may arise:

- If $l > k$, then we say that the consistency test PASSES.
- Otherwise, the verifier rejects the claim of the team to which prover $\mathbf{P}_l$ belongs.

— If the previous test passes then, V accepts, if and only if, for more than a $1 - \frac{1}{cN}$ fraction of the runs $i \in I \setminus \bigcup_{j \leq k} I_j$ the old verifier $\widehat{V}$ on random string r_i would accept if given answers $a_i^{(0)}, \ldots, a_i^{(k)}$ from provers $\widehat{\mathbf{P}}_0, \ldots, \widehat{\mathbf{P}}_k$ respectively.[12]

It is important to note that from the point of view of the verifier V there is only one branch in each of the trees $T_0, \ldots, T_k$ and $T_\exists, T_\forall$ which is of relevance.

The idea behind the above described protocol is the following: the verifier V wants to simulate $n/2$ rounds of $\widehat{V}$'s protocol. In order to do this, V generates n sets of questions as $\widehat{V}$ would have and sends them to the first k provers. Half of these questions will be used to implement the CONSISTENCY TEST using the two last provers. Moreover, these n questions are sent to the first k provers among a much larger set of $m-n$ questions, for a total of m questions. This latter type of questions are called *confuse questions*, since their purpose is to confuse cheating provers. The provers do not know which questions are/are not confuse questions. There is no dependency whatsoever between the confuse questions sent to different provers. If cheating provers answer using a global strategy, then their responses will depend on the confuse questions. This will reduce their probability of passing the CONSISTENCY TEST because the last two provers do not receive the confuse questions. We are left to argue what happens when the provers have a non-negligible probability of passing the CONSISTENCY TEST. The description of the provers' responses show how honest provers can choose their strategies. The next lemma establishes that the parameters of the above described simulation protocol can be chosen so the final error is small, and the blowup in the size of the provers' responses and the overall number of random bits used by the verifier is a constant factor.

Lemma 62 *(k odd) Let $k = 2t + 1$. For any constant α, $0 < \alpha < 1/2$. Every language L recognized by a $(k+1)$-alternating-prover proof system with parameters $([(\exists, \forall)^t, \exists, \exists])$ and error $\left(0, 1 - \frac{1}{N}\right)$, where N is a large constant, has a $(k+3)$-alternating-prover proof system with parameters $([(\exists, \forall)^t, \forall, \exists, \exists, \exists])$ and error α.*

Proof: Denote by $\widehat{V}$ the $(k+1)$-alternating-prover proof systems' verifier, and by $\widehat{\mathbf{P}}_0, \ldots, \widehat{\mathbf{P}}_k$ its provers. Moreover, let V denote the verifier which interacts with provers $\mathbf{P}_0, \ldots, \mathbf{P}_k$ and $\mathbf{P}_\exists, \mathbf{P}_\forall$ according to the

[12] Where c is a large constant whose value we will set later.

simulation protocol previously described. Clearly, the latter proof system is a $(k+3)$-alternating-prover one-round proof system where the provers are quantified according to $([(\exists,\forall)^t,\forall,\exists,\exists,\exists])$. We have to show that there is an appropriate choice of parameters for which this proof systems achieves error α, uses logarithmic number of random bits, and where the provers' responses are constant size.

First, note that honest provers never fail the CONSISTENCY TEST.

Now, consider the strategy f_0 that prover $\mathbf{P}_0$ uses in labeling the only leaf of T_0, and the strategies $f_1,\ldots,f_k$ that $\mathbf{P}_1,\ldots,\mathbf{P}_k$ use in labeling the leaves $v_1,\ldots,v_k$ of $T_0,\ldots,T_k$ with history $\mathcal{H}(T_0),\ldots,\mathcal{H}(T_0,\ldots,T_k)$ respectively. For $s\in\{0,\ldots,k\}$, let j_s be the smallest good j, as defined by Lemma 61, for f_s. Let $\widetilde{I}_s\subseteq I_s$ be the set of the first j_s indices of I_s as determined by π_s. Let $\widetilde{Q}_s=(q^{(s)}(r_i):i\in\widetilde{I}_s)$ be the set of questions on runs $\widetilde{I}_s$, and let $\widetilde{A}_s=(\widetilde{a}_i^{(s)}:i\in\widetilde{I}_s)$ be the set of answers in runs $\widetilde{I}_s$ induced by the label of the leaf v_s. Finally, define the events E_1 and E_2 as follows

Event E_1 : $\forall s\in\{0,\ldots,k\}$, the answer sets $\widetilde{A}_s$ are live for the f_s-challenge $(\widetilde{I}_s,\widetilde{Q}_s)$.

Event E_2 : $\forall s\in\{0,\ldots,k\}$, more than a ζ fraction of the f_s-challenges $(\widetilde{I}_s,\widetilde{Q}_s)$ (over the choices of $\widetilde{I}_s=\{i_1,\ldots,i_{j_s}\}$ and $(r_i:i\in\widetilde{I}_s)$) are live.

We distinguish the following cases:

If E_1 does not occur: then, there is an $s\in\{0,\ldots,k\}$ such that $\widetilde{A}_s$ is not a live answer for the f_s-challenge $(\widetilde{I}_s,\widetilde{Q}_s)$. It follows that $\mathbf{P}_s$ is a cheating prover. By Lemma 60, except for a probability of at most $n\epsilon$ (over the choices of $I\setminus\widetilde{I}_s$ and $\{r_i\mid i\in I\setminus\widetilde{I}_s\}$), $\mathbf{P}_s$ will pass the CONSISTENCY TEST with probability at most $\zeta+n\epsilon$. thus cheating provers pass the test with probability at most $\zeta+2n\epsilon\le 3\zeta$ (assuming $n\epsilon\le\zeta$).

If E_2 does not occur: then, the probability that event E_1 occurs is at most ζ. The preceding case analysis bounds the probability that cheating provers will pass the CONSISTENCY TEST when event E_1 does not occur by 3ζ. Thus, if E_2 does not occur, then cheating provers will pass the CONSISTENCY TEST with probability at most 4ζ.

If E_1 and E_2 occur: then, Lemma 61 implies that with probability at most $\frac{(k+2)}{2}\zeta$ the honest provers concede FAILURE in one of the leafs $v_0,\ldots,v_k$. Otherwise, with probability at least $1-\frac{k+2}{2}\delta$ at least $(1-(k+1)\gamma-\frac{(k+2)\zeta}{\delta})n$ of the runs $i\in I\setminus\bigcup_{j\le k}I_j$ are *good*, i.e. they are answered according to $F_{f_0,i,\widetilde{Q}_0,\widetilde{A}_0},\ldots,F_{f_k,i,\widetilde{Q}_k,\widetilde{A}_k}$.

Let $p=\frac{(k+2)}{2}(\delta+\zeta)+7\zeta$, and $n'=(1-(k+1)\gamma)n$. Now, choose $\gamma=\frac{1}{2(k+1)}$, $\delta=\sqrt{\zeta}$, ζ small enough so $\zeta\le 1/(2cN(k+2))^2$ (hence $\mu=(1-(k+1)\gamma-\frac{(k+2)\zeta}{\delta})\frac{n}{n'}\ge 1-\frac{1}{cN}$) and $p=\frac{(k+2)}{2}(\sqrt{\zeta}+\zeta)+7\zeta\le\alpha/2$, n large enough so $n\ge 2N\log(2/\alpha)$ and $n>2\zeta^{-5}/\gamma$, ϵ small enough so $8\epsilon<\zeta^5$ and $n\epsilon\le\zeta$, M large enough so if $M=m-n$, $\epsilon\ge\max\left\{\frac{256\ln M}{M},\left(\frac{8\ln M}{M}\right)^{1/3}\right\}$.

All the above mentioned parameters are independent of the size of the input. Hence, the simulation protocol requires only a constant factor blowup in the size of the provers' responses and in the amount of randomness used by the verifier.

We refer to the runs indexed by elements of $I \setminus \bigcup_{j \leq k} I_j$ as the *relevant* runs.

Assume $\omega \in L$. Then, with probability at least $1 - p \geq 1 - \alpha/2$, a $\mu \geq 1 - \frac{1}{cN}$ fraction of the relevant runs are good runs. But, a good run would make the old verifier $\widehat{V}$ accept. Thus, a fraction of at least $1 - \frac{1}{cN}$ of the relevant runs make the verifier $\widehat{V}$ accept. Hence, V accepts with probability at least $1 - \alpha$.

Assume $\omega \notin L$. Let ρ be the fraction of good runs among the relevant runs. Recall that the verifier V will accept if more than a $1 - \frac{1}{cN}$ fraction of the n' relevant runs would cause the old verifier $\widehat{V}$ to accept. Hence, V accepts if more than $\left(\rho - \frac{1}{cN}\right) n'$ of the $\rho n'$ good runs would make the old verifier $\widehat{V}$ accept. But, a good run makes $\widehat{V}$ accept with probability at most $1 - \frac{1}{N}$. Note that the probability that $\rho < \mu$ is at most $p \leq \alpha/2$. A bound on the tail of a binomial distribution (see e.g. [CLR90b, Theorem 6.2]) shows that we can choose c large enough so that the probability that more than $\left(\rho - \frac{1}{cN}\right) n'$ of the good $\rho n'$ runs make $\widehat{V}$ accept is at most $\alpha/2$. Hence the overall acceptance probability of the verifier V is at most α. □

Remark 2 *Note that the task of performing the* CONSISTENCY TEST *can be distributed among the provers. Provers* $\mathbf{P}_\exists$ *and* $\mathbf{P}_\forall$ *are only needed in order to insure the functional behavior of the last provers (among* $\mathbf{P}_0, \ldots, \mathbf{P}_k$*) that are quantified with* $\exists$ *and* $\forall$ *respectively. Indeed, the simulation protocol described in this subsection allows provers more leeway than necessary. In fact, V should also request that the tree* T_l *with which* $\mathbf{P}_l$ *responds be a* J_l*-tree, where* J_l *is the set of indices* $j \in \{0, \ldots, l-1\}$ *for which* $\mathbf{P}_j$ *is in the same team that* $\mathbf{P}_l$ *is. For every* $l \in \{0, \ldots, k\}$ *and* $j \in J_l$*, the verifier V will now check that the label of the unique edge of level j rooted at the node v of* T_l *is consistent with the set of answers that the leaf of* T_j *with history* $\mathcal{H}_{T_l}(v)$ *induces in runs* I_j*. If one of these consistency check fails, then V rejects the claim of prover* $\mathbf{P}_l$*'s team, where l is the smallest index for which one of the above consistency check fails.*

The previous observation will be crucial for achieving the goals of the next subsection.

5.7.2 Fixing the Quantifiers

We now show that the conclusion of Lemma 62 still holds even if we are restricted to using $k+1$ provers, all of whose quantifiers alternate. More precisely, we have the following:

Lemma 63 *(k odd) Let* $k = 2t + 1$*. For any constant* ϵ*,* $0 < \epsilon < 1/2$*. Every language L recognized by a* $(k+3)$*-alternating-prover proof system with parameters* $([(\exists, \forall)^t, \forall, \exists, \exists, \exists])$ *and error* α *for any constant* α*, can be recognized by a* $(k+1)$*-alternating-prover proof system with parameters* $([(\exists, \forall)^{t+1}])$ *and error* ϵ*.*

Proof: [Sketch] Again, the proof consists in simulating the first proof system by the second one. There are three major components to this simulation. Namely, the simulation protocols in which the proofs of Theorem 52, Lemma 55 and Lemma 62 are based. We denote by $\widehat{V}$ the $(k+3)$-alternating-prover proof systems' verifier,

and its provers by $\widehat{\mathbf{P}}_0, \ldots, \widehat{\mathbf{P}}_{k-1}$ and $\widehat{\mathbf{P}}'_{k-2}, \widehat{\mathbf{P}}'_{k-1}, \widehat{\mathbf{P}}''_{k-1}$. The provers' ordering is such that $\widehat{\mathbf{P}}'_{k-2}$ follows immediately after $\widehat{\mathbf{P}}_{k-2}$, and $\widehat{\mathbf{P}}'_{k-1}, \widehat{\mathbf{P}}''_{k-1}$ follow $\widehat{\mathbf{P}}_{k-1}$ (in this order). Hence, the provers with subindex $k-2$ are quantified by $\forall$, and the ones with subindex $k-1$ are quantified by $\exists$. We will simulate this proof system by a $(k+1)$-alternating-prover-proof system with verifier V, and provers $\mathbf{P}_0, \ldots, \mathbf{P}_{k-1}, \mathbf{P}_\forall$ all of whose quantifiers alternate starting with $\exists$. We refer to the former proof system as the old proof system, and to the latter one as the new proof system. Note, that an old prover with subindex i is quantified as the new prover with subindex i is. We would like to simulate an old prover with subindex i by the new prover with subindex i. Below we illustrate the relation among the old and new provers.

$$
\begin{array}{ccccccccc}
\overbrace{\phantom{\widehat{\mathbf{P}}_0}}^{\exists} & & \overbrace{\phantom{\widehat{\mathbf{P}}_{k-3}}}^{\exists} & & \overbrace{\phantom{\widehat{\mathbf{P}}_{k-2}\widehat{\mathbf{P}}'_{k-2}}}^{\forall} & & \overbrace{\phantom{\widehat{\mathbf{P}}_{k-1}\widehat{\mathbf{P}}'_{k-1}\widehat{\mathbf{P}}''_{k-1}}}^{\exists} & & \\
\underbrace{\widehat{\mathbf{P}}_0} & \cdots & \underbrace{\widehat{\mathbf{P}}_{k-3}} & & \underbrace{\widehat{\mathbf{P}}_{k-2}\widehat{\mathbf{P}}'_{k-2}} & & \underbrace{\widehat{\mathbf{P}}_{k-1}\widehat{\mathbf{P}}'_{k-1}\widehat{\mathbf{P}}''_{k-1}} & & \\
\downarrow & \cdots & \downarrow & & \downarrow & & \downarrow & & \\
\mathbf{P}_0 & \cdots & \mathbf{P}_{k-3} & & \mathbf{P}_{k-2} & & \mathbf{P}_{k-1} & & \underbrace{\mathbf{P}_\forall}_{\forall}
\end{array}
$$

The verifier V interacts with the provers as in the simulation protocol discussed in the previous subsection, but modified according to Remark 2 and as described below. We first describe the queries that V makes; For $l \leq k-1$, denote by $q^{(l)}(r)$ the question that $\widehat{V}$ asks $\widehat{\mathbf{P}}_l$ on random string r. Moreover, let $q'^{(k-1)}(r)$, $q'^{(k-2)}(r)$ and $q''^{(k-2)}(r)$ denote the questions that $\widehat{V}$, on random string r, asks $\widehat{\mathbf{P}}'_{k-1}$, $\widehat{\mathbf{P}}'_{k-2}$ and $\widehat{\mathbf{P}}''_{k-2}$ respectively.

THE QUERIES

— V chooses $I \subset \{1, \ldots, m\}$, $|I| = n$, at random, and selects for every $i \in I$ a random string r_i as $\widehat{V}$ would have. The runs indexed by I will be called the *compare* runs, the other runs will be called *confuse* runs.

— For every confuse run i and every $l \leq k-2$, V selects random string $r_i^{(l)}$ as $\widehat{V}$ would have.[13] For every run i, V randomly chooses $a_i \in \{0,1\}$ and $b_i \in \{0,1,2\}$.

For every run i and $l < k-1$, let $\rho_i^{(l)} = r_i$ if i is a compare run, and $r_i^{(l)}$ otherwise.

— For every $l \leq k-2$, the verifier V chooses $I_l \subset I \setminus \bigcup_{j<l} I_j$, $|I_l| = \gamma n$, at random together with a random ordering π_l of I_l.

— For every $l \leq k-3$, the verifier V sends to prover $\mathbf{P}_l$ the questions $\left(q^{(l)}(\rho_i^{(l)}) : i \in \{1, \ldots, m\} \setminus \bigcup_{j<l} I_j\right)$.

— V sends to $\mathbf{P}_{k-2}$ questions $\left(q_{a_i}^{(k-2)}(\rho_i^{(k-2)}) : i \in \{1, \ldots, m\} \setminus \bigcup_{j<k-2} I_j\right)$, where $q_{a_i}^{(k-2)}(r)$ equals $(q^{(k-2)}(r), 0)$ if $a_i = 0$, and $(q'^{(k-2)}(r), 1)$ otherwise.

— V sends to $\mathbf{P}_{k-1}$ questions $\left(q_{b_i}^{(k-1)}(r_i) : I \setminus \bigcup_{j<k-1} I_j\right)$, where $q_{b_i}^{(k-1)}(r)$ equals $(q^{(k-1)}(r), 0)$ if $b_i = 0$, $(q'^{(k-1)}(r), 1)$ if $b_i = 1$, and $(q''^{(k-1)}(r), 2)$ otherwise.

— For $l \leq k-2$, the verifier V sends to $\mathbf{P}_{l+1}$ the questions that were sent to $\mathbf{P}_j$ in runs I_j, the set of indices I_j, and the orderings π_j, for all $j \leq l$.

— V sends to $\mathbf{P}_\forall$ the questions that were sent to $\mathbf{P}_l$ in runs I_l, the set of indices I_l, and the orderings π_l, for all $l \leq k-2$. V also sends the pairs of questions $\left((q^{(k-2)}(r_i), q'^{(k-2)}(r_i)) : i \in I \setminus \bigcup_{j<k-2} I_j\right)$, and the triplets of questions $\left((q^{(k-1)}(r_i), q'^{(k-1)}(r_i), q''^{(k-1)}(r_i)) : i \in I \setminus \bigcup_{j<k-1} I_j\right)$,

The format of the provers' responses, the honest provers' answers, and the verifier's acceptance criteria correspond to modifications of the simulation protocol of Subsection 5.7.1. We now list these modifications.

— Prover $\mathbf{P}_l$, $l \leq k-1$, responds with a J_l-tree, denoted T_l, of the form described in Remark 2.

— Prover $\mathbf{P}_\forall$ responds with two trees, $T_\exists$ and $T_\forall$. Tree $T_\forall$ is a tree of the same kind and depth as the response tree T_{k-2}. Tree $T_\exists$ is a tree of the same kind and depth as the response tree T_{k-1}.

— Provers $\mathbf{P}_0, \ldots, \mathbf{P}_{k-3}$ should respond as in the simulation protocol of Subsection 5.7.1.

— At each leaf of its response tree prover $\mathbf{P}_{k-2}$ is required to simulate $\widehat{\mathbf{P}}_{k-2}, \widehat{\mathbf{P}}'_{k-2}$. This can be done with the help of $\mathbf{P}_\forall$ without giving too much advantage to cheating provers. Indeed, consider the leaf

[13] There are will be no questions sent to $\mathbf{P}_{k-1}$ in the confuse runs.

v of T_{k-2}. Let f denote the strategy by which $\mathbf{P}_{k-2}$ labels v. Note that f is a function defined over pairs (q, b), where q is a question and $b \in \{0, 1\}$. Hence, f implicitly defines two strategies $f(\cdot, 0)$ and $f(\cdot, 1)$. The verifier V requires that $\mathbf{P}_\forall$ label the leaf of $T_\forall$ with history $\mathcal{H}_T(v)$ according to these two strategies. Thus, $\mathbf{P}_\forall$ reveals the answers to the questions that were not sent to $\mathbf{P}_{k-2}$. Prover $\mathbf{P}_\forall$ succeeds in misrepresenting s of the answers that $\mathbf{P}_{k-2}$ would have given with probability at most $1/2^s$.

— At each leaf of its response tree prover $\mathbf{P}_{k-1}$ is required to simulate $\widehat{\mathbf{P}}_{k-1}$, $\widehat{\mathbf{P}}'_{k-1}$ and $\widehat{\mathbf{P}}''_{k-1}$. Lemma 55 shows how this can be done with the help of $\mathbf{P}_\forall$. Indeed, for each leaf v of $T_\exists$, prover $\mathbf{P}_\forall$, responds as the second prover of the two-alternating-prover proof system of Lemma 55 would have responded to a prover that answers according to the strategy that prover $\mathbf{P}_{k-1}$ uses for labeling the leaf of T_{k-1} with history $\mathcal{H}_{T_\exists}(v)$.

Upon receiving the provers responses V will perform all the consistency checks suggested by the preceding discussion. If one these checks fails, then V rejects the claim of the team found at fault. Otherwise, as in the simulation protocol of Subsection 5.7.1, V only focuses in a unique branch of $T_0, \ldots, T_k$ and $T_\exists, T_\forall$ which is of relevance. The verifiers acceptance criteria corresponds to the obvious extension of the acceptance criteria of the simulation protocol in which the proof of Lemma 55 is based.

For an appropriate choice of parameters, the proof of correctness of the simulation protocol just described follows from the same arguments used to prove Theorem 52, Lemma 55 and Lemma 62. □

We can now prove our main theorem:

Proof of Theorem 50: We prove the theorem for the case where k is odd.

For the reverse direction, assume L has a $(k+1)$-prover proof system with error $\epsilon < 1/2$. Consider the alternating Turing machine that on input ω performs the following three steps: first, it uses its i-th level of alternation, $i \in \{1, \ldots, k\}$, to guess the strategy of the i-th prover (i.e. guesses the constant-size answer to each of the $poly(|\omega|)$ many different question that the i-th prover might receive). Second, it computes the optimal strategy of the last prover (i.e. computes for each one of the $poly(|\omega|)$ many questions that the last prover might receive, the optimal constant-size response). Finally, it determines the probability that the verifier accepts (by simulating the verifier in each one of his $poly(|\omega|)$ many random strings), and accepts, if and only if, the verifier's acceptance probability is at least $1 - \epsilon$. Clearly, this machine recognizes L, hence, $L \in \Sigma_{\mathbf{k}}^{\mathbf{P}}$.

The forward direction follows from Theorem 58, Lemma 62 and Lemma 63. □

Corollary 64 *If k is a positive integer, then* $\Sigma_{\mathbf{k-1}}^{\mathbf{P}} = \mathbf{k\text{-}APP}$.

5.7.3 Comments

In this subsection we make some general comments and give evidence that the two sided error requirement of Theorem 50 is necessary. Indeed, unless the polynomial-time hierarchy collapses, all $\Sigma_{\mathbf{k-1}}^{\mathbf{P}}$ languages cannot have k-alternating-prover proof systems with one sided error.

Lemma 65 *If languages in $\Sigma_k^{\mathbf{P}}$ have $(k+1)$-alternating-prover proof system with error $(0, \epsilon)$, then the polynomial-time hierarchy collapses to its $\Sigma_{k-1}^{\mathbf{P}}$.*

Proof: We prove the lemma for the case $k = 1$. The extension is straightforward.

Let L be a language in $\mathbf{NP}$, and assume it has a two-alternating-prover proof system with verifier V and provers $\mathbf{P}_0$ and $\mathbf{P}_1$ quantified by $\exists$ and $\forall$ respectively. Assume that ω is the input to the alternating-prover proof system. For $i \in \{0,1\}$, let $q^{(i)}(r)$ be the question that V on random string r sends to $\mathbf{P}_i$, Q_i be the set of all possible question that $\mathbf{P}_i$ can receive, and A_i be the set of possible answers of $\mathbf{P}_i$. Moreover, let R the set of random strings that V can select, and let $V(\omega, r, a_0, a_1) = accept$ if V accepts on input ω, random string r, and if given response a_0 from $\mathbf{P}_0$ and a_1 from $\mathbf{P}_1$.

If the two-alternating-prover proof system under consideration has one sided error $(0, \epsilon)$, then, the following are equivalent:

— there exists a strategy P_0 for prover $\mathbf{P}_0$, such that for any strategy P_1 for prover $\mathbf{P}_1$, for all random strings $r \in R$, $V(\omega, r, a_0, a_1) = accept$, where $a_0 = P_0(q^{(0)}(r))$ and $a_1 = P_1(q^{(1)}(r))$,

— $\forall q_0 \in Q_0, \exists a_0 \in A_0$, such that $\forall a_1 \in A_1$ and $\forall r \in R$ such that $q^{(0)}(r) = q_0$, $V(\omega, r, a_0, a_1) = accept$.

The first statement holds, if and only if, $\omega \in L$. Since $|A_0|, |A_1| = O(1)$, $|R|, |Q_0|, |Q_1| = 2^{O(\log |\omega|)}$, and $V(\cdot, \cdot, \cdot, \cdot)$ is polynomial-time computable, the truth value of the last statement above can be decided in polynomial time. Thus, the language L can be recognized in polynomial time. Hence, the polynomial-time hierarchy collapses to $\mathbf{P}$. □

Under complexity theoretic assumptions we can use Theorem 50 and Theorem 58 to prove hardness of approximation results. This results hold for either artificially constructed optimization problems or problems of only theoretical importance. We hope non-approximability results for non-artificially constructed problems will be found.

5.8 Concluding Remarks

More direct proofs – ones that avoid the Feige and Kilian lemmas – of the results presented in this chapter are desirable. Algebraic arguments are a natural candidate for simplifying our proofs. We hope this will be accomplished in the near future.

Chapter 6

Symmetric Alternation

6.1 Motivation

Since the inclusion of randomness among those resources for which we have agreeable computational models, determination of the exact relationship between randomness and other such computational resources has become a major project. The relationship between *space* and randomness, elucidated by startling pseudo-random constructions [AKS87, NZ93], is remarkably well understood. These constructions demonstrate that space-bounded computation benefits little by the use of randomness. The analogous relationship with *time* has proved less tractable: the only (non-trivial) relationships depend on unproven complexity-theoretic assumptions [ILL89, Yao82]. We continue the study initiated by Sipser [Sip83] of the relationship between randomness and *quantification*, that is, the relationship between $\mathbf{BPP}$ and classes arising by appropriate quantification of polynomial-time predicates (e.g., $\mathbf{NP}$, $\mathbf{coNP}$ and other classes in the polynomial-time hierarchy [Sto76b]).

6.2 Summary of Results

$\mathbf{BPP}$ was first shown to lie in the polynomial-time hierarchy in [Sip83] which demonstrates that $\mathbf{BPP} \subseteq \Sigma_2^P$ (and hence that $\mathbf{BPP} \subset \Sigma_2^P \cap \Pi_2^P$). We introduce a natural quantified class, $\mathbf{S}_2^P$, and demonstrate that

$$\mathbf{BPP} \subset \mathbf{S}_2^P \subset \Sigma_2^P \cap \Pi_2^P.$$

Given the unnaturality of $\Sigma_2^P \cap \Pi_2^P$, the naturality of $\mathbf{S}_2^P$ suggests that $\mathbf{S}_2^P \subsetneq \Sigma_2^P \cap \Pi_2^P$ and hence that this containment is more than a solely conceptual improvement.

The class $\mathbf{S}_2^P$ consists of those languages $\mathcal{L}$ which may be decided by a polynomial-time machine that receives counsel from two provers in such a manner that when the input $x \in \mathcal{L}$ there is a "witness" w_1 which the first prover may provide so that regardless of the information provided by the second prover, the machine

accepts and, similarly, when $x \notin \mathcal{L}$, there is a "witness" w_2 which the second prover may supply so that, regardless of the information supplied by the first prover, the machine rejects. The special sort of alternation involved here we refer to as *symmetric alternation*. The $\mathbf{S}_2$ operator enjoys some remarkable structural properties:

- $\mathbf{S}_2 \cdot \mathbf{BP} \cdot \mathbf{P} \subseteq \mathbf{S}_2 \cdot \mathbf{P}$,
- $\mathbf{P}^{(\mathbf{S}_2 \cdot P)} = \mathbf{S}_2 \cdot P$.

We use these structural properties to conclude that

- $\mathbf{BPP} \subseteq \mathbf{MA} \subset \mathbf{S}_2^P$, and
- $\Delta_2^P = \mathbf{P}^{\mathbf{NP}} \subseteq \mathbf{S}_2^P$.

Considering the strong structural properties which $\mathbf{S}_2^P$ enjoys, it seems unlikely that $\mathbf{S}_2^P = \Sigma_2^P \cap \Pi_2^P$. In light of the above containment theorems, however, we would like to present as much evidence as possible in this direction. One standard method of offering evidence that two classes are different is to demonstrate an oracle which separates them. In §6.5 we construct an oracle which separates S_2^P from $\Sigma_2^P \cap \Pi_2^P$ under the assumption that the machines involved are monotone. The framework we develop to build this oracle can be used to simplify the construction given by [BS79] of an oracle separating Σ_2^P and Π_2^P.

6.3 Previous Work

The original motivation for defining and studying the notion of symmetric alternation was a question posed by Uriel Feige. Independently, Ran Canetti has studied an alternative notion of symmetric alternation, and defined a class ϕ_2^P. He shows that $\mathbf{BPP} \subseteq \phi_2^P$ [Can95]. In fact, an easy argument shows that $\mathbf{S}_2^P = \phi_2^P$. [FST88] and [FKS94] study situations, in an interactive setting, where the provers do not have complete access to each other's strategies. As a step towards characterizing the class of languages accepted by such interactive proof systems we decided to formalize and study the associated non-interactive version.

Section 6.4 contains some definitions and our containment results; Section 6.5 contains the relativized result showing the separation between monotone $\mathbf{S}_2^P$ and $\Sigma_2^P \cap \Pi_2^P$; Section 6.6 contains some final remarks.

6.4 Containment Results

Σ is used to denote the alphabet and may be assumed to be $\{0,1\}$ without loss of generality. Throughout, the variable n denotes $|w|$, the length of the input in question. For $m \in \mathbb{N}$, we use $\exists^m x$ as shorthand for $\exists x(|x| = m)$, $\exists^m! x$ if such x is unique. $\forall^m x$ and $\forall^m! x$ is similarly used.

Definition 16 ($\mathbf{S}_2 \cdot \mathfrak{C}$) *For a complexity class $\mathfrak{C}$ we define $\mathbf{S}_2 \cdot \mathfrak{C}$ to be the complexity class consisting of those languages $\mathcal{L}$ for which there exists $C \in \mathfrak{C}$ and a polynomial q so that*

- $w \in \mathcal{L} \Rightarrow \exists^{q(n)}x, \forall^{q(n)}y, \langle w, x, y\rangle \in C$, *and*
- $w \notin \mathcal{L} \Rightarrow \exists^{q(n)}y, \forall^{q(n)}x, \langle w, x, y\rangle \notin C$.

Notice that the acceptance criteria for the $\mathbf{S}_2$ operator has the form of the acceptance criteria for Σ_2^P. The rejection criteria is similarly related to that of Π_2^P. $\mathbf{S}_2 \cdot \mathbf{P}$, then, is clearly inside $\Sigma_2^P \cap \Pi_2^P$. Notice that $\mathbf{S}_2^P$, like **BPP** and **IP**, is a *promise* class – the criteria for acceptance and rejection are not complements of each other.

Definition 17 ($\mathbf{S}_2^P$) $\mathbf{S}_2^P \stackrel{\text{def}}{=} \mathbf{S}_2 \cdot \mathbf{P}$.

Definition 18 ($\mathbf{S}_{2k}^P$) $\mathbf{S}_{2k}^P \stackrel{\text{def}}{=} \overbrace{\mathbf{S}_2 \cdot \mathbf{S}_2 \cdots \mathbf{S}_2}^{k} \cdot \mathbf{P}$.

Theorem 66 $\Sigma_1^P \bigcup \Pi_1^P \subset \mathbf{S}_2^P \subset \Sigma_2^P \bigcap \Pi_2^P$.

Proof: $\Sigma_1^P \subset \mathbf{S}_2^P \subset \Sigma_2^P$ and $\mathbf{S}_2^P$ is closed under complement.
□

Corollary 67 $\Sigma_k^P \bigcup \Pi_k^P \subset \mathbf{S}_{2k}^P \subset \Sigma_{2k}^P \bigcap \Pi_{k2}^P$.

This allows us to conclude that $\mathbf{PH} \stackrel{\text{def}}{=} \bigcup_k \Sigma_2^P = \bigcup_k \mathbf{S}_{2k}^P$, so that $\mathbf{S}_{2k}^P$ for $k = 1, 2 \ldots$ form a hierarchy which collapses if and only if the polynomial hierarchy collapses.

Theorem 68 $\mathbf{P}^{\mathbf{S}_2^P} \subset \mathbf{S}_2^P$.

Proof: Let $\mathfrak{S} \in \mathbf{S}_2^P$ and $S(\cdot, \cdot, \cdot)$ be a polynomial time machine accepting $\mathfrak{S}$ according to the definition of $\mathbf{S}_2^P$. Let $\mathcal{L} \in \mathbf{P}^{\mathfrak{S}}$ and let $D^{\mathfrak{S}}$ be a deterministic polynomial-time machine deciding $\mathcal{L}$. A *computation suggestion* of $D^{\mathfrak{S}}(w)$ consists of a sequence of pairs $(\mathfrak{d}_i, \mathfrak{a}_i)$ for $i = 1, \ldots, t$ so that

- each $\mathfrak{d}_i$ is an instantaneous description (see [HU79]) of $D^{\mathfrak{S}}$,
- $\mathfrak{d}_1$ is the initial instantaneous description of $D^{\mathfrak{S}}(w)$,
- $\mathfrak{d}_t$ is a final instantaneous description of $D^{\mathfrak{S}}$,
- if $\mathfrak{d}_i$ does not find $D^{\mathfrak{S}}$ in its query state, then $\mathfrak{d}_i \vdash_D \mathfrak{d}_{i+1}$,
- if $\mathfrak{d}_i$ finds $D^{\mathfrak{S}}$ querying q_i, then there is a response r_i so that $\mathfrak{d}_i \vdash_D \mathfrak{d}_{i+1}$ with response r_i and $\mathfrak{a}_i$ is a string of length appropriate for the quantified inputs of S on input w.

We construct a machine $T(\cdot, \cdot, \cdot)$ accepting $\mathcal{L}$ according to the definition of $\mathbf{S}_2^P$. $T(w, x, y)$ first examines x, rejecting unless x is a computation suggestion for $D^{\mathfrak{S}}$ so that $x = (\mathfrak{d}_i^x, \mathfrak{a}_i^x)_{i \le t_x}$. T then examines y, accepting unless y is a computation suggestion for $D^{\mathfrak{S}}$ (so that $y = (\mathfrak{d}_i^y, \mathfrak{a}_i^y)_{i \le t_y}$). If $t_x = t_y$ and $\forall i \in \{1, \ldots t_x\}$ we have that $\mathfrak{d}_i^x = \mathfrak{d}_i^y$ then T accepts exactly when $\mathfrak{d}_{t_x}^x = \mathfrak{d}_{t_y}^y$ is an accepting description. Otherwise there is i_0 so

that $\mathfrak{d}^x_{i_0} = \mathfrak{d}^y_{i_0}$ but $\mathfrak{d}^x_{i_0+1} \neq \mathfrak{d}^y_{i_0+1}$. Evidently, $\mathfrak{d}^x_{i_0}$ is a query state of $D^{\mathfrak{S}}$ (otherwise there is a unique next state, upon which both x and y must agree if they are computation suggestions). Let q_{i_0} be the query appearing in this description. Assume without loss of generality that the computation suggestion of x claims that $q_{i_0} \in \mathfrak{S}$. T then simulates $S(q_{i_0}, \mathfrak{a}^x_{i_0}, \mathfrak{a}^y_{i_0})$ accepting exactly when S accepts. Notice that for input w, there is a *correct* computation suggestion $\Delta_w = (\mathfrak{d}_i, \mathfrak{a}_i)_{i \leq t}$ (that is, one in which every oracle query is answered correctly) so that for each query q_i,

- if $q_i \in \mathfrak{S}$ then $\forall z, S(q_i, \mathfrak{a}_i, z)$ accepts, and
- if $q_i \notin \mathfrak{S}$ then $\forall z, S(q_i, z, \mathfrak{a}_i)$ rejects.

Then

- if $w \in \mathcal{L}$, then $\forall^{q(n)} y, T(w, \Delta_w, y)$ accepts, and
- if $w \notin \mathcal{L}$, then $\forall^{q(n)} x, T(w, x, \Delta_w)$ rejects,

as desired.

□

Corollary 69 $\Delta^P_2 = \mathbf{P}^{\mathbf{NP}} \subset \mathbf{P}^{\mathbf{S}^P_2} \subset \mathbf{S}^P_2$.

Corollary 70 $\Delta^P_{k+1} \subset \mathbf{S}^P_{2k}$.

The proof of Theorem 71 below is a generalization of the argument of Lautemann [Lau83].

Theorem 71 $\mathbf{S}_2 \cdot \mathbf{BP} \cdot \mathbf{P} \subset \mathbf{S}_2 \cdot \mathbf{P}$.

Proof: Let $\mathcal{L} \in \mathbf{S}_2 \cdot \mathbf{BP} \cdot \mathbf{P}$. Let $\mathcal{D} \in \mathbf{P}$ and q, r be polynomials such that

- $w \in \mathcal{L} \implies \exists^{q(n)} x, \forall^{q(n)} y, \Pr_{r \in_R \Sigma^{r(n)}}[\langle w, x, y, r\rangle \in \mathcal{D}] \geq 1 - 2^{-q(n)-n}$
- $w \notin \mathcal{L} \implies \exists^{q(n)} y, \forall^{q(n)} x, \Pr_{r \in_R \Sigma^{r(n)}}[\langle w, x, y, r\rangle \in \mathcal{D}] \leq 2^{-q(n)-n}$

Fix $w \in \Sigma^*$ and let $\hat{x} \in \Sigma^{q(n)}$ be so that for all $y \in \Sigma^{q(n)}$, $\Pr_{r \in_R \Sigma^{r(n)}}[\langle w, \hat{x}, y, r\rangle \in \mathcal{D}] \geq 1 - 2^{-q(n)-n}$. Let $\mathcal{W}_y \subset \Sigma^{r(n)}$ be the collection of random strings r for which $\langle w, \hat{x}, y, r\rangle \in \mathcal{D}$ and let $\mathcal{W} \stackrel{\text{def}}{=} \bigcap_y \mathcal{W}_y$ (these are the random strings r such that $\langle w, \hat{x}, y, r\rangle \in \mathcal{D}$ for all y). For a set B let $\mu(B)$ denote the measure of the set. Then $\forall y, \mu(\mathcal{W}_y) \geq 1 - 2^{-q(n)-n}$ so that $\mu(\mathcal{W}) \geq 1 - 2^{-n}$. As in [Lau83], we demonstrate that there exists a set $\{\sigma \stackrel{\text{def}}{=} \sigma_1, \ldots, \sigma_{r(n)}\}$ of elements of $\Sigma^{r(n)}$ so that for every $\tau \in \Sigma^{r(n)}$, there is some i so that $\sigma_i \oplus \tau \in \mathcal{W}$ ($\oplus$ stands for the binary operator that returns the bitwise XOR of the operands). Selecting $\sigma_1, \ldots, \sigma_{r(n)}$ uniformly and independently at random from $\Sigma^{r(n)}$, let $\mathcal{B}_\tau$ be the event that for each i, $\sigma_i \oplus \tau \notin \mathcal{W}$. Then $\Pr[\mathcal{B}_\tau] \leq 2^{-nr(n)}$ so that

$$\Pr[\bigvee_{\tau \in \Sigma^{r(n)}} \mathcal{B}_\tau] \leq \sum_\tau \Pr[\mathcal{B}_\tau] = 2^{r(n)} 2^{-nr(n)} = 2^{r(n)(1-n)} < 1.$$

Hence there is a set $\{\sigma = \sigma_1, \ldots, \sigma_{r(n)}\}$ so that for all τ, there is i so that $\sigma_i \oplus \tau \in \mathcal{W}$.

Suppose now that $w \notin \mathcal{L}$. There is then $\hat{y} \in \Sigma^{q(n)}$ so that for all $x \in \Sigma^{q(n)}$, $\Pr_r[\langle w, x, \hat{y}, r\rangle \in \mathcal{D}] \leq 2^{-q(n)-n}$. As before, let $\mathcal{W}_x \stackrel{\text{def}}{=} \{\, r \in \Sigma^{r(n)} \mid \langle w, x, \hat{y}, r\rangle \in \mathcal{D} \mid \}$. Define $\mathcal{W} \stackrel{\text{def}}{=} \bigcup_x \mathcal{W}_x$ so that $\mu(\mathcal{W}) \leq 2^{-n}$. Then we show that there is a set $\tau \stackrel{\text{def}}{=} \{\tau_1, \ldots, \tau_{r(n)^2}\}$ of elements of $\Sigma^{q(n)}$ so that for all $\sigma = \{\sigma_1, \ldots, \sigma_{r(n)}\}$, $\exists j, \forall i, \sigma_i \oplus \tau_j \notin \mathcal{W}$. Selecting $\tau_1, \ldots, \tau_{r(n)^2}$ independently and uniformly at random, let $\mathcal{B}_\sigma$ be the event that $\forall j, \exists i, \sigma_i \oplus \tau_j \in \mathcal{W}$. Then

$$\Pr[\mathcal{B}_\sigma] \leq \prod_j \sum_{i=1}^{r(n)} \Pr[\sigma_i \oplus \tau_j \in \mathcal{W}] = \prod_j \sum_{i=1}^{r(n)} 2^{-n} = \left(\frac{r(n)}{2^n}\right)^{r(n)^2}.$$

Hence,

$$\Pr[\bigvee_\sigma \mathcal{B}_\sigma] \leq \sum_\sigma \Pr[\mathcal{B}_\sigma] = 2^{r(n)^2} \left(\frac{r(n)}{2^n}\right)^{r(n)^2} < 1.$$

Therefore, there is a set $\{\tau_1, \ldots, \tau_{r(n)^2}\}$ with the property that for any set $\{\sigma_1, \ldots, \sigma_{r(n)}\}$, there is some j so that $\sigma_i \oplus \tau_j \notin \mathcal{W}$ for all i. In light of this, consider the deterministic polynomial time machine $M(\cdot, \cdot, \cdot)$ which, on input (w, α, β),

- checks the format of α, rejecting unless $\alpha = \langle x; \sigma_1, \ldots, \sigma_{r(n)}\rangle$,
- checks the format of β, accepting unless $\beta = \langle y; \tau_1, \ldots, \tau_{r(n)^2}\rangle$, and
- accepts iff for each τ_j, there is some σ_i so that $\langle w, x, y, \sigma_i \oplus \tau_j\rangle \in \mathcal{D}$.

From above, if $x \in \mathcal{L}$ then setting α to be the $\langle \hat{x}; \sigma_1, \ldots, \sigma_{r(n)}\rangle$ promised in the first part of the above discussion we have that D accepts regardless of β. Similarly, if $x \notin \mathcal{L}$ then setting β to be the $\langle \hat{y}; \tau_1, \ldots, \tau_{r(n)^2}\rangle$ promised in the second part of the above discussion, we have that D rejects regardless of α.

□

Corollary 72 $\mathbf{MA} \subset \mathbf{S}_2^P$.

Corollary 73 $\mathbf{BPP} \subset \mathbf{S}_2^P$

6.5 An Oracle Separating Monotone $\mathbf{S}_2^P$ and Monotone $\Sigma_2^P \cap \Pi_2^P$

Definition 19 *We shall call an oracle Turing machine* $M^O(\cdot, \ldots, \cdot)$ monotone *if for any inputs of the machine* $x_1, \ldots, x_n$,

$$O_1 \subseteq O_2 \wedge M^{O_1}(x_1, \ldots, x_n) \text{ accepts} \implies M^{O_2}(x_1, \ldots, x_n) \text{ accepts.}$$

We then define $\mathbf{mS}_2^{P,O}$ *to be the class of languages accepted by some monotone machine according to the* $\mathbf{S}_2^P$ *acceptance rules with oracle* O. $\mathbf{m\Sigma}_2^{P,O}$ *and* $\mathbf{m\Pi}_2^{P,O}$ *are defined similarly.*

The above conclusion that $\mathbf{BPP} \subset \mathbf{S}_2^P \subset \Sigma_2^P \cap \Pi_2^P$ is interesting commensurate with the extent to which we believe that the inclusion $\mathbf{S}_2^P \subset \Sigma_2^P \cap \Pi_2^P$ is strict. We offer evidence for the strictness of this inclusion by demonstration of an oracle O so that $\mathbf{mS}_2^{P,O} \subsetneq \mathbf{m}\Sigma_2^{P,O} \cap \mathbf{m}\Pi_2^{P,O}$.

We begin with some definitions relevant to our construction. For $n \geq 0$, consider subsets T of Σ^{2n} satisfying the Π_2^P predicate $\forall^n x \exists^n y, xy \in T$. This predicate is monotone. Collect together the minterms to form

$$\mathfrak{T} = \{T \subset \Sigma^{2n} \mid \forall^n x \exists^n ! y, xy \in T\}.$$

This set has size 2^{n2^n}. Given a family of minterms $\mathcal{T} \subset \mathfrak{T}$ and a set $W \subset \Sigma^{2n}$ we define

$$\mathcal{T}_W \stackrel{\text{def}}{=} \{T \in \mathcal{T} \mid W \subset T\}$$

which we shall call $\mathcal{T}$ *pinched at* W. A family $\mathcal{T} \subset \mathfrak{T}$ is said to be ϵ-*concentrated at* w if $\Pr_{T \in \mathfrak{T}}[w \in T] \geq \epsilon$. We shall say that $\mathcal{T} \neq \emptyset$ is ϵ-*diffuse on* S if for all $w \in S$, $\mathcal{T}$ is not ϵ-concentrated at w. A family $\mathcal{T}$ which is ϵ-concentrated at τ may be *pinched* at $\{\tau\}$, resulting in $\mathcal{T}_{\{\tau\}}$, with $|\mathcal{T}_{\{\tau\}}| \geq \epsilon|\mathcal{T}|$. A ϵ-*concentration sequence* for a family $\mathcal{T}$ is a sequence $\tau_1, \ldots, \tau_r$ such that

- $\mathcal{T}$ is ϵ-concentrated at τ_1,
- for $i \in \{1, \ldots, r-1\}$, $\mathcal{T}_{\{\tau_1, \ldots, \tau_i\}}$ is ϵ-concentrated at τ_{i+1},
- and $\mathcal{T}_{\{\tau_1, \ldots, \tau_r\}}$ is ϵ-diffuse on $\Sigma^{2n} - \{\tau_1, \ldots, \tau_r\}$.

Lemma 74 *Let $\tau_1, \ldots, \tau_r$ be an ϵ-concentration sequence for $\mathcal{T} \subset \mathfrak{T}$. Then*

$$r \leq \frac{-\log \mu(\mathcal{T})}{\log \epsilon + n}$$

where $\mu(\mathcal{T})$ is the density of $\mathcal{T}$ in $\mathfrak{T}$.

Proof: Let $\tau = \{\tau_1, \ldots, \tau_r\}$. Since $\mathcal{T} \subset \mathfrak{T}$, $\mathcal{T}_\tau \subset \mathfrak{T}_\tau$ so that

$$\epsilon^r \mu(\mathcal{T}) \leq \mu(\mathcal{T}_\tau) \leq \mu(\mathfrak{T}_\tau) = 2^{-nr}$$

and hence

$$r \leq \frac{-\log \mu(\mathcal{T})}{\log \epsilon + n}$$

as desired.

□

Theorem 75 *There exists an oracle O so that $\mathbf{mS}_2^{P,O} \subsetneq \mathbf{m}\Sigma_2^{P,O} \cap \mathbf{m}\Pi_2^{P,O}$.*

Proof: For a pair of oracles $O_1, O_2 \subset \Sigma^*$, define $O_1 \oplus O_2 \stackrel{\text{def}}{=} \{0x \mid x \in O_1\} \cup \{1y \mid y \in O_2\}$. Let $\mathcal{C}$ be the set of all oracles $O = O_1 \oplus O_2$ so that $\forall n \geq 0$,

$$\exists^n x \forall^n y, xy \in O_1 \iff \forall^n x \exists^n y, xy \in O_2.$$

Let $\mathcal{A}_n = \{C_0 \oplus C_1 \in \mathcal{C} \mid \exists^n x, \forall^n y, xy \in C_0\}$. Define $\mathcal{L}(O_1 \oplus O_2) = \{1^n \mid \exists^n x \forall^n y, xy \in O_1\}$ and notice that for $O \in \mathcal{C}$, we have that $\mathcal{L}(O) \in \mathbf{m}\Sigma_2^{P,O} \cap \mathbf{m}\Pi_2^{P,O}$. We shall construct an oracle $C = C_1 \oplus C_2 \in \mathcal{C}$ so that $\mathcal{L}(C) \notin \mathbf{m}S_2^{P,C}$. Let $\{D_i(\cdot,\cdot,\cdot)\}$ be an enumeration of monotone oracle-S_2^P machines so that for every O, $\mathbf{m}S_2^{P,O} = \{L(D_i^O)\}$. Define $Q_i(n)$ to be the maximum size, over all oracles, x values, and y values, of any query made by $D_i(1^n, x, y)$. C shall be constructed in stages $C_1 \subset C_2 \subset \cdots$ so that $C = \bigcup_i C_i$, stage i constructed to foil a specific monotone S_2^P machine. We shall have that

- for $i < j$, $C_i \subset C_j$ and $C_j - C_i \subset \Sigma^{k_j}$ for some k_j,
- $i < j \implies k_i < k_j$,
- for any oracle O with $O \bigcap \Sigma^{\leq k_s} = C_s$, $\mathcal{L}(O) \neq L(D_i^O)$ for any $i \leq s$.

Assume we have constructed the first $t-1$ stages, thus defining the oracle to length k_{t-1} and foiling the first $t-1$ machines. We shall construct C_t, foiling D_t. Let $p(n)$ be the running time of D_t. Select n so that $2n > k_{t-1}$, $2n > Q_{t-1}(k_{t-1})$, and $2^n > 2p(n)$. Set $k_t = 2n$. Assume, for contradiction, that regardless of our choice of $C_t \in \mathcal{C}$ (with $C_t \cap \Sigma^{\leq k_{t-1}} = C_{t-1}$), $D_t^{C_t}(1^n, \cdot, \cdot)$ accepts exactly when $C_t \in \mathcal{A}_n$.

For each $u \in \Sigma^n$, let $S_u \stackrel{\text{def}}{=} \{uv \mid |v| = n\}$ and consider the family of oracles

$$\{C_{t-1} \cup S_u \oplus T \mid T \in \mathfrak{T}\}.$$

Associate with each oracle O in this family an appropriate x so that $\forall y, D_t^O(1^n, x, y)$ accepts. There are at most $2^{p(n)}$ various values for x, so some x_u is associated with a fraction of this family of density at least $2^{-p(n)}$. Let $\mathcal{F}(u)$ be the (sub) family so associated with this x_u. Then define $\mathcal{T}(u) = \{T \mid C_{t-1} \cup S_u \oplus T \in \mathcal{F}_u\}$. Let $\tau(u)_1, \tau(u)_2, \ldots, \tau(u)_{t(u)}$ be a $p(n)^{-1}$-concentration sequence for $\mathcal{T}(u)$ and $\tau(u) = \{\tau(u)_1, \tau(u)_2, \ldots, \tau(u)_{t(u)}\}$. By Lemma 74 the length of this sequence, $t(u)$, is at most

$$\frac{-\log \mu(\mathcal{T}(u))}{\log p(n)^{-1} + n} \leq \frac{p(n)}{n - \log p(n)} \stackrel{(*)}{\leq} p(n)$$

where the second inequality follows because $2^n > 2p(n)$. Reiterating, for each $u \in \Sigma^n$, we have selected a family of minterms $\mathcal{F}(u)$ all associated with a certain x_u and a $p(n)^{-1}$-concentration sequence $\tau(u)$ for $\mathcal{T}(u)$.

We similarly construct such sets of maxterms. For a subset $X \subset \Sigma^{2n}$, let $\hat{X} \stackrel{\text{def}}{=} \Sigma^{2n} - X$ be the relative complement of X. For each $v \in \Sigma^n$, consider

$$\{C_{t-1} \cup \hat{T} \oplus \hat{S}_v \mid T \in \mathfrak{T}\}.$$

As above, let y_v be associated with a family of these maxterms $\mathcal{G}(v) \stackrel{\text{def}}{=} \{C_{t-1} \cup \hat{T} \oplus \hat{S}_v \mid T \in \mathcal{T}(v)\}$ so that $\mu(\mathcal{T}(v)) \geq 2^{-p(n)}$ (so that for any oracle O in this set and any x, $D_t^O(1^n, x, y_v)$ rejects). Let $\tau(v)_1, \ldots, \tau(v)_{t(v)}$ be a $p(n)^{-1}$-concentration sequence for $\mathcal{T}(v)$ and $\tau(v) = \{\tau(v)_1, \ldots, \tau(v)_{t(v)}\}$. Again $t(v) \leq p(n)$.

Selecting u and v uniformly and independently at random from Σ^n, we have that

$$\Pr_{u,v}[\exists i, \tau(u)_i \in S_v \text{ or } \exists j, \tau(v)_j \in S_u] \leq \frac{t(u) + t(v)}{2^n} \leq \frac{2p(n)}{2^n} < 1$$

so that there exists a pair $(\tilde{u}, \tilde{v})$ with the property that

- $\forall i, \tau(\tilde{u})_i \notin S_v$ and
- $\forall j, \tau(\tilde{v})_j \notin S_u$.

Notice that an oracle O selected from $\mathcal{F}(\tilde{u}) \cup \mathcal{G}(\tilde{v})$ is accepted by $D_t^O(1^n, x_{\tilde{u}}, y_{\tilde{v}})$ exactly when $O \in \mathcal{F}(\tilde{u})$. Let $M = \Sigma^{2n} - \tau(\tilde{v}) \oplus \tau(\tilde{u})$ and consider the behavior of $D_t^M(1^n, x_{\tilde{u}}, y_{\tilde{v}})$. Suppose that $D_t^M(1^n, x_{\tilde{u}}, y_{\tilde{v}})$ accepts. Since D is monotone, every oracle $O \in \{C_{t-1} \cup \hat{T} \oplus \hat{S}_{\tilde{v}} \mid T \in \mathcal{T}(\tilde{v})_{\tau(\tilde{v})}\}$ must disagree with M in one of the at most $p(n)$ places which $D_t^M(1^n, x_{\tilde{u}}, y_{\tilde{v}})$ queries, which contradicts that $\mathcal{T}(\tilde{v})_{\tau(\tilde{v})}$ is $p(n)^{-1}$-diffuse on $\Sigma^{2n} - \tau(\tilde{v})$. The case when $D_t^M(1^n, x_{\tilde{u}}, y_{\tilde{v}})$ rejects is handled dually.

□

6.6 Concluding Remarks

In this note we studied the notion of *symmetric alternation* by defining the complexity class $\mathbf{S}_2^P$. We observed certain structural properties of the $\mathbf{S}_2$ operator which allowed us to prove some interesting containment results. We were able to show that $\mathbf{BPP} \subset \mathbf{S}_2^P$ using a nontrivial variant of Lautemann's proof [Lau83].

It is an interesting open problem to construct an oracle separating $\mathbf{S}_2^P$ and $\Sigma_2^P \cap \Pi_2^P$. This would provide more evidence of the fact that the two classes are indeed different. The interactive version of symmetric alternation, the original motivation for this study, has not been characterized exactly and continues to remain an area of active study.

Bibliography

[ABP90] B. Awerbuch, A. Baratz, and D. Peleg. Cost-sensitive analysis of communication protocols. In *Proceedings of the 9th Symposium on Principles of Distributed Computing*, pages 177–187, 1990.

[ACPS93] S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *Journal of the ACM*, 40(5):1134–1164, 1993.

[AFMP94] E. M. Arkin, S. P. Fekete, J. S. B. Mitchell, and C. D. Piatko. Optimal covering tour problems. In *Proceedings of the 5th International Symposium on Algorithms and Computation*, 1994.

[AH91] E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. Technical Report 968, School of OR&IE, Cornell University, July 1991.

[AKR95] A. Agrawal, P. Klein, and R. Ravi. When trees collide: an approximation algorithm for the generalized steiner problem on networks. *SIAM Journal on Computing*, 24:440–456, 1995.

[AKS87] Miklós Ajtai, János Komlós, and E. Szemerédi. Deterministic simulation in LOGSPACE. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 132–140, New York City, 25–27 May 1987.

[ALM+92a] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *33rd Annual Symposium on Foundations of Computer Science*, pages 14–23, Pittsburgh, Pennsylvania, 24–27 October 1992. IEEE.

[ALM+92b] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 14–23, Pittsburgh, Pennsylvania, October 1992. IEEE.

[ALS91] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12:308–340, 1991.

[AR95] Y. Aumann and Y. Rabani. Improved bounds for all optical routing. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms*, pages 567–576. SIAM, 1995.

[Aro94] S. Arora. *Probabilistic checking of proofs and hardness of approximation problems*. PhD thesis, University of California, Berkeley, 1994.

[AS92a] N. Alon and J. H. Spencer. *The probabilistic method*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., first edition, 1992.

[AS92b] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 2–13, Pittsburgh, Pennsylvania, October 1992. IEEE.

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pages 421–429, Providence, Rhode Island, 6–8 May 1985.

[Bab92] L. Babai. Transparent proofs and limits to approximation. In A. Joseph, F. Mignot, F. Murat, B. Prum, and R. Rentschler, editors, *Proceedings of the first European Congress of Mathematics*, volume I of *Progress in Mathematics*, pages 31–91, Paris, France, July 1992. Birkhäuser Verlag.

[BFL90] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 16–25, St. Louis, Missouri, October 1990. IEEE.

[BFLS91] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 21–31, New Orleans, Louisiana, May 1991. ACM.

[BH92] R. A. Barry and P. A. Humblet. Bounds on the number of wavelengths needed in WDM networks. In *LEOS Summer Topical Meeting Digest*, pages 21–22, 1992.

[BH93] R. A. Barry and P. A. Humblet. Latin routers, design and implementation. *IEEE/OSA Journal of Lightwave Technology*, pages 891–899, May/June 1993.

[BLW87] M. W. Bern, E. L. Lawler, and A. L. Wong. Linear-time computation of optimal subgraphs of decomposable graphs. *Journal of Algorithms*, 8:216–235, 1987.

[Bod88] H. L. Bodlaender. Dynamic programming on graphs of bounded treewidth. In *Proceedings of the 15th International Colloquium on Automata Language and Programming*, volume 317 of *LNCS*, pages 105–118, 1988.

[Bod92] H. L. Bodlaender. A tourist guide through treewidth. Technical Report RUU-CS-92-12, Utrecht University, 1992.

[BOGKW88] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 113–131, Chicago, Illinois, May 1988. ACM.

[BRV96] A. Blum, R. Ravi, and S. Vempala. A constant-factor approximation algorithm for the k-MST problem. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computation*, 1996. To appear.

[BS79] Theodore P. Baker and Alan L. Selman. A second step toward the polynomial hierarchy. *Theoretical Computer Science*, 8:177–187, 1979.

[Can95] Ran Canetti. More on BPP and the polynomial-time hierarchy. Manuscript, April 1995.

[CB95] C. Chen and S. Banerjee. Optical switch configuration and lightpath assignment in wavelength routing multihop lightwave networks. In *Proceedings of INFOCOM*, pages 1300–1307. IEEE, 1995.

[CFLS93] A. Condon, J. Feigenbaum, C. Lund, and P. Shor. Probabilistically checkable debate systems and approximation algorithms for PSPACE-hard functions. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 305–314, San Diego, California, May 1993. ACM.

[CFLS94] A. Condon, J. Feigenbaum, C. Lund, and P. Shor. Random debaters and the hardness of approximting stochastic functions. In *Proceedings of the 9th Annual Conference on Structure in Complexity Theory*, pages 280–293, Amsterdam, The Netherlands, June 1994. IEEE.

[CG82] P. M. Camerini and G. Galbiati. The bounded path problem. *SIAM Journal on Algebraic and Discrete Methods*, 3(4):474–484, 1982.

[CGK92] I. Chlamtac, A. Ganz, and G. Karmi. Lightpath communications: An approach to high bandwidth optical WANs. *IEEE Transactions on Communication*, 40(7):1171–1182, July 1992.

[Cha77] R. Chandrasekaran. Minimum ratio spanning trees. *Networks*, 7:335–342, 1977.

[Cho91] C. H. Chow. On multicast path finding algorithms. In *Proceedings of IEEE INFOCOM*, pages 1274–1283, 1991.

[CK95] P. Crescenzi and V. Kann. A compendium of np-optimization problems. Manuscript, 1995.

[CKS81] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. of the Association for Computing Machinery*, 28(1):114–133, January 1981.

[CLR90a] T. H. Cormen, C. E. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, Boston, 1990.

[CLR90b] T. H. Cormen, C. E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, first edition, 1990. Thirteenth printing, 1994.

[CNW90] N. K. Cheung, K. Nosu, and G. Winzer, editors. *JSAC: Special Issue on Dense WDM Networks*, volume 8. IEEE, August 1990.

[Col84] C. J. Colbourn. The complexity of completing partial latin squares. *Discrete Applied Mathematics*, 8:25–30, 1984.

[Con88] A. Condon. *Computational models of games*. PhD thesis, University of Washington, 1988.

[Coo71] S. A. Cook. The complexity of theorem-proving procedures. In *Conference Record of Third Annual ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, Ohio, 3–5 1971 1971.

[CS89] J. T. Current and D. A. Schilling. The covering salesman problem. *Transportation Science*, 23:208–213, 1989.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[DK74] J. Denes and A. D. Keedwell. *Latin Squares and Their Applications*. Academic Press, Inc., New York, 1974.

[DK91] J. Denes and A. D. Keedwell. *Latin Squares: New Developments in the Theory and Applications*. Number 46 in Annals of Discrete Mathematics. North-Holland, 1991.

[Edm65] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[Eva60] T. Evans. Embedding incomplete latin squares. *American Mathematical Monthly*, 67:958–961, 1960.

[Fei96] U. Feige. A threshold of $\ln n$ for approximating set cover. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computation*, 1996. To appear.

[FGL+91a] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete (preliminary version). In *32nd Annual Symposium on Foundations of Computer Science*, pages 2–12, San Juan, Puerto Rico, 1–4 October 1991. IEEE.

[FGL+91b] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pages 2–12, San Juan, Puerto Rico, October 1991. IEEE.

[FK94] U. Feige and J. Kilian. Two prover protocols – low error at affordable rates. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 172–183, Montréal, Québec, Canada, May 1994. ACM.

[FKS94] J. Feigenbaum, D. Koller, and P. Shor, November 1994. Unpublished Manuscript.

[FKS95] J. Feigenbaum, D. Koller, and P. Shor. A game-theoretic classification of interactive complexity classes. In *Proceedings of the 10th Annual Conference on Structure in Complexity Theory*, pages 227–237, Minneapolis, Minnesota, June 1995. IEEE.

[FL92] U. Feige and L. Lovász. Two-prover one-round proof systems: Their power and their problems. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 733–744, Victoria, British Columbia, Canada, May 1992. ACM.

[FRS88] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. In *Proceedings of the 3rd Annual Conference on Structure in Complexity Theory*, pages 156–161, Georgetown University, Washington D.C., June 1988. IEEE.

[FST87] U. Feige, A. Shamir, and M. Tennenholtz. The noisy oracle model. In *Advances in Cryptology — CRYPTO'87*, pages 284–296, August 1987.

[FST88] U. Feige, A. Shamir, and M. Tennenholtz. The noisy oracle problem. In *Proceedings of CRYPTO 1988*, pages 284–296, 1988.

[FWB85] A. Frank, L. Wittie, and A. Bernstein. Multicast communication in network computers. *IEEE Software*, 2(3):49–61, 1985.

[GGP+94] M. X. Goemans, A. V. Goldberg, S. Plotkin, D. B. Shmoys, E. Tardos, and D. P. Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 1994.

[GGS95] J. L. Ganley, M. J. Golin, and J. S. Salowe. The multi-weighted spanning tree problem. In *Proceedings of the First Conference on Combinatorics and Computing*, LNCS, pages 141–150, 1995.

[Gil77] J. Gill. Computation complexity of probabilistic turing machines. *SIAM Journal of Computing*, 6(4):675–695, 1977.

[GJ79a] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman, San Francisco, 1979.

[GJ79b] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of* NP-*completeness*. W. H. Freeman and Company, first edition, 1979.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 291–304, Providence, Rhode Island, May 1985. ACM.

[Gol80] M. C. Golumbic. *Algorithmic graph theory and perfect graphs*. Academic Press, 1980.

[Gol94] O. Goldreich. Probabilistic proof systems (A survey). Technical Report TR94-008, Electronic Colloquium on Computational Complexity, 1994.

[GOT77] F. R. Giles, T. Oyama, and L. E. Trotter. On completing partial latin squares. In *Proceedings of the Eighth Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 523–543, 1977.

[Gre92] P. E. Green. *Fiber-Optic Networks*. Prentice-Hall, 1992.

[GW92] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 307–316, 1992.

[Hal48] M. Hall. Distinct representatives of subsets. *Bulletin of the American Mathematical Society*, 54:922–926, 1948.

[Has92] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, 1992.

[HK73] J. E. Hopcroft and R. M. Karp. An $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal of Computing*, 2(4):225–231, December 1973.

[HLCW89] J. Ho, D. T. Lee, C. H. Chang, and C. K. Wong. Bounded diameter spanning tree and related problems. In *Proceedings of the Annual ACM Symposium on Computational Geometry*, pages 276–282, 1989.

[Hoc95] D.S. Hochbaum, editor. *Approximation algorithms for NP-hard problems*. PWS Publishing Company, Boston, MA, 1995.

[HS65] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Series in Computer Science. Addison-Wesley, Reading, Massachusetts, 1979.

[IEE93] IEEE. *IEEE/OSA Journal of Lightwave Technology, special issue on Broad-Band Optical Network*, volume 11, May/June 1993.

[IK92] M. Irshid and M. Kavehrad. A WDM cross-connected star topology for multihop lightwave networks. *IEEE/OSA Journal of Lightwave Technology*, pages 828–835, June 1992.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 12–24, Seattle, Washington, 15–17 May 1989.

[JCGP+96] J.-C.Bermond, L. Gargano, S. Perennes, A. A. Rescigno, and U. Vaccaro. Efficient collective communication in optical networks. In *Proceedings of the 23rd International Colloquium on Automata, Languages, and Programming*, page To appear., 1996.

[Joh74a] D. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer System Sciences*, 9:256–278, 1974.

[Joh74b] D. Johnson. Fast algorithms for bin-packing. *Journal of Computer System Sciences*, 8:272–314, 1974.

[Joh88] D. S. Johnson. The NP-completeness column: An ongoing guide. *J. of Algorithms*, 9(3):426–444, September 1988.

[Joh92] D. S. Johnson. The NP-completeness column: An ongoing guide. *J. of Algorithms*, 13(3):502–524, September 1992.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[KJ83] B. Kadaba and J. Jaffe. Routing to multiple destinations in computer networks. *IEEE Transactions on Communications*, COM-31:343–351, March 1983.

[KLR+94] M. Kiwi, C. Lund, A. Russell, R. Sundaram, and D. Spielman. Alternation in interaction. In *Proceedings of the 9th Annual Conference on Structure in Complexity Theory*, pages 294–303, Amsterdam, The Netherlands, June 1994. IEEE.

[KP95] G. Kortsartz and D. Peleg. Approximation algorithms for minimum time broadcast. *SIAM Journal on Discrete Mathematics*, 8(3):401–427, 1995.

[KPP92a] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicasting for multimedia applications. In *Proceedings of IEEE INFOCOM*, May 1992.

[KPP92b] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Two distributed algorithms for the constrained steiner tree problem. Technical Report CAL-1005-92, Computer Systems Laboratory, University of California, San Diego, October 1992.

[KPP93] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Transactions on Networking*, pages 286–292, 1993.

[KPRS96] S. R. Kumar, R. Panigrahy, A. Russell, and R. Sundaram. A note on optical routing. Unpublished manuscript, 1996.

[KR95] P. N. Klein and R. Ravi. A nearly best-possible approximation for node-weighted steiner trees. *Journal of Algorithms*, 19:104–115, 1995.

[KRS96a] S. R. Kumar, A. Russell, and R. Sundaram. Approximating Latin square extensions. To appear in the Proceedings of the Second Annual International Computing and Combinatorics Conference, November 1996.

[KRS96b] S. R. Kumar, A. Russell, and R. Sundaram. Faster algorithms for optical switch configuration. Submitted to the IEEE Global Telecommunications Conference, 1996.

[Kru56] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the AMS*, 7(1):48–50, 1956.

[KRY95] S. Khuller, B. Raghavachari, and N. Young. Balancing minimum spanning and shortest path trees. *Algorithmica*, 14(4):305–321, 1995.

[KV94] S. Khuller and U. Vishkin. Biconnectivity approximations and graph carvings. *Journal of the ACM*, 41:214–235, 1994.

[Lau83] C. Lautemann. **BPP** and the polynomial hierarchy. *Information Processing Letters*, 17:215–217, 1983.

[LE77] C. C. Lindner and T. Evans. *Finite Embedding Theorems for Partial Designs and Algebras*. Les Presses de L'Universite de Montreal, Montreal, 1977.

[Len83] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[Lev73] L. Levin. Universal sequential search problems. *Problems of Information Transaction*, 9(3):265–266, 1973.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 2–10, St. Louis, Missouri, October 1990. IEEE.

[LL93] K. C. Lee and V. Li. Routing and switching in a wavelength convertible optical network. In *Proceedings of INFOCOM*, pages 578–585. IEEE, March 1993.

[LS91] D. Lapidot and A. Shamir. Fully parallelized multi prover protocols for NEXP-time. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pages 13–18, San Juan, Puerto Rico, October 1991. IEEE.

[Meg83] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM*, 30:852–865, 1983.

[MKR95] M. Mihail, C. Kaklamanis, and S. Rao. Efficient access to optical bandwidth. In *Proceedings of the 36th Annual IEEE Foundations of Computer Science*, pages 134–143, 1995.

[MRS+95] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III. Bicriteria network design problems. In *Proceedings of the International Colloquium on Automata, Languages and Programming*, number 944 in LNCS, pages 487–498, 1995.

[MRS96] M. Marathe, R. Ravi, and R. Sundaram. Service-constrained network design problems. In *Proceedings of the 5th Scandinavian Workshop on Algorithm Theory*, 1996.

[MS77] F. J. MacWilliams and N. J. A Sloane. *The theory of error-correcting codes*. Elsevier Science Publisher B.V., first edition, 1977. (Eight impression: 1993).

[NZ93] Noam Nisan and David Zuckerman. More deterministic simulation in logspace. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 235–244, San Diego, California, 16–18 May 1993.

[Phi93] C. Philips. The network inhibition problem. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 776–785, 1993.

[PR79] G. L. Peterson and J. H. Reif. Multiple-person alternation. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 348–363, San Juan, Puerto Rico, October 1979. IEEE.

[Pri57] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[Ram93] R. Ramaswami. Multi-wavelength lightwave networks for computer communication. *IEEE Communications Magazine*, 31(2):78–88, February 1993.

[Rav94] R. Ravi. Rapid rumor ramification: approximating the minimum broadcast time. In *Proceedings of the 35th Annual IEEE Foundations of Computer Science*, pages 202–213, 1994.

[Raz95] R. Raz. A parallel repetition theorem. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 447–456, Las Vegas, Nevada, June 1995. ACM.

[Rei79] John H. Reif. Complexity of the mover's problem and generalizations (extended abstract). In *20th Annual Symposium on Foundations of Computer Science*, pages 421–427, San Juan, Puerto Rico, 29–31 October 1979. IEEE.

[RG96] R. Ravi and M. X. Goemans. The constrained spanning tree problem. In *Proceedings of the 5th Scandanavian Workshop on Algorithmic Theory*, 1996. To appear.

[RMR+93] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III. Many birds with one stone: multi-objective approximation algorithms. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 438–447, 1993.

[Ros77] A. Rosa. Problem session. *Proceedings of the Second Caribbean Conference on Combinatorics and Computing*, 1977.

[RS86] N. Robertson and P. Seymour. Graph minors ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[RS94] R. Ramaswami and K. Sivarajan. Optimal routing and wavelength assignment in all-optical networks. In *Proceeding of INFOCOM*, pages 970–979. IEEE, June 1994.

[RS95] A. Russell and R. Sundaram. Symmetric alternation captures **BPP**. Technical Report MIT-LCS-TM-541, MIT, 1995.

[RSM+94] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi. Spanning trees short or small. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 546–555, 1994.

[RU94] P. Raghavan and E. Upfal. Efficient routing in all-optical networks. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computation*, pages 548–557, 1994.

[Sch86] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, first edition, 1986.

[Sha90] Adi Shamir. IP = PSPACE. In *31st Annual Symposium on Foundations of Computer Science*, volume I, pages 11–15, St. Louis, Missouri, 22–24 October 1990. IEEE.

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 330–335, Boston, Massachusetts, 25–27 April 1983.

[Sme81] B. Smetaniuk. A new construction of latin squares I. A proof of the Evans conjecture. *Ars Combinatorica*, 11:155–172, 1981.

[Sto76a] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, October 1976.

[Sto76b] L. J. Stockmeyer. The polynomial time hierarchy. *Theoretical Computer Science*, 3:1–22, 1976.

[Sud92] M. Sudan. *Efficient checking of polynomials and proofs and the hardness of approximation problems*. PhD thesis, University of California, Berkeley, May 1992.

[Tar83] R. E. Tarjan. *Data Structures and Network Algorithms*, volume 44 of *Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics, 1983.

[War87] A. Warburton. Approximation of Pareto optima in multiple-objective, shortest path problems. *Operations Research*, 35:70–79, 1987.

[WM91] O. Wolfson and A. Milo. The multicast policy and its relationship to replicated data placement. *ACM Transactions on Database Systems*, 16:181–205, 1991.

[Yao82] Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 November 1982. IEEE.

[ZA94] Z. Zhang and A. Acampora. A heuristic wavelength assignment algorithm for multihop WDM networks with wavelength routing and wavelength reuse. In *Proceedings of the INFOCOM*, pages 534–543. IEEE, June 1994.

[ZPD94] Q. Zhu, M. Parsa, and W. W. M. Dai. An iterative approach for delay-bounded minimum steiner tree construction. Technical Report UCSC-CRL-94-39, UC Santa Cruz, Santa Cruz, 1994.