ROUTING IN UNRELIABLE NETWORKS

by

Isidro Marcos Castiñeyra Figueredo

Ingeniero Electrónico Universidad Simón Bolívar
(1976)

S.M. Massachusetts Institute of Technology
(1980)

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, 1986

Signature of Author _____
Department of Electrical Engineering and Computer Science
May 20, 1986

Certified by _____
Professor Pierre A. Humblet
Thesis Supervisor

Accepted by _____
Professor Arthur C. Smith
Chairman, Departmental Commitee on Graduate Students

Routing in Unreliable Networks

by

Isidro Marcos Castiñeyra Figueredo

Submitted to the Department of Electrical Engineering and Computer Science on May 20, 1986, in partial fulfillment of the requirements for the Degree of Doctor of Philosophy.

Abstract

We consider strategies for selecting message routes in unreliable data-communication networks. We make the following assumptions: the nodes of the network are perfectly reliable; the links are unreliable, and for each link l there is a probability p_l, where $0 < p_l < 1$ is a rational number , that the link is operative; links fail independently of each other. We focus on approaches that use two transmission paths simultaneously to transfer a message, maximizing the probability that at least one of the copies of the message arrives at the destination node. The intent is to trade increased use of network resources for a lowered transmission delay. We consider different versions of this problem: in one version one of the paths is fixed, and we are asked to find a second one that results in maximum message reception probability; in another version we look for two link-disjoint path that maximize the same objective as before; in the third version we dispense with the disjointedness restriction of the previous problem. Complexity results are given for these problems and solution techniques proposed.

Thesis Supervisor: Pierre A. Humblet
Title: Associate Professor of Electrical Engineering

Acknowledgements

I thank professor Pierre Humblet, my thesis supervisor, for his guidance and support throughout my graduate studies.

I thank professors Robert Gallager and Sanjoy Mitter, my thesis readers, for their interest and advice.

I thank most especially Ellen Hahne.

I thank my officemates Erdal Arikan, Julio Escobar, Eli Gafni, Jeannine Mosely and Edward Tiedemann.

I thank my fellow L.I.D.S. inmates Utpal Mukherji, Jerry Prince, Jean Régnier and John Spinelli.

I thank my friends María Felicia Canto, Jeff Gerecht, Emilio Giráldez and Clint Roth.

I thank Richard Stroud and Robert Street, instructors for the M.I.T. Aikido Club.

I thank my fellow aikidoka John Aspinall, Andrew Crane, Edoardo Cunha, Gordon Hannah, Warren Krueger, Ralph Muha, and Trey Peck.

Ésta va dedicada a Isidro, Yuya, Gloria, Abel y Tati.

Da steh' ich nun, ich armer Tor

Und bin so klug als wie zuvor.

Goethe's Faust, lines 358 and 359.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Unreliable Networks

By a data-communication network we mean a collection of geographically distant computers (nodes) that can communicate with each other by exchanging messages via communication channels (links). From our point of view the purpose of a data-communication network is to transport messages reliably between its component nodes. Most conventional wire-networks have highly reliable links. In this kind of network the usual approach to reliable message transfer is to include in each message enough redundancy to make error-*detection* possible. Typically the *polynomial code*, also known as cyclic redundancy code or **CRC**, is used. This code satisfies the standard reliability-enhancing requirements, and is easily implementable in hardware, see [17] for more details. Upon detection of an error most link-level transmission protocols request that the message be re-sent. Typical of this class of

protocols is **HDLC** (High-level Data Link Control) used in CCITT's X.25 network interface standard. The net effect of an error is to add to the transmission delay, since one has to wait until the message is retransmitted. Also, the retransmitted copy of the message shows up as an increase in the total load of the network. If the bit-error rate were to increase, the transmission delay might become unsatisfactory. In this contingency we need to seek different means to achieve reliable communication. There are other types of network, e.g. packet radio networks, that are inherently unreliable and would benefit from techniques designed to improve message transmission reliability.

In this thesis we focus on approaches that use two or more transmission paths simultaneously to transfer a message. We intend to trade increased use of network resources, two or more paths instead of one, for a lowered transmission delay. We can argue in favor of this approach, with its consequent reduction in total throughput, by saying that when using the more conventional approach an error results in the repeated use of the same path for the same message. Thus, with a 'high' bit error rate, in a network that uses the standard approach, it is likely that, in the end, we will use extra resources anyhow.

The extreme case of multiple path transmission is flooding the network with copies of the message. The message is routed so that, in the case of no failures, a copy of the message is sent along every link of the network. This is maximum network resource consumption. We are looking for intermediate strategies between the extremes of sending a single copy of the message and flooding the network with many copies of it.

The choice of the specific approach to be used depends on exactly how link-failures behave. When we talk about link-failures, we do not necessarily mean the actual misfunction of the hardware, but also any condition that results in an unsuccessful transmission. Here we are mostly concerned about the correlation between the outcomes, whether failures or successes, of consecutive attempts at transmission. In other words, if a link fails, for how long does it stay 'down'? In this thesis we assume that the status of a link, i.e. whether 'up' or 'down', changes so fast that keeping track of it would be pointless. The delay formed by the time it takes to determine a link's status plus the time necessary to disseminate that datum is sufficiently large that the received information is already obsolete. This is because the information in question is received from the network itself.

A packet radio network with mobile nodes fits these characteristics. In this scenario, the mobility of the transmitters and the fragility of the medium conspire to produce what we can safely call an 'unreliable' network. More so, if we consider the possibility that the network might find itself under attack, an external agency might try to prevent successful communication, by jamming, say.

We next mention some approaches used to deal with node and link failures in the more usually considered case when the network state remains stable for long enough to make possible the use of knowledge about actual network configuration. Strategies proposed range from the very dynamic to the completely static. An example of the former type is the one used in the ARPANET in which traffic is rerouted in an attempt to use low delay routes and avoid links with long queueing delays. An example of the latter is the one proposed for the IBM network in which for each source-destination pair there is a list of allowed routes. When a link failure

is detected all routes utilizing this link are deleted from the allowed route lists. S. G. Finn proposes in [4] a class of network synchronization procedures, called *Resynch Procedures.* These are mechanism for bringing all nodes of a distributed network to a known state simultaneously, despite arbitrary finite delays between nodes. The resynch procedures are used to implement a network protocol that can guarantee that no packets will be lost and no duplicate packets will be inadvertently received, despite arbitrary node and link failures. J. A. Roskind in his doctoral thesis, see [13], proposes the use of precomputed disjoint spanning trees to transmit information about topological changes. J.M. Spinelli in his Master's thesis, see [15], presents an algorithm to maintain a correct view of the network topology without transmitting any information other than the operational status of links.

This thesis examines some approaches for choosing routes for two copies of a message. We focus on maximizing the probability that at least one of the copies arrives at the destination node. We ask how hard is it to calculate these routes. We conclude that looking for a pair of routes that satisfy this very natural criterion is a computationally intractable problem.

The rest of this chapter is as follows. In section 1.2 we define probabilistic networks. This is the basic model used in this thesis. In section 1.3 we give an overview of related work to be found in the literature. In section 1.4 we consider alternative ways to use several paths simultaneously for transmitting one message. Section 1.5 gives an overview of this thesis including a summary of results.

1.2 Probabilistic Networks

A directed network is a tuple (N, L), where N is a finite set of nodes and $L \subseteq N \times N$ is the set of links of the network. Notice that this definition excludes the existence of more than one link between any two nodes. A link of the network is denoted either by a single letter, e.g. l, or by the ordered pair of nodes connected by the link, e.g. $\langle i, j \rangle$, we assume that the network includes no self-loops, i.e. there is no link of the form $\langle i, i \rangle$ in the network. In the links of a directed network, traffic can flow only in the direction given by the ordering of the nodes, i.e. in link $\langle i, j \rangle$ we can have traffic going from node i to node j, but not in the opposite direction. The network includes two distinguished nodes, nodes s and t. Node s is the *origin* node, node t the *destination.* In a similar manner we can define an *undirected* network in such a network a link is an *unordered* pair, e.g. (i, j), traffic can flow in both directions of an undirected link. In what follows we shall be referring to directed networks unless explicitly stated.

A simple path joining nodes s and t is a sequence of nodes $s, n_1, n_2, \ldots, n_m, t$, such that the links $\langle s, n_1 \rangle$; $\langle n_i, n_{i+1} \rangle$ for $i = 1, \ldots, m-1$; and $\langle n_m, t \rangle$ are all members of L, and such that no node in this sequence is repeated. I.e. the path does not fold onto itself.

To each link $l \in L$ we associate a *rational* number $0 \leq p_l \leq 1$, the probability that link l is operative. Links fail independently of each other. This independence assumption is unrealistic but necessary, we shall see that even with its inclusion many questions pertaining to probabilistic networks are very difficult to answer. If we give up this assumption it can be shown that finding *one* simple path, P,

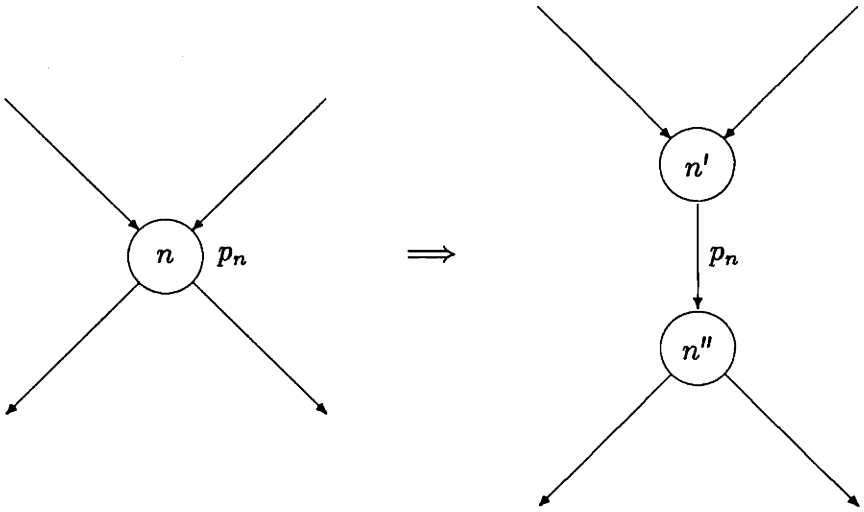


Figure 1.1: Transforming an Unreliable Node

connecting nodes s and t in a given probabilistic network such that the probability that P is operative is greater or equal than a given rational threshold is an *NP*-complete problem. An intermediate approach between assuming independence or assumming unrestricted dependencies would be to consider locally correlated failures, i.e. if a link has failed it is likely that 'nearby' links have failed too. We have not pursued this. We shall assume that the nodes of the network are perfectly reliable. This implies no loss of generality because any unreliability in the nodes of the network can be modelled by unreliable links. To wit, see figure 1.1 given an unreliable node n, and given p_n, the probability that node n is operative, one can separate node n into two nodes, n' and n'', joined by a link l with operation probability equal to p_n. Make all links incoming into n arrive at n', and all outgoing links depart n''. The two versions are equivalent.

A path P connecting nodes s and t is said to be operative if every link $l \in P$ is operative. Given a path P, $\mathcal{P}(P)$ the probability that this path is operative is

given by

$$\mathcal{P}(P) = \prod_{l \in P} p_l. \tag{1.1}$$

We consider only simple paths because for any non-simple path P, a path that loops onto itself, there exists a simple path $P_s \subset P$ such that $\mathcal{P}(P_s) \geq \mathcal{P}(P)$. The main questions concerning probabilistic networks deal with the existence of operative simple s-t-paths. We next give an overview of these questions and the related results to be found in the literature.

1.3 Related Problems

The problem of determining "network reliability", i.e. the probability that there is a path connecting node s to every other node in the network has been proved to be $\#P$-complete, to be defined below, also see [11]. However, if the graph is directed and acyclic, network reliability can be computed in polynomial time, see [1]. The $\#P$-completeness result holds for directed and undirected networks. Approximating the reliability to a given $\epsilon > 0$ is also a $\#P$-complete problem, see [11]. The category of $\#P$-complete (read "number P") problems was proposed in [18]. It includes many enumeration problems, i.e. "given a problem, how *many* solutions are there?". For example, associated with the **Satisfiability** problem is the enumeration problem of counting how many truth assignments satisfy simultaneously the clauses of the problem. This category of problems is 'harder' than 'plain' NP-complete problems. It is interesting that there are $\#P$-complete problems for which the associated search problem can be solved in polynomial time. For more

on $\#P$-completeness see [6].

Similar results exist for the "s-t reliability" problem where we are given two nodes s and t and asked what is the probability that there is an operative path between them. Valiant has proven that this problem is $\#P$-complete for general directed or undirected graphs, see [19]. There are algorithms to calculate s-t reliability in time proportional with $|N|$ and the number of minimal s-t cuts in G. This number can be very large, exponential in $|N|$, but need not be so, see [12].

The related problem of finding a most reliable path in a probabilistic network has a polynomial time algorithm. The algorithm is equivalent to calculating a shortest path between the nodes s and t with the length of a link given by $-\log p_l$, see [9].

Another related problem is that of finding the maximum number of disjoint bounded s-t paths, bounded in the sense that the probability that each path is operative should be greater or equal than a given threshold. Not only this, but also the problem of determining if *two* disjoint bounded paths exist have been shown to be NP-complete problems, see [7]. Contrastingly the problem of finding K disjoint paths such that the probability that *all* of them are operative is maximized has a polynomial time algorithm, see [16]. This optimization is done by a minimum cost flow algorithm.

1.4 Alternative Approaches

Given the basic idea of using multiple paths, the actual implementation can be done in many different ways. The main question is, how are the paths going to be chosen? A related issue is how to characterize the reliability of the links of the network.

Given two nodes, once we have chosen several paths that connect them, these paths can be used in different ways. The simplest approach would be to send identical copies of the message we want to transmit along each one of the selected paths. Alternatively, we could map the message, by a suitable coding scheme, into a relatively large number of 'micro-packets'. The coding would ensure that the recovery of a given fraction of these μ-packets, say eighty of a hundred, would enable us to reconstruct the original message. Here we shall call such a coding scheme a 'holographic' code. Each path would carry some fraction of the μ-packets. The former procedure, identical copies, would be preferable in a situation in which there is some correlation between the failure or succes of succesive attempts at transmission along network links. The latter seems more suited when we can assume that the outcome of these trials are independent random variables. In the rest of this section, we go briefly over alternative approaches suggested by the basic multiple-path transmission idea.

1.4.1 Modeling Reliability with Probabilistic Networks

The most common way to quantify the reliability of a link is to associate with it a probability. The most basic problem this setup suggests is to look for two paths, P_1 and P_2, such that the probability that at least one of them is operative is maximized. That is, we are trying to maximize

$$\prod_{\{l:l\in P_1\}} p_l + \prod_{\{l:l\in P_2\}} p_l - \prod_{\{l:l\in P_1\cup P_2\}} p_l.$$

We obtain variants of this problem by including further restrictions on the paths, e.g. we could ask that P_1 and P_2 should be disjoint, i.e. they should have no links in common. In the same vein, one could look for disjoint paths P_1 and P_2 such that the probability that *both* of them are operative is maximized. That is, we are maximizing

$$\prod_{\{l:l\in P_1\cup P_2\}} p_l.$$

This last formulation has the advantage that, after the appropriate transformation, the problem can be solved by existing special purpose efficient algorithms, see [16]. The problem is transformed into a minimum cost flow problem.

A different approach is to fix one path P_1 and look for a path P_2, not necessarily disjoint from the first, such that the probability that at least one of the two paths is operative is maximized. We say that P_2 complements P_1. This procedure can be iterated; after finding P_2, take it as the fixed path and then find P_3 complementing P_2. This can be repeated until we are satisfied with the answer, or until we feel that the progress being made is not worth the effort.

1.4.2 Modeling Reliability with Additive Costs

An alternative, and more heuristic, approach is to assign to each link l coefficients $c_l^1, c_l^2, \ldots, c_l^m$, where m is the number of paths $P_1, P_2, \ldots, P_m$, that we are looking for. The number $\sum_{j=1}^{i} c_l^j$ corresponds to the cost incurred when i paths use link l. The c_l^i are chosen to reflect the reliability of link l; the more unreliable a link, the higher the cost associated with its use. We want paths $P_1, \ldots, P_m$, that minimize the total cost. That is, m paths minimizing

$$\sum_{l} \sum_{j=1}^{i(l)} c_l^j.$$

Here $i(l)$ is the number of paths that use node l.

If we define $c_l^1 = -\ln p_l$, then a set of minimum cost paths P_i that happen to be disjoint will maximize the probability that all the P_i are operative.

If the c_l^i satisfy $0 < c_l^1 < c_l^2 < \ldots < c_l^m$, the optimum can be found by casting the problem as a minimum cost flow problem. To each link $l = \langle i, j \rangle$ correspond m parallel links, i.e. these m links go between nodes i and j, $l^1, l^2, \ldots, l^m$, with associated costs $c_l^1, c_l^2, \ldots, c_l^m$ and unit capacity. We seek a minimum cost flow of m units between nodes s and t. Viewing the problem in these terms allows us to use special purpose efficient algorithms. Also, the matrix associated with the capacity and flow conservation constraints of this problem is unimodular. This guarantees that if the problem is feasible there will be an optimal integer solution.

1.4.3 Interpreting the Probabilites as Loss Coefficients

Another approach is to interpret the probabilities p_l as 'transmission coefficients'. That is, if a link l has the entering flow of f_l^{in}, the flow coming out would be given by $f_l^{\text{out}} = p_l f_l^{\text{in}}$. To each link we assign unit capacity. Given that there are m units of flow entering s, we want to find a feasible flow that will maximize the flow entering node t. A feasible flow is such that for each link l, $f_l^{\text{in}} \leq 1$, and for each node n the flow entering it is equal to the flow coming out, i.e.

$$\sum_{\{l:l \text{ enters } n\}} p_l f_l^{\text{in}} = \sum_{\{l:l \text{ leaves } n\}} f_l^{\text{in}}. \tag{1.2}$$

This specifies a linear program. Notice that the solution to this problem is not guaranteed to be integer. The expression

$$\alpha_l = \frac{f_l^{\text{in}}}{\sum_{\{l:l \text{ leaves } n\}} f_l^{\text{in}}} \tag{1.3}$$

can be interpreted as a relative routing variable for outgoing link l. That is, of all the flow entering node n a fraction of it given by α_l is routed along link l.

This scheme is compatible with the 'holographic' coding, since the subdivision of a message into μ-packets is better adapted to the fractional routing described. In this case the capacity of a link would be set to the number of μ-packets a message is mapped into.

From a probabilistic point of view this approach is equivalent to maximizing the expected number of μ-packets arriving at the destination node.

We can observe that in a scheme as the above, maximum flow in a network with losses plus 'holographic' coding, each μ-packet should contain enough redundancy to allow the detection of transmission errors. The link protocol would most likely be designed not to ask for μ-packet retransmission when a link transmission error is detected; rather retransmission would be provided on an end to end basis using time-outs.

1.5 Summary of Results

The underlying question we are addressing is: what can be done when we do not really know what is happening in the network? Most published routing algorithms assume accurate knowledge of the conditions in the network: topology, and traffic matrices are taken as known; the message arrival processes are often considered to be Poisson, and so on. In practice the situation is different. We need ways to deal with data-communication networks when we do not have exact knowledge of their status. We need to decide how to represent the uncertainty: one could, for example, assume a probability density on the inputs, or one could take that it is only known that the inputs are inside a given range. Corresponding to the former approach, probabilistic weighting of inputs, the logical next step would be to optimize the expected value of an appropiate objective function, including, perhaps, restrictions on variance. Corresponding to the latter approach, the input within a range, a min-max optimization could be attempted.

This thesis is an exploratory foray in the direction expressed above. We look at a basic problem: what to do when we do not know the topology of the net-

work. More precisely, what can be done when our knowledge about any link of the network is the probability that this link is operative. We are looking for a *robust* routing scheme, i.e. a scheme that would produce satisfactory results in the face of uncertainty. The essence of most attempts at designing robust systems, as for example in control theory, is that we need to overdesign. We need to know how to overdesign intelligently, i.e. how to get the most, where the meaning of 'most' has yet to be agreed upon, from the additional expenses.

In the rest of this section we outline the rest of this thesis.

Chapter Two: Complementary Path Problem

The problem considered in this chapter is as follows. Given nodes s and t, given P_1, a path that connects these two nodes; find P_2, a path also connecting s and t, not necessarily disjoint from P_1, such that the probability that at least one of these two paths is operative is maximized. We prove that for directed or undirected networks this problem is *NP*-complete. The proof is a reduction to Partition, and applies to the restricted case of directed acyclic networks. We also give a pseudo-polynomial algorithm for the solution of this problem.

Chapter Three: Routing Two Messages Along Disjoint Paths

We consider the following problem. Given nodes s and t, find P_1 and P_2, two *disjoint* paths joining nodes s and t, such that the probability that at least one of the two paths is operative is maximized. We show that this problem is *NP*-complete

for general directed networks. The problem is shown to be *strongly NP*-complete if we allow the possibility $p_l = 1$, or if the probabilities are given in logarithmic format, i.e. $p_l = a^{n_l}$ for rational $0 < a < 1$ and positive integers n_l. We give a pseudo-polynomial algorithm for acyclic networks. It is not known if the problem is *NP*-complete for acyclic graphs. The related problem of finding P_1 and P_2, two disjoints paths joinng nodes s and t, such that the *expected number* of operative paths is maximized is also shown to be *NP*-complete.

Chapter Four: Routing Two Messages Along Non-disjoint Paths

We address the same problem considered in chapter three, with the difference that the path-disjointedness restriction is dispensed with. The problem is shown to be *NP*-complete, even in the case of an acyclic network. We give a pseudo-polynomial algorithm for acyclic networks. It is not known if the case of a general network, a network that might contain cycles, results in a strongly *NP*-complete problem.

The fifth chapter contains the conclusions and suggestions for further research.

Chapter 2

Finding a Complementary Path

2.1 Introduction

Onee approach to routing two copies of a message in a data-communication network is to fix the route of one of the two copies. The second copy is then routed to optimize a given criterion that is a function of both copies's routes. In this thesis the criterion to be optimized, maximized, is the probability that at least one of the two copies reaches their destination. The first copy might be sent, for example, along a most reliable path of the network, or through any chosen path that the specific situation suggests. The paths followed by these two copies need not be disjoint. If they are, then the optimal route for the second copy is a most reliable path in the network obtained by removing from the original network the links belonging to the path followed by the first copy.

The failure of a link that is shared by both copies's paths is catastrophic but,

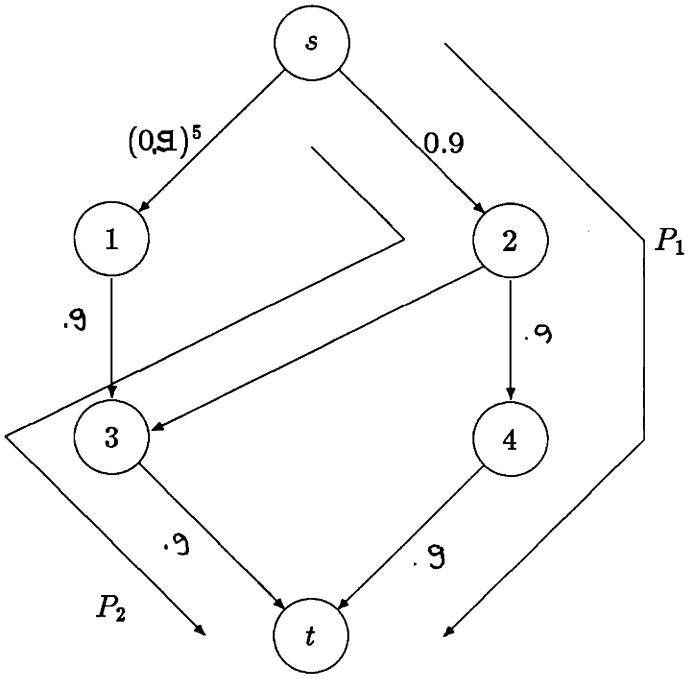


Figure 2.1: An example where sharing links is indicated

as in the case of a very reliable link, sharing might be preferrable to using an alternative unreliable link also such a link need not exist. The network of figure 2.1 presents such a case, the number labeling each link in that figure is the probability that the link is operative. Given P_1, the fixed path, as in the figure, the optimum P_2, the path followed by the second copy, shares the link going between node s and node 2. This link is much more reliable than the only alternative link that connects node s to node 1.

The problem described in the previous paragraph is one of the simplest that can be proposed under the idea of sending multiple copies of a message. We would like

to obtain an efficient, i.e. polynomial, algorithm for its solution. Unfortunately in this chapter we show by a reduction to Partition that this problem is NP-complete. Therefore a polynomial algorithm is ruled out under the assumption $P \neq NP$. We present a pseudo-polynomial algorithm to pick the routing for the second copy given a fixed route to be followed by the first copy. The algorithm is an adaptation of that given in [9] to find a shortest length path with time constrains.

In section 2.2 the problem is stated more precisely. In section 2.3 we give the NP-completeness proof. In section 2.4 we consider a variation of the problem in which a different format is used to represent the probabilities. In section 2.5 we obtain an equivalent problem. The algorithm is developed in section 2.6.

2.2 Problem Statement

In this section we define precisely the problem introduced in the previous section.

Complementary Path Problem

INSTANCE: Given $G = (N, L)$ a directed graph, where N is the set of nodes of the graph, and $L \subseteq N \times N$ is the set of links. For each link $l \in L$ we are given a rational number p_l, $0 \leq p_l \leq 1$, the probability that link l will be operative. Also given is p_{th} a rational threshold, and $P_1 \subset L$ a simple directed path in G connecting nodes s and t.

QUESTION: Is there P_2, another simple directed path that connects nodes s and t, such that the probability that at least one of these two paths is operative is larger than p_{th}? A path is operative if every link in that path is operative. The paths P_1

and P_2 need not be disjoint. More precisely, is there a P_2 so that

$$\mathcal{P}(P_1, P_2) = \left(\prod_{l \in P_1} p_l + \prod_{l \in P_2} p_l - \prod_{l \in P_1 \cup P_2} p_l \right) \geq p_{th} \tag{2.4}$$

A P_2 that maximizes $\mathcal{P}(P_1, P_2)$ is said to *complement* P_1.

2.3 The Complementary Path Problem is *NP*-Complete

In this section we prove that the complementary path problem is *NP*-complete. This is done by giving a construction that results in a reduction to partition. Next we define the partition problem.

Partition

INSTANCE: Given S a finite set of positive integers, $S = \{c_1, c_2, \ldots, c_m\}$.

QUESTION: Is there a subset $A \subset S$ such that

$$\sum_{i \in A} c_i = \sum_{i \in S \setminus A} c_i ? \tag{2.5}$$

The sets A and $S \setminus A$ constitute a partition of S. A reduction to partition is often used to show *NP*-hardness of problems in whose statement, loosely said, many numbers, usually positive integers, appear. For example, the well known Knapsack problem is easily shown to be *NP*-hard by a reduction to Partition the numbers

appearing when specifying an instance of the Knapsack problem are the weights and values of the elements. Partition is not a strongly *NP*-complete problem i.e. it can be solved in pseudo-polynomial time by dynamic programming. Therefore, for any problem shown to be *NP*-hard by a reduction to partition it is possible that there exists a pseudo-polynomial time algorithm for its solution. In fact, this is the case for the complementary path problem, in section 2.6 we offer such an algorithm. Partition appeared in the original list of 21 *NP*-complete problems published by Karp in 1972, see [8].

2.3.1 The Reduction

Theorem 2.1 *The complementary path problem is NP-complete.*

Proof. The problem is easily seen to be in *NP*, i.e. for any given pair of paths P_1, P_2, and threshold p_{th}, it can be checked in polynomial time whether $\mathcal{P}(P_1, P_2) \geq p_{th}$. Consider the network shown in figure 2.2, with P_1 as shown. P_1 includes all links $l_j^0, j = 1, \ldots, m$. Define $p_{l_j^0} = h_j$, $p_{l_j^1} = t_j$, let all other links in the network be perfectly reliable. A path P_2 connecting node s to node t includes one and only one of either l_j^0 or l_j^1, $j = 1, \ldots, m$. Given P_2 define $T = \{j : l_j^1 \in P_2\}$, $H = \{j : l_j^0 \in P_2\}$. Notice that $T \cup H = \{1, \ldots, m\}$, and $T \cap H = \emptyset$. With these definitions we have

$$\begin{aligned} \mathcal{P}(P_1, P_2) &= \prod_{i=1}^{m} h_i + \prod_{i \in T} t_i \prod_{i \in H} h_i - \prod_{i=1}^{m} h_i \prod_{i \in T} t_i \\ &= \prod_{i=1}^{m} h_i \left(1 + \prod_{i \in T} \left(\frac{t_i}{h_i} \right) - \prod_{i \in T} t_i \right). \end{aligned} \tag{2.6}$$

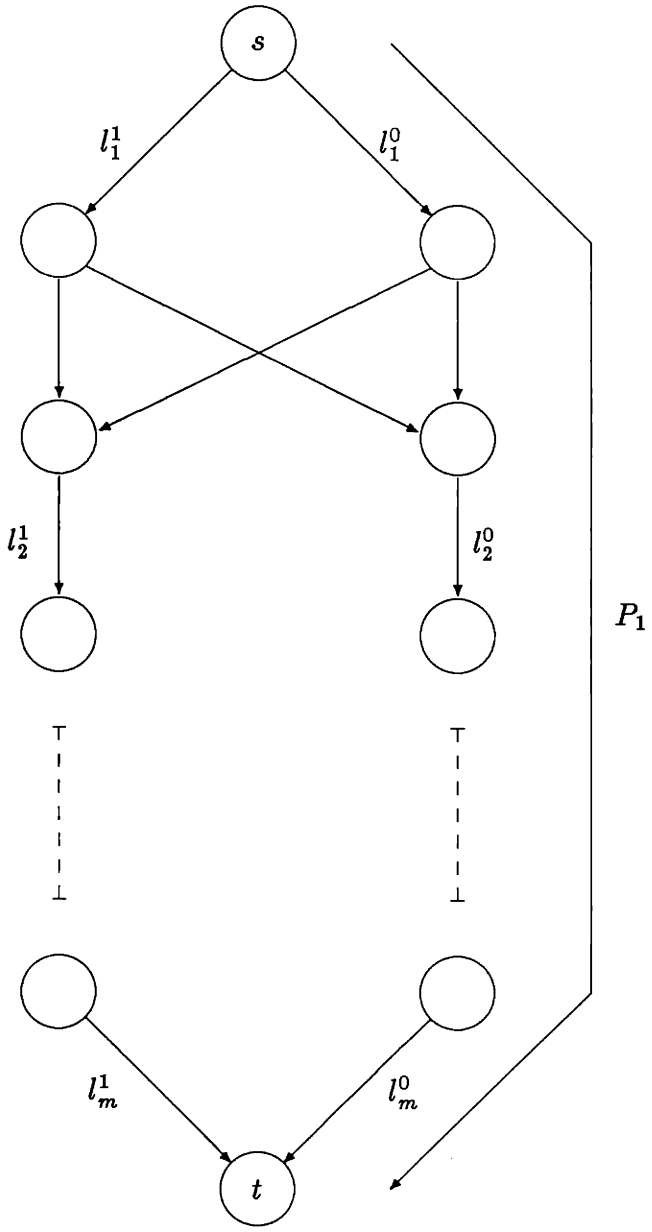


Figure 2.2: Network used to reduce the Complementary Path Problem to Partition.

Take now $t_i = (h_i)^2$. We then have

$$\mathcal{P}(P_1, P_2) = \prod_{i=1}^{m} h_i \left(1 + \prod_{i \in T} h_i - \prod_{i \in T} h_i^2\right). \tag{2.7}$$

Defining now $b = \prod_{i=1}^{m} h_i$ and $f(x) = x - x^2$ we have

$$\mathcal{P}(P_1, P_2) = b\left(1 + f\left(\prod_{i \in T} h_i\right)\right). \tag{2.8}$$

Maximizing $\mathcal{P}(P_1, P_2)$ is equivalent to finding $T \subseteq \{1, \ldots, m\}$ that maximizes $f\left(\prod_{i \in T} h_i\right)$. $f(x)$ is a continuous, concave function, its maximum is located at $x = 1/2$, with $f(1/2) = 1/4$. Therefore maximizing $\mathcal{P}(P_1, P_2)$ is equivalent to finding T such that $\left|\prod_{i \in T} h_i - 1/2\right|$ is minimized.

Given $S = \{c_1, \ldots, c_m\}$, where the c_i are positive integers and $\sum_{i=1}^{m} c_i = M$. Define $h_i^* = (1/2)^{2c_i/M}$. With these definitions we can see that there is T such that

$$\prod_{i \in T} h_i^* = \left(\frac{1}{2}\right)^{\frac{2}{M}\sum_{i \in T} c_i} = \frac{1}{2} \tag{2.9}$$

if and only if T is a partition of S. Or equivalently, if and only if for the corresponding P_2 we have

$$\mathcal{P}(P_1, P_2) \geq b\left(1 + f\left(\frac{1}{2}\right)\right) = \frac{5}{4}b = p_{th}^*, \tag{2.10}$$

where the last equality is a definition.

The h_i^*'s are not necessarily rational numbers, and therefore they cannot be

taken as input data for the complementary path problem. We next give a sequence of lemmas by which we prove that using a suitably defined rational approximation of the h_i^*'s, which we denote as h_i's, we obtain a reduction to Partition. The gist of the argument is as follows. Given any $T_1, T_2 \subseteq \{1, \ldots, m\}$ we first prove that either $\prod_{i \in T_1} h_i^* = \prod_{i \in T_2} h_i^*$ or $|\prod_{i \in T_1} h_i^* - \prod_{i \in T_2} h_i^*| \geq \Delta$. We then prove that if we have h_i's that are 'close enough' to the h_i^*'s we shall have that for any T $|\prod_{i \in T} h_i^* - \prod_{i \in T} h_i| \leq \Delta/4$. Thus, given the value of $\prod_{i \in T} h_i$ it is possible to determine the value of $\prod_{i \in T} h_i^*$. From which, if we have $|\prod_{i \in T} h_i - 1/2| \leq \Delta/4$ we can conclude $\prod_{i \in T} h_i^* = 1/2$. Hence, $\sum_{i \in T} c_i = (1/2)M$. Then we prove that it is possible to find the h_i's in polynomial time.

Lemma 2.1 *If for any $T \subseteq \{1, \ldots, m\}$ we have $|\prod_{i \in T} h_i^* - \prod_{i \in T} h_i| < \frac{1}{16M}$, then $\sum_{i \in T} c_i = \frac{1}{2}M$ if and only if there exists P_2 such that*

$$\mathcal{P}(P_1, P_2) = \prod_{i=1}^{m} h_i \left(1 + f\left(\prod_{I \in T} h_i\right)\right) \geq \prod_{i=1}^{m} h_i \left(\frac{5}{4} - \frac{1}{64M^2}\right) = p_{th}, \qquad (2.11)$$

in which the last equality is a definition.

Proof. Define $g(i) = (1/2)^{2i/M}$, $i = 1, \ldots, M$. Notice that for any $T \subseteq \{1, \ldots, m\}$ there is an integer i, $1 \leq i \leq m$ such that $\prod_{i \in T} h_i^* = g(i)$. Therefore given $T_1, T_2 \subseteq \{1, \ldots, m\}$, such that $\prod_{i \in T_1} h_i^* \neq \prod_{i \in T_2} h_i^*$ there exist integers $1 \leq i, j \leq M, i \neq j$ such that

$$|\prod_{i \in T_1} h_i^* - \prod_{i \in T_2} h_i^*| \;=\; |g(i) - g(j)| \geq |g(M-1) - g(M)|$$

$$\geq \frac{1}{4}|2^{\frac{2}{M}} - 1| \geq \frac{1}{4}|1 + \frac{2\ln 2}{M} + R_2 - 1|$$

$$> \frac{\ln 2}{2M} > \frac{1}{4M} = \Delta, \tag{2.12}$$

in which the last equality is a definition. We have used a Taylor's series expansion for 2^x, in which $R_2 \geq 0$ is the remainder. We have proved that, given $T_1, T_2 \subseteq \{1, \ldots, m\}$ if $\prod_{i \in T_1} h_i^* \neq \prod_{i \in T_2} h_i^*$ then

$$|\prod_{i \in T_1} h_i^* - \prod_{i \in T_2} h_i^*| > \Delta \tag{2.13}$$

If we are given rational approximations h_i of the h_i^*'s $(i = 1, \ldots, m)$ such that for any $T \subseteq \{1, \ldots, m\}$ it is guaranteed $|\prod_{i \in T} h_i^* - \prod_{i \in T} h_i| \leq \Delta/4 = 1/16M$. Then $\prod_{i \in T} h_i^* = g(j)$ implies that the corresponding product of the h_i's over the same set T is going to be closer to $g(j)$ than to any other $g(k)$ for $k \neq j$. Hence, for any $T \subseteq \{1, \ldots, m\}$, if j minimizes $|\prod_{i \in T} h_i - g(k)|$ over $k = 1, \ldots, M$ we can conclude that $\prod_{i \in T} h_i^* = g(j)$. Consequently, if there is T such that $|\prod_{i \in T} h_i - 1/2| < 1/16M$, we can infer that $\prod_{i \in T} h_i^* = 1/2$, which implies that $\sum_{i \in T} c_i = 1/2$ and viceversa. Since $f(x) = x - x^2$ is a concave function with maximum at $1/2$ the above is also equivalent to

$$f\left(\prod_{i \in T} h_i\right) \geq f\left(\frac{1}{2} - \frac{1}{16M}\right) = \left(\frac{1}{4} - \frac{1}{64M^2}\right), \tag{2.14}$$

and thus equivalent to the existence of a path P_2 such that

$$p(P_1, P_2) \geq \prod_{i=1}^{m} h_i \left(\frac{5}{4} - \frac{1}{64M^2}\right) = p_{th}, \tag{2.15}$$

where the last equation is a definition.

∎

Lemma 2.2 *If* $|h_i^* - h_i| < 3^{-m}/16M$, *for* $i = 1, \ldots, m$, *then*

$$|\prod_{i \in T} h_i^* - \prod_{i \in T} h_i| < 1/16M \tag{2.16}$$

for any $T \subseteq \{1, \ldots, m\}$.

Proof. We first proof an auxiliary lemma.

Lemma 2.3 *If* $|h_i^* - h_i| < \epsilon$, *for* $i = 1, \ldots, m$, *where* $0 < \epsilon, h_i^*, h_i < 1$. *then we have*

$$|\prod_{i \in T} h_i^* - \prod_{i \in T} h_i| < 3^{|T|}\epsilon. \tag{2.17}$$

Proof. We prove this by induction in $|T|$. For $|T| = 1$ this statement corresponds to the hypothesis. Suppose now that equation (2.17) holds true for any T such that $|T| = n$. Consider now a set $Q \subseteq \{1, \ldots, m\}$ such that $|Q| = n + 1$. Take, without loss of generality, that $Q = \{1, 2, \ldots, n + 1\}$. Let $T = \{1, \ldots, n\}$.Then we have $\prod_{i \in T} h_i = \prod_{i \in T} h_i^* + \delta_n$, for some δ_n such that $|\delta_n| < 3^n\epsilon$, also $h_{n+1} = h_{n+1}^* + \delta$, for some δ such that $|\delta| < \epsilon$. Therefore

$$\left(\prod_{i \in T} h_i\right) h_{n+1} = \left(\prod_{i \in T} h_i^* + \delta_n\right)\left(h_{n+1}^* + \delta\right) = \left(\prod_{i \in T} h_i^*\right) h_{n+1}^* + \delta \prod_{i \in T} h_i^* + \delta_n h_{n+1}^* + \delta\delta_n. \tag{2.18}$$

From which have

$$\begin{aligned} |\prod_{i\in Q} h_i^* - \prod_{i\in Q} h_i| &= |\delta \prod_{i\in T} h_i^* + \delta_n h_{n+1}^* + \delta\delta_n| \\ &\leq |\delta \prod_{i\in T} h_i^*| + |\delta_n h_{n+1}^*| + |\delta\delta_n| \\ &< |\delta| + |\delta_n| + |\delta_n| < \epsilon + 3^n\epsilon + 3^n\epsilon < 3^{n+1}\epsilon. \end{aligned} \tag{2.19}$$

∎

We now require that for any T we have $|\prod_{i\in T} h_i^* - \prod_{i\in T} h_i| \leq \Delta/4 = 1/16M$. We have from the previous lemma that to obtain $|\prod_{i\in T} h_i^* - \prod_{i\in T} h_i| \leq 3^{|T|}\epsilon \leq 3^m\epsilon$ it is enough to ask for $|h_i^* - h_i| < \epsilon$, for $i = 1,\ldots,m$ with $3^m\epsilon \leq 1/16M$, i.e. $\epsilon \leq 3^{-m}/16M$. I.e. if we allow an error of $\epsilon \leq 3^{-m}/16M$ in the approximation of the h_i* by the h_i, we will obtain that the $\prod_{i\in T} h_i$ are within a tolerance of $\Delta/4 = 1/16M$.

∎

It rests to show that the h_i can be calculated in polynomial time, and that we will obtain h_i whose binary representation will have length polynomially related to the length of the binary representation of the particular instance of the partition problem reduced. First we need a new definition. Given $a = a'/a''$, where a is a rational, and both a', a'' are integers, $a'' \neq 0$, we say that the *height* of a is $\max\{|a'|, |a''|\}$.

Lemma 2.4 *Given $c_1, c_2, \ldots, c_m$, positive integers such that $\sum_{i=1}^{m} c_i = M$, there exists an algorithm that finds rationals h_i, for $i = 1, \ldots, m$, such that $|h_i - (1/2)^{2c_i/M}| < 3^{-m}/16M$ in time polynomial in m and $\log M$. Moreover the height of the h_i's has a binary encoding of length polynomial in m and $\log M$.*

Proof. We want to calculate $(1/2)^x$ where $0 \leq x = 2c_i/M \leq 2$. We have $(1/2)^x = 1/e^{x \ln 2}$. Notice also that if $x \ln 2 > 1$ we have $e^{x \ln 2} = e e^{x \ln 2 - 1}$, with $0 < x \ln 2 - 1 < 1$. Therefore to approximate $(1/2)^x, 0 \leq x \leq 2$ within a precision of $\epsilon = 3^{-m}/16M$ it is sufficient to be able to calculate $e^x, 0 \leq x \leq 1$ with precision $O(\epsilon)$, and to be able to calculate $\ln 2$ within $O(\epsilon)$. For the calculation of $\ln 2$ and of $e^x, 0 \leq x \leq 1$ we have the following lemmas.

Lemma 2.5 *Let x be a rational $0 \leq x \leq 1$ such that the height of x has a binary encoding of d bits. There exists an algorithm that finds a rational q such that $|e^x - q| < \epsilon$ in time that is polynomial in d and $\log \epsilon^{-1}$.*

Proof. Using a Taylor's series expansion for e^x in the interval $x \in [0, 1]$ we get

$$e^x = 1 + \frac{x^2}{2!} + \cdots + \frac{x^{n-1}}{(n-1)!} + R_n \tag{2.20}$$

where $R_n = x^n e^{\theta x}/n!, 0 \leq \theta \leq 1$. It is plain that since $e < 3$ we have $R_n < 3/n!$. Choose a minimal n such that $3/n! < \epsilon$, or equivalently $\log n! > \log \epsilon^{-1} + \log 3$. Therefore $\log n! = O(\log \epsilon^{-1})$. From Stirling's formula we have that $\ln n! \approx n \ln n$, which implies $n = O(n \ln n) = O(\log \epsilon^{-1})$. Now, define

$$T = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^{n-1}}{(n-1)!}, \tag{2.21}$$

this is a rational approximation of e^x. The height of each of the terms in T, $x^{(k-1)}/(k-1)!$ for $k = 1, \ldots, n-1$, has a binary representation of $O((n-1)d + \log(n-1)!) = O((\log \epsilon^{-1})d + \log \epsilon^{-1}) = O((\log \epsilon^{-1})d)$. T has $O(\log \epsilon^{-1})$ terms. Therefore the height of T has a binary encoding of $O((\log \epsilon^{-1})^2 d)$. Define now $a_0 = 1, a_{i+1} = xa_i/i, i = 1, \ldots, n-2$, then $T = \sum_{i=1}^{n-1} a_i$, from which we conclude that T can be calculated with $O(\log \epsilon^{-1})$ additions and the same number of multiplications. Each of these involve rationals whose heights have binary encodings of lenght $O((\log \epsilon^{-1})d)$. Take $q = T$.

∎

Lemma 2.6 [Chandrasekaran & Tamir] *There exists an algorithm that finds a rational q such that $|\ln 2 - q| < \epsilon$, for $0 < \epsilon < 1$, in time polynomial in $\log \epsilon^{-1}$. Moreover the height of q has a binary encoding of lenght polynomial in $\log \epsilon^{-1}$.*

Proof. We have that

$$\ln x = \sum_{k=0}^{n} \frac{2}{2k+a} \left(\frac{x-1}{x+1}\right)^{2k+1} + R. \tag{2.22}$$

where

$$0 \leq R \leq \frac{(x+1)^2}{2x(2x+3)} \left(\frac{x-1}{x+1}\right)^{2n+3}, \text{ for } x > 1 \tag{2.23}$$

Using equation (2.23) with $x = 2$ we get $R \leq 3^{-(2n+3)}$, we choose a minimal n such

that $3^{-(2n+3)} < \epsilon$, i.e. $n = O(\log \epsilon^{-1})$. $\ln 2$ is now approximated by

$$q = \sum_{k=0}^{n} \frac{2}{2k+1} 3^{-(2k+1)} \tag{2.24}$$

in $O(\log \epsilon^{-1})$ time, the height of q has an encoding of length $O((\log \epsilon^{-1})^2)$ bits.

∎

Notice that we have also proved that the following problem is *NP*-complete.

Rational Multiplicative Partition.

INSTANCE: Given $q_1, q_2, \ldots, q_m$, rational numbers such that $q_i \geq 1/2, i = 1, \ldots, m$.

QUESTION: Is there $T \subseteq \{q_1, \ldots, q_m\}$ such that $\prod_T q_i = 1/2$?

2.4 Complementary Path Problem. Logarithmic Input.

In this section we consider the complementary path problem when the link probabilities are given by $p_l = a^{n_l}$, for a rational $0 < a < 1$, and a positive integer n_l. The input of the problem consists of the rational number a, and the integers $n_l, l \in L$. We say that this is a *logarithmic* representation of the p_l. We give a similar result to that of theorem 2.1;that is we prove that this problem is *NP*-hard, the proof is very similar to that of theorem 2.1.

The complementary path problem with logarithmic inputs is not known to be in *NP*. That is, for a rational a; integers n_1, n_2; and a rational p_{th}; it is not known if it can be decided in polynomial time whether

$$a^{n_1} + a^{n_2} - a^{n_1+n_2} \geq p_{th}.$$

The exponentiation operation, a^n, can be realized in $O(\log n)$ multiplications, but the length of the binary representation of the result is not a polynomial in the length of the input. It can be argued though that in comparison with the *standard* method of representing the probabilities, i.e. as a ratio of two integers, the logarithmic representation contains 'more information', in the following sense. If we are interested in representing probabilities p in the range $0 < \epsilon < p < 1$, using the standard representation we will need $O(\log \epsilon^{-1})$ bits to encode the p's, on the other hand, using logarithmic representation, assuming that this is acceptable, we need $O(\log \log \epsilon^{-1})$ bits. Consequently, with logarithmic representation, $p = a^n$, for some n, can be calculated in $O(\log \epsilon^{-1})$.

In [2] it is shown that there exists a polynomial algorithm for deciding whether $a_1^{n_1} \geq a_2^{n_2}$, for rationals a_1, a_2 and integers n_1, n_2. The techniques used there do not seem to be useful for the problem we consider here.

Theorem 2.2 *The complementary path problem with logarithmic inputs is NP-hard.*

Proof. Consider the same situation as in theorem 2.1, with the same definitions as made there. Consider an instance of partition as above. Define $a^* = (1/2)^{2/M}$,

$n_{l_j^0} = c_j$, and $n_{l_j^1} = 2c_j$. With these definitions we have $h_i^* = a^{*c_i}, t_i^* = a^{*2c_i}$, and therefore $t_i^* = h_i^{*2}$. As before there is T such that

$$\prod_{i \in T} h_i^* = \left(\frac{1}{2}\right)^{\frac{2}{M}\sum_{i \in T} c_i} = \frac{1}{2} \tag{2.25}$$

if and only if $\sum_{i \in T} c_i = (1/2)M$. a^* is not necessarily a rational number, therefore it cannot be used as part of the input for the complementary path problem with logarithmic representation. We next prove that a suitably defined rational approximation of a^* results in a reduction to Partition.

Lemma 2.7 *There exists an algorithm that finds a rational a such that $|a^{*k} - a^k| \leq (1/16M) = \Delta/4$, for any $k = 1, \ldots, M$ in time polynomial in $\log M$. Moreover the height of a has a binary encoding of length polynomial in $\log M$.*

Proof. Consider the function $g(x) = x^k, 0 < x < 1$. By a Taylor's series expansion we have

$$g(x_0 + \Delta x) = g(x_0) + \Delta x g'(x_0 + \theta \Delta x), \;\; 0 \leq \theta \leq 1. \tag{2.26}$$

Therefore

$$\begin{aligned} |g(x_0 + \Delta x) - g(x_0)| &= |\Delta x||g'(x_0 + \theta \Delta x)| = |\Delta x||k(x_0 + \theta \Delta x)^{k-1}| \\ &\leq k|\Delta x| \leq M|\Delta x|. \end{aligned} \tag{2.27}$$

Where we have taken that Δx is small enough to guarantee that $0 \leq x_0 + \Delta x \leq 1$.

Therefore to guarantee $|a^{*k} - a^k| = |g(a^*) - g(a)| \leq 1/16M$ it is sufficient to require $M|a^* - a| \leq 1/16M$, or equivalently $|a^* - a| \leq 1/16M^2$. I.e. we need to calculate $a^* = (1/2)^{2/M}$ within an accuracy of $1/16M^2$. The proof is completed by an argument similar to that of lemma 2.4.

∎

Given any $T \subseteq \{1, \ldots, m\}$ we have that

$$|a^{*\sum_{i\in T} c_i} - a^{\sum_{i\in T} c_i}| \leq \frac{\Delta}{4}.$$

Therefore $|a^{\sum_{i\in T} c_i} - 1/2| \leq \Delta/4$ if and only if $\sum_{i\in T} c_i = 1/2$. Which is equivalent to the existence of a path P_2 such that

$$p(P_1, P_2) \geq \prod_{i=1}^{m} h_i \left(\frac{5}{4} - \frac{1}{64M^2}\right) = p_{th}.$$

∎

2.5 An Equivalent Problem

In this section we are going to show that the complementary path problem is a special case of a more general problem that has a 'cleaner' structure. The algorithm to be presented in section 2.6 can be used to solve this more general problem. In

the rest of this chapter we shall assume that we have $p_l = a^{n_l}$ where $0 < a < 1$ is a rational and n_l is a positive integer, i.e. logarithmic input.

Deleting the first constant term in the expression for $P(P_1, P_2)$, and factorizing $\prod_{l \in P_2} p_l$,equation (2.4), we get.

$$\begin{aligned} R' &= \max_{P_2} \prod_{l \in P_2} p_l \left(1 - \prod_{l \in P_1 \setminus P_2} p_l \right) \\ &= \max_{P_2} \prod_{l \in P_1} p_l \left(1 - \prod_{l \in P_2} p_l \left(\prod_{l \in P_2 \bigcap P_1} p_l \right)^{-1} \right). \end{aligned} \tag{2.28}$$

Define now

$$\begin{aligned} q_l &= \begin{cases} p_l, & \text{if } l \in P_1; \\ 1, & \text{otherwise,} \end{cases} \\ \beta &= \prod_{l \in P_1} p_l. \end{aligned} \tag{2.29}$$

We have then

$$R' = \max_{P_2} \prod_{l \in P_2} p_l \left(1 - \beta \left(\prod_{l \in P_2} q_l \right)^{-1} \right). \tag{2.30}$$

The logarithm is a monotonically increasing function, therefore

$$r = -\ln R' = \min_{P_2} \left\{ \sum_{l \in P_2} (-\ln p_l) - \ln \left(1 - \beta \left(\prod_{l \in P_2} q_l \right)^{-1} \right) \right\}$$

$$= \min_{P_2} \left\{ \sum_{l \in P_2} (-\ln p_l) - \ln \left(1 - \beta \exp \left(\ln \left(\prod_{l \in P_2} q_l \right)^{-1} \right) \right) \right\}$$

$$= \min_{P_2} \left\{ \sum_{l \in P_2} (-\ln p_l) - \ln \left(1 - \beta \exp \left(\sum_{l \in P_2} (-\ln q_l) \right) \right) \right\} \quad (2.31)$$

We assume that for $l \in L$, $p_l = a^{n_l}$; with $0 < a < 1$, n_l a non-negative integer. Defining now

$$\begin{aligned} \alpha &= -\ln a, \\ d_l &= -\ln p_l = n_l \alpha, \\ w_l &= -\ln q_l, \\ f(x) &= -\ln(1 - \beta e^x). \end{aligned} \quad (2.32)$$

we get the following equivalent problem. Find P_2 achieving

$$r = \min_{P_2} \left\{ \sum_{l \in P_2} d_l + f \left(\sum_{l \in P_2} w_l \right) \right\}. \quad (2.33)$$

Note that f is an increasing function. For any P_2,

$$f \left(\sum_{l \in P_2} w_l \right) = -\ln(1 - \prod_{l \in P_1 \setminus P_2} p_l) \geq 0. \quad (2.34)$$

2.6 The Algorithm

Here we address the following problem.

Generalized Complementary Path Problem.

INSTANCE: Given $G = (N, L)$ a directed graph as in section 2.2, given for each link l two integers $d_l > 0$ and $w_l \geq 0$. Given f, a monotonically increasing positive real function.

TASK: Find P, a simple directed path in the network connecting nodes s and t, such that

$$\sum_{l \in P} d_l + f\left(\sum_{l \in P} w_l\right)$$

is minimized.

With the definitions given in equations (2.29) and (2.32) this is equivalent to the original problem as defined in section 2.2. In this section we derive an algorithm for this more general formulation.

Given P_1, P_2, two s-t paths in G we say that P_1 dominates P_2 if $\sum_{l \in P_1} d_l \leq \sum_{l \in P_2} d_l$ and $\sum_{l \in P_1} w_l \leq \sum_{l \in P_2} w_l$ and at least one of these two inequalities is strict. An s-t-path P_1 is said to be non-dominated if there is no other path that dominates it. We now have

Lemma 2.8 *The minimum of* $g(P) = \sum_{l \in P} d_l + f(\sum_{l \in P} w_l)$ *occurs at a non-dominated path.*

Proof. Define $d_P = \sum_{l \in P} d_l$, $w_P = \sum_{l \in P} w_l$. $g(P) = d_l + f(w_l)$ is a monotonously

increasing function of both d_P and w_P. Suppose that P^* minimizes $g(P)$ and that P^{**} dominates P^*, then either $d_{P^{**}} < d_{P^*}$ or $w_{P^{**}} < w_{P^*}$, this implies $g(P^{**}) < g(P^*)$, which is a contradiction.

■

Let P^j denote a path from node s to node j. Define $c_j(D)$ as the minimum over all P^j of $\sum_{l \in P^j} w_l$ subject to the condition that $\sum_{l \in P^j} d_l \leq D$. If for a given D there is no s-j path with $\sum_{l \in P^j} d_l \leq D$ we say $c_j(D) = +\infty$.

It is easy to see that $c_j(D)$ is a decreasing function of D. Also if $c_t(D) < c_t(D-1)$ we have that there exists P^t a non-dominated s-t path with $\sum_{l \in P^t} w_l = c_t(D)$ and $\sum_{l \in P^t} d_l = D$. Also every pair (w, d) corresponding to a non-dominated path shows as $w = c_t(d) < c_t(d-1)$.

We can now set up the following relations

$$
\begin{aligned}
c_s(0) &= 0, \\
c_j(D) &= +\infty \text{ for } j \neq s \text{ and } D \leq 0, \\
c_j(D) &= \min\{c_j(D-1), \min_{\langle k,j \rangle}\{c_k(D - d_{\langle k,j \rangle}) + w_{\langle k,j \rangle}\}\} \qquad (2.35)
\end{aligned}
$$

For a given D, assuming that $c_j(d)$ is known for $x < D$, it takes $O(|L|)$ operations to calculate $c_j(D)$. We can start with $D = 1$ and proceed until $D = M$, where as M we can use an upper bound for the length, as measured by $\sum_{l \in P} d_l$, a simple s-t path

in G can have. I.e. $M \geq \sum_{l \in P} d_l$, for any simple path P between nodes s and t. For example, $M = \sum_{l \in L} d_l$ is such a bound. Given that a simple path in the network can have no more than $|N| - 1$ links, if we suppose, without loss of generality, $d_1 \geq d_2 \geq \cdots \geq d_n$, then $M = \sum_{i=1}^{|N|-1} d_l$ is a tighter bound. $M = (|N| - 1)d_1$ can also serve as a bound. If after deleting all links l such that $w_l > 0$, nodes s and t remain connected in the resulting network G', then we can use as M the lenght of a d-shortest s-t path in G'.

This results in an $O(M|L|)$ calculation. The presence of M here classifies this algorithm as pseudo-polynomial. To find the optimum of $g(P)$ it is sufficient to identify all pairs (w, d) corresponding to non-dominated paths, of which there will be at most M, and identify one that achieves the minimum, an operation that takes $O(M)$ steps. Notice that since the minimum of $g(P)$ coincides with the minimum of $\mathcal{P}(P_1, P_2)$ in the complementary path problem we can compare the different non-dominated points by evaluating $\mathcal{P}(P_1, P_2)$ for these points.

Chapter 3

Routing Two Messages Along Disjoint Paths

3.1 Introduction

In this chapter we address the following problem. We wish to transmit a message between two given nodes, nodes s and t, of an unreliable data-communication network. To increase the probability that the message is received we will send two copies of the message. In addition we include the restriction that the paths followed by the two copies should be link-disjoint, i.e. they should share no links. In contrast to the problem considered in chapter two neither of the two paths is fixed. How should these two copies be routed so as to maximize the probability that at least one of them is received at node t?

In section 3.2 of this chapter the problem is precisely stated and the solution

points are characterized. In section 3.3 we give a solution for the particular case of an acyclic network. In section 3.4 we prove that this problem is *NP*-complete for an unrestricted network, i.e. a not necessarily acyclic network. For logarithmic inputs the problem is shown to be strongly *NP*-complete. In the next chapter we address the same problem without the disjointedness restriction.

3.2 Disjoint Paths

We now consider the following problem.

Directed two link-disjoint path maximum reliability problem.

INSTANCE: Given a network $G = (N, L)$, where N is the set of nodes of the network, and $L \subseteq N \times N$ is the set of links. Given for each link l a number p_l, $0 \leq p_l \leq 1$. Where this p_l is the probability that link l is operative. For every link l we have $p_l = a^{n_l}$, where n_l is a non-negative integer, and $0 < a < 1$.

TASK: Find two simple link-disjoint paths P_1 and P_2, both joining node s to node t, such that the probability that at least one of the two paths is operative is maximized. A path is operative if every link in the path is operative.

Given two nodes i, j, if the link $\langle i, j \rangle \notin L$, then we shall take $n_{\langle i,j \rangle} = \infty$. We want to find paths P_1 and P_2 such that $\mathcal{P}(P_1, P_2)$, the probability that at least one of these two paths is operative, is maximized.

$$\mathcal{P}(P_1, P_2) = \prod_{l \in P_1} p_l + \prod_{l \in P_2} p_l - \prod_{l \in P_1} p_l \prod_{l \in P_2} p_l \tag{3.36}$$

$$= \quad a^{\sum_{l\in P_1} n_l} + a^{\sum_{l\in P_2} n_l} - a^{\left(\sum_{l\in P_1} n_l + \sum_{l\in P_2} n_l\right)}. \qquad (3.37)$$

Define now

$$g(r_1, r_2) = a^{r_1} + a^{r_2} - a^{r_1+r_2}. \qquad (3.38)$$

Then, we have

$$\mathcal{P}(P_1, P_2) = g\left(\sum_{l\in P_1} n_l, \sum_{l\in P_2} n_l\right). \qquad (3.39)$$

Theorem 3.1, which follows, allows us to characterize an optimum pair of paths P_1^*, P_2^*. With this characterization we shall later be able to produce an algorithm to find an optimum pair of disjoint paths for an acyclic network.

Theorem 3.1 *Given R, a finite subset of points (r_1, r_2) belonging to the positive orthant of $\mathcal{R}^2$, where by $\mathcal{R}$ we denote the real line. A point $(r_1^*, r_2^*) \in R$, maximizing the function*

$$g(r_1, r_2) = a^{r_1} + a^{r_2} - a^{r_1+r_2} \qquad \text{over } R \qquad (3.40)$$

is an extreme point of the convex hull of $\overline{R}^d$, By $\overline{R}$ we mean the convex hull of R, and by $\overline{R}^d$ the union of $\overline{R}$ with all points dominated by some point of $\overline{R}$. A point (r_1^*, r_2^*) dominates a point (r_1, r_2) if and only if $r_1^* \leq r_1$ and $r_2^* \leq r_2$ and at least one of the two inequalities is strict. I.e.

$$\overline{R}^d = \overline{R} \cup \{x : \exists r \in \overline{R} \text{such that } r \text{ dominates } s\} \qquad (3.41)$$

Notice that $R \subseteq \overline{R} \subseteq \overline{R}^d$, and an extreme point of $\overline{R}^d$ is a non-dominated point, i.e. it is not dominated by any other point in $\overline{R}^d$.

We shall first prove three auxiliary lemmata, and then give the proof for theorem 3.1.

Lemma 3.1 *$g(r_1, r_2)$ is a decreasing function of both of its arguments.*

Proof. $g(r_1, r_2) = a^{r_1}(1 - a^{r_2}) + a^{r_2}$, for fixed $r_2 = c > 0$ we have

$$f(r_1) = g(r_1, c) = (1 - a^c)a^{r_1} + a^c \tag{3.42}$$

and

$$\frac{df}{dr_1} = (1 - a^c)a^{r_1} \ln a < 0. \tag{3.43}$$

∎

Lemma 3.2 *A point (r_1^*, r_2^*) maximizing the function g over R is non-dominated.*

Proof. Suppose that there exists $(r_1, r_2) \in R$ that dominates (r_1^*, r_2^*). Then, either $r_1 < r_1^*$, or $r_2 < r_2^*$ from which, from lemma 3.1, we get $g(r_1^*, r_2^*) < g(r_1, r_2)$, this is a contradiction.

∎

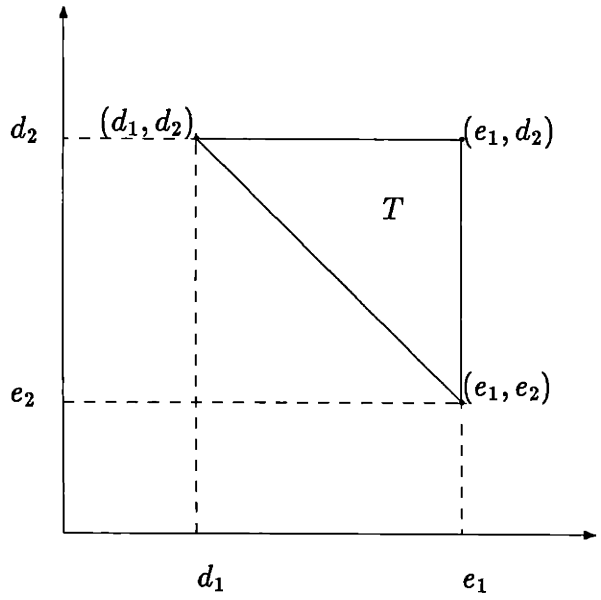


Figure 3.1: Boundary of R

Lemma 3.3 *Given points (d_1, d_2) and (e_1, e_2), such that $d_1 < e_1$ and $e_2 < d_2$, the maximum of g in T, the convex hull of (d_1, d_2), (e_1, e_2) and (e_1, d_2) is attained at (d_1, d_2) or (e_1, e_2), see figure 3.1.*

Proof. The region T is defined by, $r_1 \leq e_1$, $r_2 \leq d_2$ and $r_2 \geq -br_1 + c$, where $b = (d_2 - e_2)/(e_1 - d_1) > 0$ and $c = e_1 b + e_2 > 0$. By lemma 3.2 the maximum of g in T will occur at some point of L, the straight line connecting points (d_1, d_2) and (e_1, e_2), every other point in T is dominated by some point in L. Now we need only show that the maximum of g in L is attained at (d_1, d_2) or (e_1, e_2).

Define $p_1 = a^{r_1}$, $p_2 = a^{r_2}$. This change of variables maps the positive orthant

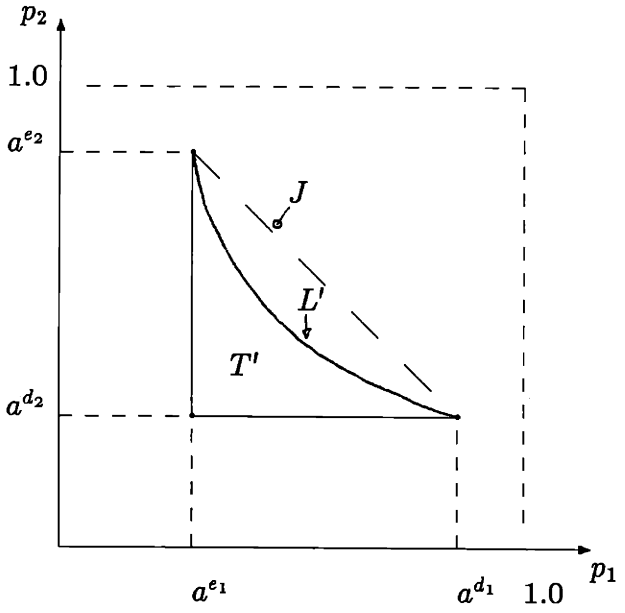


Figure 3.2:

of $\mathcal{R}^2$ to the unit square $0 \leq p_1 \leq 1$, $0 \leq p_2 \leq 1$. T', the image of T under this transformation is defined by $p_1 \geq a^{e_1}$, $p_2 \geq a^{d_2}$, and $p_2 \leq a^c p_1^{-b}$, see figure 3.2.

Evaluating $g(r_1, r_2)$ is equivalent to evaluating $f(p_1, p_2) = p_1 + p_2 - p_1 p_2$. The maximum of f in T' is in some point of L', the image of L. Consider now J, the straight line that joins (a^{d_1}, a^{d_2}) to (a^{e_1}, a^{e_2}). For any point $(p_1, p_2) \in L'$, there is a point $(p_1', p_2') \in J$ such that $f(p_1', p_2') \geq f(p_1, p_2)$. We have

$$J = \{(p_1, p_2) : \ p_2 = -mp_1 + n, \ a^{e_1} \leq p_1 \leq a^{d_1}; \text{ for some } m, n > 0\}. \qquad (3.44)$$

The function g evaluated in J is

$$g(p_1, -mp_1 + n) = mp_1^2 + (1 - m - n)p_1 + n, \tag{3.45}$$

this is a convex function of p_1. This implies that the maximum of g in J is attained at either $p = a^{e_1}$ or at $p = a^{d_1}$, i.e. at either (a^{e_1}, a^{e_2}) or at (a^{d_1}, a^{d_2}). These points belong to L', this implies that one of them maximizes g in L' too. Therefore, f in T is maximized by either (e_1, e_2) or (d_1, d_2).

∎

Lemma 3.4 *A non-dominated point of R, (r_1, r_2), is either an extreme point of $\overline{R}^d$, or there are two extreme points of $\overline{R}^d$, nominally, (e_1, e_2), (d_1, d_2), such that $d_1 \leq r_1 \leq e_1$, and $e_2 \leq r_2 \leq d_2$.*

Proof. Comes directly from the definitions.

∎

Proof of Theorem 3.1: Consider $(r_1, r_2) \in R$, suppose (r_1, r_2) is not dominated by any point of R, and (r_1, r_2) is not an extreme point of R^d, see figure 3.3. Then, by lemma 3.4, there exist points (e_1, e_2), (d_1, d_2) as in lemma 3.3, extreme points of $\overline{R}^d$ such that $(r_1, r_2) \in T$, as defined in the lemma. Then, either (e_1, e_2) or (d_1, d_2) achieve a larger value of g than that achieved by (r_1, r_2).

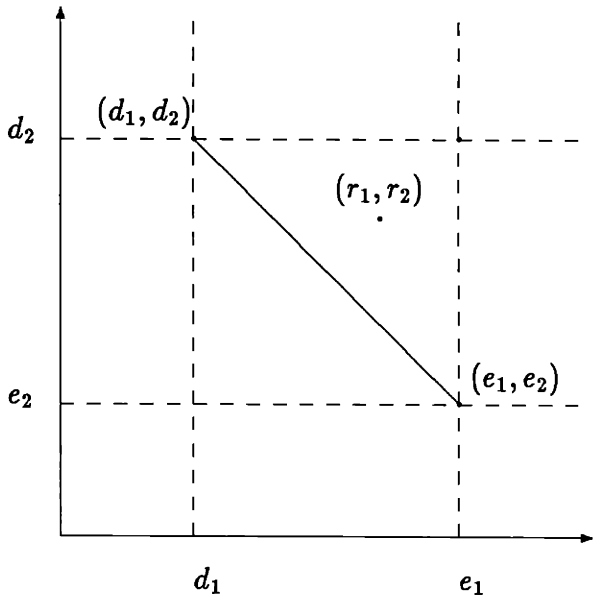


Figure 3.3:

Thus, one way to compute the optimum of $\mathcal{P}(P_1, P_2)$ is to generate the set of all extreme points of $\overline{R}^d$, and then find which of them maximizes g. If the number of these points is not excessive this approach is practical. In the rest of this section we whall give an upper bound to the number of extreme points of $\overline{R}^d$. As R we take the set of all pairs (r_1, r_2), where $r_i = \sum_{l \in P_i} n_l$, $i = 1, 2$, and P_1, P_2 are node-disjoint paths that connect nodes s and t in the network G.

3.2.1 A Bound on the Number of Non-Dominated Points of R^d

Observe from figure 3.1 that R is symmetrical, i.e. if (r_1, r_2) is in R, then (r_2, r_1) also belongs to R. Let r^* be the length of a shortest path in the network, a path connecting nodes s and t, where the length of link l is given by n_l. Thus, a^{r^*} is the probability that a most reliable path in the network is operative. Pick a shortest path of the network, it need not be unique. Delete from the network all the links belonging to this shortest path. Suppose that in the resulting network, G', the nodes s and t remain connected. Calculate in G' r^{**} the length of a shortest path. For any non-dominated point (r_1, r_2), we have that $r_1, r_2 \geq r^*$, and, as $(r^*, r^{**}) \in R$, we have $r_1, r_2 \leq r^{**}$. Therefore, the number of non-dominated points is bounded by $(r^{**} - r^* + 1)$, see figure 3.4. Notice also that if the point $(r^*, r^*) \in R$, i.e. if there are two disjoint paths both of shortest length, then this point is a maximum of the function g on R. Notice also that r^{**}, as generated here, need not be uniquely defined, its value might depend on which shortest path was originally deleted from

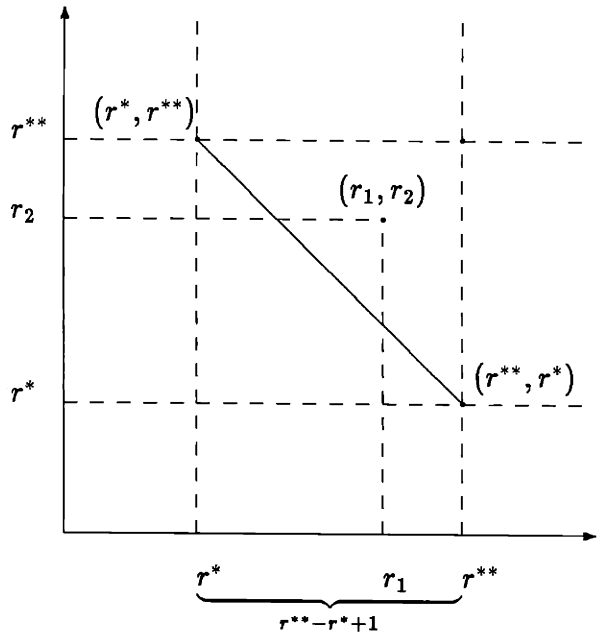


Figure 3.4: A Bound in the number of non-dominated points

the network.

3.3 A Solution for Acyclic networks

3.3.1 Generating the Extreme Points of R^d. Stratified Networks

Given G, an acyclic network, the set of nodes N can be partitioned into disjoint layers $S_1, S_2, \ldots, S_m$, where $m \leq n$, and $\bigcup_{i=1,\ldots,m} S_i = N$, and such that the layers have the property that for $k \in N$ all the links emanating from k go to a node in a higher numbered layer. Additionally we also impose the condition that $S_1 = \{s\}$, and $S_m = \{n\}$. Suppose, without loss of generality, that for $k \in S_i$ all the links emanating from k go to S_{i+1}. We say that a network that satisfies these properties is a *stratified* network. We call the S_i the *strata* of the network. See figure 3.5.

3.3.2 Squared Network

Given a stratified network $G = (N, L)$, with strata $S_1, \ldots, S_m$, we form G^2, the *squared* network, in the following way. We denote the nodes of G^2 by a pair of integers e.g. ij, G^2 consists of m strata $S_1^2, \ldots, S_m^2$. Similarly to G, $S_1^2 = \{ss\}$, and $S_m^2 = \{nn\}$.

The strata S_i^2 for $1 < i < m$ are created as follows. For every ordered pair of nodes $k, j \in S_i$ there exists a node $kj \in S_i^2$. There is an arc $\langle kl, rs \rangle$, connecting nodes kl in S_i^2 to node rs in S_{i+1}^2, if and only if there is an arc connecting node $k \in S_i$ to node $r \in S_{i+1}$, and an arc connecting node $l \in S_i$ to node $r \in S_{i+1}$, unless $k = l$ and $r = s$. This last requirement serves to guarantee disjoint paths. See

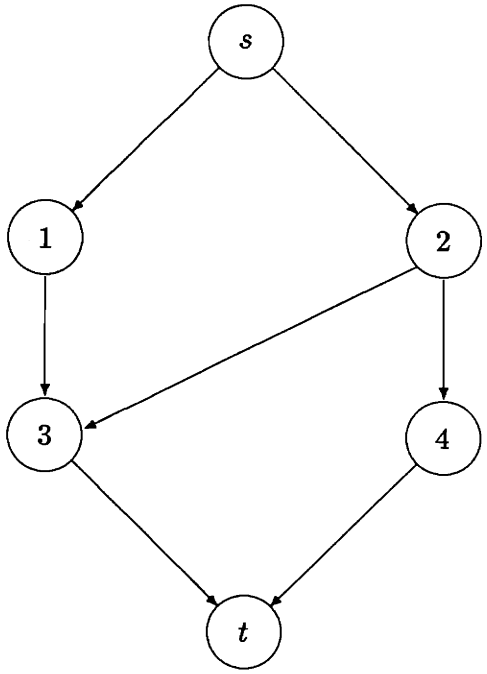


Figure 3.5: A stratified network

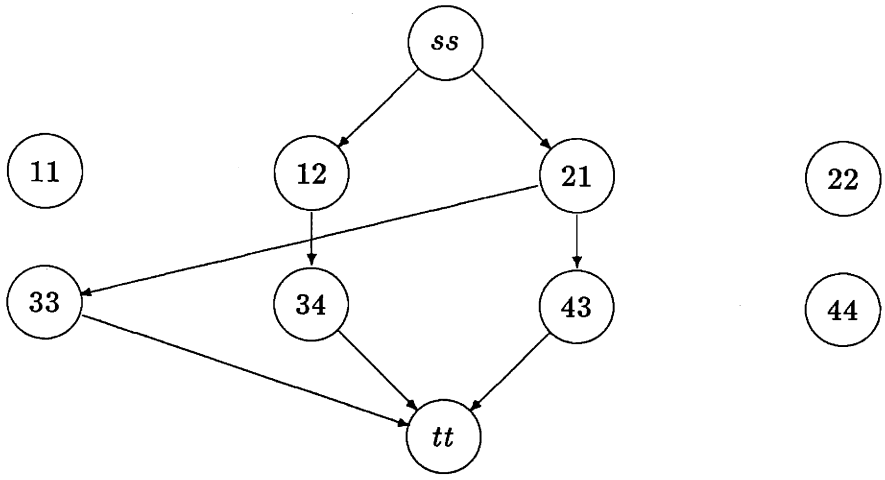


Figure 3.6: The network of the previous figure squared

figure 3.6.

A link $\langle ik, rs \rangle \in G^2$ is labeled by two coefficients, $c^1_{\langle ik,rs \rangle} = n_{\langle i,r \rangle}$ and $c^2_{\langle ik,rs \rangle} = n_{\langle k,s \rangle}$.

To every path connecting node ss to node tt in G^2 correspond two disjoint paths in G, both connecting node s to node t, and viceversa. The squared network device transforms the search for two disjoint paths in G into a search for a single path in G^2.

3.3.3 Characterization of Extreme Points

To each path from node ss to node tt in G^2 we can associate two numbers v_1 and v_2. These numbers are the lengths of the corresponding disjoint paths P_1 and P_2

in G. For a given path P in G^2, v_1 is obtained by adding the c^1's of the links in P, v_2 is obtained in analogous way. That is, $v_1 = \sum_{l \in P} c_l^1$, and $v_2 = \sum_{l \in P} c_l^2$.

Obtaining a path P^* minimizing one of these objectives can be formulated as a minimum-cost-flow problem. To each link $l \in L^2$ we associate a positive number f_l, which we interpret as the flow along link l. These f_l should satisfy the following constraints. With the exception of nodes ss and tt the flow in each node of G^2 should be conserved. The net flow in node ss should be outgoing, more flow going out than entering, with a value of one. The net flow in node tt should be entering, more flow entering than going out, with a value of one. That is,

$$\sum_{\{kl:\langle rs,kl\rangle \in L^2\}} f_{\langle rs,kl\rangle} - \sum_{\{kl:\langle kl,rs\rangle \in L^2\}} f_{\langle kl,rs\rangle} = 0, \; rs \in L^2, rs \neq ss, tt. \quad (3.46)$$

$$\sum_{\{kl:\langle ss,kl\rangle \in L^2\}} f_{\langle ss,kl\rangle} - \sum_{\{kl:\langle kl,ss\rangle \in L^2\}} f_{\langle kl,ss\rangle} = 1. \quad (3.47)$$

$$\sum_{\{kl:\langle tt,kl\rangle \in L^2\}} f_{\langle tt,kl\rangle} - \sum_{\{kl:\langle kl,tt\rangle \in L^2\}} f_{\langle kl,tt\rangle} = -1. \quad (3.48)$$

This is a linear program with two objective functions, v_1' and v_2'.

$$\begin{aligned} v_1' &= \sum_{\langle rs,kl\rangle \in L^2} c_l^1 f_{\langle rs,kl\rangle} \\ v_2' &= \sum_{\langle rs,kl\rangle \in L^2} c_l^2 f_{\langle rs,kl\rangle} \end{aligned} \quad (3.49)$$

Notice that the feasible region defined by these constraints corresponds to $\overline{R}$, the convex hull of R. If this linear program is expressed in matrix form the matrix corresponding to the flow-conservation constraints is unimodular. This guarantees

that for any objective linear function, if there is an optimum solution, there is an integer optimum solution. Which means that we may use the techniques and results of linear programming to find P^*. Moreover there exist standard algorithms, see [10], to find the set of all non-dominated points of a multiple-objective linear program. In their simplest form, the basic approach is to form a combined objective function $v_\lambda = v'_1 + \lambda v'_2$, for $\lambda > 0$. Non-dominated points with respect to the objectives v'_1 and v'_2 correspond to extreme points of $\overline{R}^d$. Varying λ the different non-dominated points are obtained. We need consider only as many different values of λ as there are non-dominated points. We are using only the fact that the points of R that optimize (3.36) are non-dominated points. To obtain a more efficient approach we would have to use additional properties of the function we are optimizing. In the next section we will present an algorithm to find two paths optimizing v_λ without explicitly constructing the squared network.

3.3.4 Two Minimum-Cost Paths with Different Path Costs

We first give an algorithm for a more general problem, and then specialize it to optimize v_λ. Given $G = (N, L)$ an acyclic network, in which $|N| = m$, we can take without loss of generality that the nodes in this network are so ordered that for any link $\langle i, j \rangle$ we have $i < j$. This numbering of the nodes can always be done in an acyclic network, see [9]. Let then s, the origin node, be node 1; and t, the destination node, be node m. To each link l in the network we associate two non-negative numbers c_l^1 and c_l^2.

Define for i, j, a pair of, not necessarily distinct, nodes, $S(i,j)$ as the set of all

pairs (P_1, P_2) of directed link-disjoint paths in which P_1 connects node s to node i, and P_2 connects nodes t to node j. Notice that for any link $\langle q, r\rangle$ in P_1 we have $q, r \leq i$, and similarly for links of P_2.

Define

$$d(i,j) = \min_{(P_1,P_2)\in S(i,j)} \left\{ \sum_{l\in P_1} c_l^1 + \sum_{l\in P_2} c_l^2 \right\}. \tag{3.50}$$

Our objective is to find $d(t,t)$. We take $d(s,s) = 0$. That is, we are looking for two minimum cost link-disjoint paths both joining node s to node t, for which the cost of a link l belonging to the first path is given by c_l^1, and the cost of a link l belonging to the second path is given by c_l^2. If we specialize the c_l^i by taking $c_l^2 = \lambda c_l^1$,$c_l = n_l$, we are minimizing v_λ, as defined in the previous section.

With this definitions we can write

$$d(i,k) = \min_{j<k} \left\{ d(i,j) + c^2_{\langle j,k\rangle} \right\} \quad i < k. \tag{3.51}$$

Similarly

$$d(k,i) = \min_{j<k} \left\{ d(j,i) + c^1_{\langle j,k\rangle} \right\} \quad i < k. \tag{3.52}$$

And

$$d(k,k) = \min_{\substack{i,j<k \\ j\neq i}} \left\{ c^1_{\langle i,k\rangle} + c^2_{\langle j,k\rangle} + d(i,j) \right\}. \tag{3.53}$$

This is because any link of the form $\langle i,k\rangle$ cannot be part of any P_1, P_2 in $S(i,j)$ for $i,j < k$. We can calculate the $d(i,j)$ by starting with $d(1,1) = 0$, and

for increasing values of k calculating the $d(i,k)$, $d(k,i)$, and $d(k,k)$ for $i < k$. We do this for $m-1$ different values of k, and each time we take $O(k^2)$ operations. We have an algorithm that uses $O(m^3)$ steps. The optimizing paths can be obtained by recording the values of the indices that achieve the minimum in 3.53.

Take now $c_l^1 = n_l$ and $c_l^2 = \lambda n_l$, then $d(s,s)$ corresponds to the optimum for v_λ. One could try to generalize this approach to general networks, i.e. not necessarily acyclic networks, but in the next section we prove that the problem of finding two minimum-cost link-disjoint paths with different path costs in a general network is *NP*-complete. Not only that, but the directed two link-disjoint path maximum reliability problem is also *NP*-complete, as will be shown in next section.

3.4 The Directed Two Path Maximum Reliability Problem is *NP*-Complete

We address the following problem.

Directed two link-disjoint path maximum reliability problem. (Decision version.)

INSTANCE: Given a network $G = (N, L)$, where N is the set of nodes of the network, and $L \subseteq N \times N$ is the set of links. Given for each link l a number p_l, $0 \leq p_l \leq 1$. Where this p_l is the probability that link l is operative. For every link l we have $p_l = a^{n_l}$, where n_l is a non-negative integer, and $0 < a < 1$.

QUESTION: Are there two link-disjoint paths in the network P_1 and P_2, such that

both these paths connect node s to node t, and the probability that at least one of these paths is operative is larger than a given threshold p_{th}?

Notice that the existence of two link-disjoint paths between s and t can be established by a maximum flow computation, see [3], this is a polynomial operation.

Given P_1 and P_2, $\mathcal{P}(P_1, P_2)$, the probability that at least one of these two paths is operative is given by

$$\mathcal{P}(P_1, P_2) = \prod_{l \in P_1} p_l + \prod_{l \in P_2} p_l - \prod_{l \in P_1} p_l \prod_{l \in P_2} p_l. \tag{3.54}$$

If the p_l are rational numbers, i.e. if a is a rational number, then this problem is in *NP*. I.e. given P_1 and P_2 one can decide in polynomial time if $\mathcal{P}(P_1, P_2) \geq p_{th}$ for a given p_{th}.

We shall now prove that the problem is *NP*-hard. This will be done by showing that if we can solve the two link-disjoint path reliability problem, as defined above, then we can also solve another problem that already been proved to be *NP*-complete. The problem we are going to use is as follows.

Directed two link-disjoint path problem with distinct origins and destinations.

INSTANCE: Given a directed network $G' = (N', L')$, and given s_1, s_2, t_1, t_2, four distinguished nodes of this network.

QUESTION: Are there in this network two link-disjoint paths P_1 and P_2 such that P_1 connects s_1 to t_1 and P_2 connects s_2 to t_2?

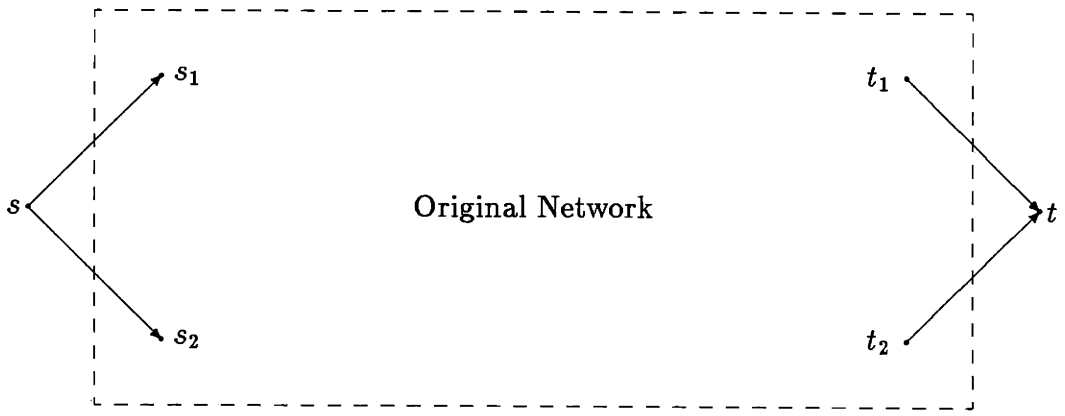


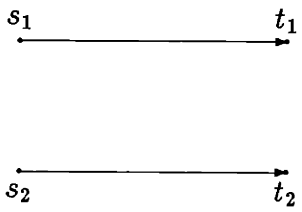
Figure 3.7: The original network with added links

For the *NP*-completeness proof of this problem see [5].

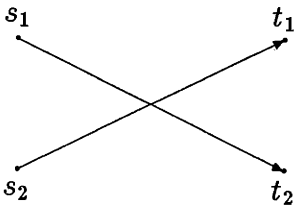
3.4.1 The Reduction

Given G' and s_1, s_2, t_1, t_2, we augment the network in the following way, see figure 3.7. Add two nodes s and t, also add four links: $\langle s, s_1 \rangle$, $\langle s, s_2 \rangle$, and $\langle t_1, t \rangle$, $\langle t_2, t \rangle$. Assign link probabilities to this augmented network as follows: $p_{\langle s,s_1 \rangle} = p_{\langle t_1,t \rangle} = (1/2)^{3n}$, and $p_l = 1/2$ for all other links, where n is the number of nodes in the network. Notice that if we use standard binary representation for the p_l, the number of bits necessary to specify them is $O(n^2)$.

Any pair (P_1, P_2) of link-disjoint paths joining s to t will necessarily use the four links that were added to the original network. Take, without loss of generality,



"Straight Paths"



"Crossed Paths"

Figure 3.8: 'Straight' and 'Crossed' paths

that P_1 uses $\langle s, s_1 \rangle$, and P_2 uses, necessarily, $\langle s, s_2 \rangle$. In the case that P_1 uses $\langle t_1, t \rangle$, and P_2 uses $\langle t_2, t \rangle$, we say that (P_1, P_2) is a *straight* pair. Otherwise, if P_1 uses $\langle t_2, t \rangle$, and P_2 uses $\langle t_1, t \rangle$, we say that (P_1, P_2) is a *crossed* pair of paths, see figure 3.8.

Lemma 3.5 *If $|N'| = n \geq 2$, i.e. if the network has at least two nodes, then for any straight pair (P_1^s, P_2^s), and any crossed pair (P_1^c, P_2^c), with the probabilities as assigned above, we have*

$$\mathcal{P}(P_1^s, P_2^s) > \mathcal{P}(P_1^c, P_2^c). \tag{3.55}$$

Proof. We have

$$\mathcal{P}(P_1^s, P_2^s) = \prod_{l \in P_1^s} p_l + \prod_{l \in P_2^s} p_l - \prod_{l \in P_1^s} p_l \prod_{l \in P_2^s} p_l \tag{3.56}$$

$$\geq \prod_{l \in P_2^s} p_l \geq (1/2)^{n-1+2} > (1/2)^{2n}. \tag{3.57}$$

Also,

$$\mathcal{P}(P_1^c, P_2^c) = \prod_{l \in P_1^c} p_l + \prod_{l \in P_2^c} p_l - \prod_{l \in P_1^c} p_l \prod_{l \in P_2^c} p_l \tag{3.58}$$

$$\leq \prod_{l \in P_1^c} p_l + \prod_{l \in P_2^c} p_l \leq p_{\langle s,s_1 \rangle} + p_{\langle t_1,t \rangle} \tag{3.59}$$

$$\leq 2(1/2)^{3n} = (1/2)^{3n-1} < (1/2)^{2n}. \tag{3.60}$$

Therefore,

$$\mathcal{P}(P_1^s, P_2^s) > (1/2)^{2n} > \mathcal{P}(P_1^c, P_2^c). \tag{3.61}$$

∎

Theorem 3.2 *The directed two link-disjoint path maximum reliability problem is NP-complete.*

Proof. Given G and s_i, t_i, $i = 1, 2$, augment this network and assign link probabilities as above. From Lemma 3.5 we know that there is a pair of link-disjoint paths P_1 and P_2 connecting s to t with $\mathcal{P}(P_1, P_2) \geq p_{th} = (1/2)^{2n}$, if and only if there exist two link-disjoint paths P_1' and P_2' so that P_1' connects s_1 to t_1 and P_2' connects s_2 to t_2.

∎

In the proof of lemma 3.5 all the probabilities have the form $p_l = (1/2)^{g(n)}$, where $n_l = g(n)$ is a polynomial in n. From this we have, see [6], following

Corollary 3.1 *The directed two link-disjoint paths maximum reliability problem is* strongly *NP-complete.*

Also, if we allow $p_l = 1$, lemma 3.5 has an analogue with $p_{\langle t_1,t\rangle} = p_{\langle s,s_1\rangle} = 1/2$, and $p_l = 1$ for the rest of the nodes. If the data are given in the non-logarithmic format, i.e. p_l is a rational number, the p_l are $O(1)$, and therefore a polynomial of n hence we have

Corollary 3.2 *The directed two link-disjoint path maximum reliability problem with standard input format is* strongly *NP-complete if perfectly reliable links are allowed.*

It is interesting that the existence of two link-disjoint paths if G is an *undirected* network is not an *NP*-complete problem. Y. Shiloach has given a polynomial algorithm, see [14].

3.4.2 Related Problems

The intermediate problem treated in section 3.3.4 can also be shown to be *NP*-complete when the network is not assumed to be acyclic. The proof is very similar to that given in the previous section. The problem is as follows.

Directed two link-disjoint weighted path problem.

INSTANCE: Given a directed network G as in 3.1 Given for each link l of this network a non-negative integer n_l. Given also a rational number $\lambda > 0$.

QUESTION: Are there two link-disjoint paths P_1 and P_2 such that

$$F(P_1, P_2) = \sum_{l \in P_1} n_l + \lambda \sum_{l \in P_2} n_l \leq f_{th} \tag{3.62}$$

for a given rational threshold value $f_{th} \geq 0$?

Notice that for $\lambda = q/r$, where q and r are positive integers this is equivalent to

$$F'(P_1, P_2) = r \sum_{l \in P_1} n_l + q \sum_{l \in P_2} n_l \leq q f_{th}. \tag{3.63}$$

It is easy to see that this is an NP problem. I.e., given two paths P_1 and P_2 it is possible to decide in polynomial time if 3.63 holds or not.

To prove that this is an NP-complete problem we reduce the same problem used in the previous section. The proof is very similar. Given a directed network G' and four distinguished nodes as before, we augment the network as in the previous section. Assign link numbers in the following way: $n_{\langle s,s_1 \rangle} = n_{\langle t_1,t \rangle} = 10n$, the rest of the links are assigned $n_l = 1$. Take $q = 10$ and $r = 1$. Then we have.

Lemma 3.6 *For any crossed pair of paths (P_1^c, P_2^c) so that both these paths connect s to t we have*

$$F'(P_1^c, P_2^c) > 50n. \tag{3.64}$$

Proof. We have

$$F'(P_1^c, P_2^c) = \sum_{l \in P_1^c} n_l + 10 \sum_{l \in P_2^c} n_l \geq 10(10n) = 100n. \tag{3.65}$$

∎

Lemma 3.7 *If there is a straight pair (P_a^s, P_b^s) so that both these paths connect s to t, then there is a straight pair (P_1^s, P_2^s) such that*

$$F'(P_1^s, P_2^s) < 50n. \tag{3.66}$$

Proof. Of the paths P_a^s and P_b^s one of them must use the link $\langle s, s_1 \rangle$, take that path as P_1^s, take the remaining one as P_2^s. Then we have

$$F'(P_1^s, P_2^s) = \sum_{l \in P_1^s} n_l + 10 \sum_{l \in P_2^s} \leq 2(10n) + (n-1) + 10(n-1+2) \tag{3.67}$$

$$< 30n < 50n. \tag{3.68}$$

∎

Now we can prove

Theorem 3.3 *The directed two link-disjoint weighted path problem is strongly NP-complete.*

Proof. Given G' and its four distinguished nodes, augment this network and assign the n_l as above. From the lemmata 3.6 and 3.7, we know that there is a

pair of link-disjoint paths P_1 and P_2 connecting s to t with $F(P_1, P_2) \leq f_{th} = 50n$, if and only if there exist two link-disjoint paths P_1' and P_2' such that P_1' connects s_1 to t_1 and P_2' connects s_2 to t_2. All coefficients in the problem are a polynomial in n, therefore the problem is proved to be strongly NP-complete.

■

Notice that from this we have that the more general problem of deciding if there exist link-disjoint paths P_1 and P_2 that connect s to t and such that $\sum_{l \in P_1} c_l^1 + \sum_{l \in P_2} c_l^2 \leq c_{th}$, for given non-negative integers c_{th}, c_l^i, $l \in L$, $i = 1, 2$, is also an NP-complete problem.

Also it is interesting that if in the proof of section 3.4. we change the definition of $\mathcal{P}(P_1, P_2)$ to read

$$\mathcal{P}(P_1, P_2) = \prod_{l \in P_1} p_l + \prod_{l \in P_2} p_l, \tag{3.69}$$

the proof still holds. Of course, now $\mathcal{P}(P_1, P_2)$ is not a probability any more. It can be interpreted to be the expected number of paths that are operative, or equivalently, the expected number of copies that arrive at node t. This also would have made sense as an alternative routing criterion. I.e. we could have started by trying to find two paths P_1 and P_2 such that if we were to send one copy of the message along each one of them the expected number of copies arriving at node t is maximized. As we just saw this problem is also NP-complete.

Chapter 4

Routing Two Messages Along Two Non-Disjoint Paths

4.1 Introduction

The problem we deal with in this chapter is similar to the one addressed in chapter three the difference is that here we dispense with the requirement that the paths followed by the two copies of the message be link-disjoint. This gives us more freedom in choosing these routes, offering thus, possibly, a higher probability that the message be received. The problems addressed in chapters two and three could be considered to be restrictions of this one.

In section 4.2 we prove that this is an *NP*-complete problem if the probabilities are given as rational numbers, and *NP*-hard if the probabilities are given in logarithmic format. In section 4.3 we give a pseudo-polynomial algorithm for its

solution when we have logarithmic inputs.

4.2 The Two Non-Disjoint Path Maximum Reliability Problem is *NP*-Complete

The problem is as follows.

Two path maximum reliability problem.

INSTANCE: Given a network $G = (N, L)$, where N is the set of nodes of the network, and $L \subseteq N \times N$ is the set of links. Given, for each link l, a rational number p_l, $0 \leq p_l \leq 1$, where p_l is the probability that link l is operative. Also given is a rational threshold p_{th}.

QUESTION: Are there two simple paths P_1 and P_2, both joining node s to node t, such that the probability that at least one of the two paths is operative is greater or equal than p_{th}.

We prove that this is an *NP*-hard problem by considering a specific topology, see figure 4.1, we show that by solving the problem for this network we solve the complementary path problem, considered in chapter two, for the same network; this proves that the problem is reduced to partition.

Consider the network of figure 4.1. This is the same network as in figure 2.2. As in chapter two we define $p_{l_j^1} = p_{l_j^0}^2$; this implies $p_{l_j^0} > p_{l_j^1}$.

Given the tuple (P_1, P_2) consisting of two paths that connect nodes s and t,

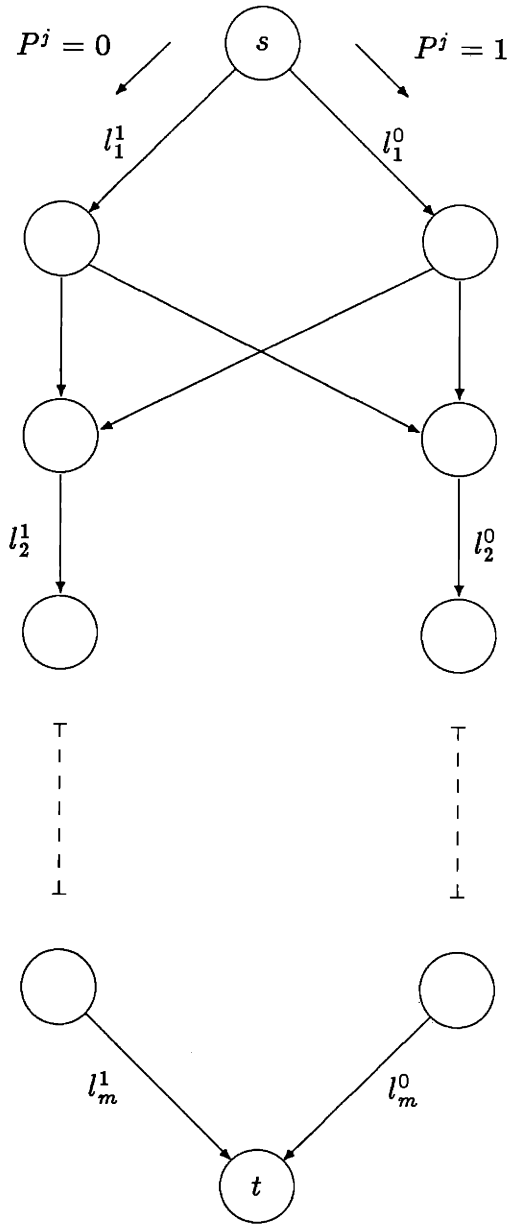


Figure 4.1:

let $\mathcal{P}(P_1, P_2)$ be the probability that at least one of the two paths is operative. A path P in this network can be characterized by specifying whether P includes l_j^0 or l_j^1 for $j = 1, \ldots, m$. To a given path P we now associate m binary variables $P^j, j = 1, \ldots, m$. Where $P^j = 0$, if $l_j^0 \in P$, $P^j = 1$, if $l_j^1 \in P$. We prove two lemmas that shall then be used to prove the main result.

Lemma 4.1 *If (P_1, P_2) is the optimum solution for the two non-disjoint paths maximum reliability problem, then either $P_1^j = 0$ or $P_2^j = 0$, for $j = 1, \ldots, m$. The 'or' in the previous statement is not exclusive.*

Proof. Suppose this is not true. I.e. for some j we have that $P_1^j = P_2^j = 1$; this means that the link l_j^1 is in both P_1 and P_2. Therefore,

$$\begin{aligned} \mathcal{P}(P_1, P_2) &= \prod_{l \in P_1} p_l + \prod_{l \in P_2} p_l - \prod_{l \in P_1 \cup P_2} p_l \\ &= p_{l_j^1} p_a + p_{l_j^1} p_b - p_{l_j^1} p_c, \end{aligned} \tag{4.70}$$

for some $0 < p_a, p_b, p_c < 1$. If we now make $P_1^j = 0$, changing thus P_1 into P_1' which includes l_j^0 instead of l_j^1, we have

$$\mathcal{P}(P_1', P_2) = p_{l_j^0} p_a + p_{l_j^1} p_b - p_{l_j^0} p_{l_j^1} p_c. \tag{4.71}$$

Therefore,

$$\mathcal{P}(P_1', P_2) - \mathcal{P}(P_1, P_2) = (p_{l_j^0} - p_{l_j^1}) p_a + (1 - p_{l_j^0} p_{l_j^1} p_c) > 0. \tag{4.72}$$

Which implies

$$\mathcal{P}(P_1', P_2) > \mathcal{P}(P_1, P_2). \tag{4.73}$$

Which is a contradiction.

∎

Lemma 4.2 *In an optimum solution* (P_1, P_b) *one of the two paths, say* P_1*, is such that* $P_1^j = 0$ *for* $j = 1, \ldots, m$.

Proof. Suppose that the tuple (P_a, P_b) is an optimum solution, and that for neither of these two paths are all $P^j = 0$. From the previous lemma we have that $P_a^j = P_b^j = 1$ cannot occur for any j. Consider now the pair (P_1, P_2) defined by

$$P_1^j = 0 \quad \text{for} \quad j = 1, \ldots, m$$

$$P_2^j = \begin{cases} 1 & \text{if } P_b^j = 0 \text{ and } P_a^j = 1; \\ P_b^j & \text{otherwise.} \end{cases}$$

This means

$$P_a^j = 1, P_b^j = 0 \quad \Rightarrow \quad P_1^j = 0, P_2^j = 1$$

$$P_a^j = 0, P_b^j = 1 \quad \Rightarrow \quad P_1^j = 0, P_2^j = 1$$

$$P_a^j = 0, P_b^j = 0 \quad \Rightarrow \quad P_1^j = 0, P_2^j = 0. \tag{4.74}$$

From this construction we have that $P_1 \cup P_2 = P_a \cup P_b$; we also have

$$\prod_{l \in P_1} p_l \prod_{l \in P_2} p_l = \prod_{l \in P_a} p_l \prod_{l \in P_b} p_l. \tag{4.75}$$

We want to prove that $\mathcal{P}(P_1, P_2) > \mathcal{P}(P_a, P_b)$, or equivalently that

$$\mathcal{P}(P_1, P_2) - \mathcal{P}(P_a, P_b) > 0. \tag{4.76}$$

We have

$$\begin{aligned} \mathcal{P}(P_1, P_2) - \mathcal{P}(P_a, P_b) &= \prod_{l \in P_1} p_l + \prod_{l \in P_2} p_l - \prod_{l \in P_1 \cup P_2} p_l \\ &\quad - \left(\prod_{l \in P_a} p_l + \prod_{l \in P_b} p_l - \prod_{l \in P_a \cup P_b} p_l \right) \\ &= \prod_{l \in P_1} p_l + \prod_{l \in P_2} p_l - \left(\prod_{l \in P_a} p_l + \prod_{l \in P_b} p_l \right). \end{aligned} \tag{4.77}$$

Define now

$$\begin{aligned} x_1 &= \prod_{l \in P_1} p_l, \qquad x_2 = \prod_{l \in P_2} p_l, \\ x_a &= \prod_{l \in P_a} p_l, \qquad x_b = \prod_{l \in P_b} p_l. \end{aligned} \tag{4.78}$$

Take, without loss of generality, $x_a \geq x_b$. We also have, by construction, $x_1 > x_a$

and $x_1 \geq x_2$. Define $c = x_1 x_2 = x_a x_b$. Then,

$$\begin{aligned} P(x_1, x_2) - P(x_a, x_b) &= (x_1 + \frac{c}{x_1}) - (x_a + \frac{c}{x_a}) \\ &= f(x_1) - f(x_a). \end{aligned} \tag{4.79}$$

Where $f(x) = x + \frac{c}{x}$. For $x \geq \sqrt{c}$, $f(x)$ is an increasing function. We have $x_1 \geq x_2$ and $c = x_1 x_2$, and this guarantees $x_1 \geq \sqrt{c}$. Similarly we have that $x_a \geq \sqrt{c}$. Therefore

$$x_1 > x_a \Rightarrow f(x_1) > f(x_a). \tag{4.80}$$

Which implies $P(x_1, x_2) > P(x_a, x_b)$.

■

We can now prove

Theorem 4.1 *The two non-disjoint path maximum reliability problem is NP-complete.*

Proof. The problem is easily seen to be in *NP*, i.e. for given P_1 and P_2 it can be determined polynomially if the probability that at least one of the two paths is operative is greater or equal than p_{th}. From the two previous lemmata we have that we can always take P_1 to be fixed, namely $P_1^j = 1, j = 1, \ldots, m$. The problem is then reduced to the particular case of the complementary case problem that was used in chapter two for the reduction to partition.

We also have the following.

Theorem 4.2 *The two non-disjoint path maximum reliability problem with logarithmic inputs is NP-hard.*

It is not known if this problem is in *NP*.

4.3 The Algorithm

In this section we present a pseudo-polynomial algorithm for the problem described in section 4.2. The algorithm is basically dynamic programming. Given $G = (N, L)$, a directed, acyclic, stratified network, we generate $G^2 = (N^2, L^2)$, the corresponding squared network as in section 3.3.2, with the difference that we omit the requirement that in 3.3.2 guarantees path disjointedness. That is, if there is a node $k \in S_i$, and a node $l \in S_{i+1}$, with a link $\langle k, l \rangle \in L$, then, in G^2 there will be a link $\langle kk, ll \rangle$. To every path connecting node ss to node tt in G^2 correspond two, not necessarily disjoint, paths in G, both connecting nodes s and t.

A link $\langle ik, rs \rangle \in G^2$ is labeled by three coefficients: $c^1_{\langle ik,rs \rangle} = n_{\langle i,r \rangle}$; $c^2_{\langle ik,rs \rangle} = n_{\langle k,s \rangle}$; and $c^3_{\langle ik,rs \rangle} = 0$, unless $i = k$ and $r = s$, in which case $c^3_{\langle kk,ss \rangle} = c^1_{\langle kk,ss \rangle} = c^2_{\langle kk,ss \rangle} = n_{\langle k,s \rangle}$.

Given a path P in G^2, the probability that at least one of the two corresponding

paths, P_1 and $P_2 \in G$, is operative is given by

$$\begin{aligned} \mathcal{P}(P_1, P_2) &= a^{\sum_{l\in P_1} n_l} + a^{\sum_{l\in P_2} n_l} - a^{\sum_{l\in P_1} n_l + \sum_{l\in P_2} n_l - \sum_{l\in P_1\cap P_2} n_l} \\ &= a^{\sum_{l\in P'} c_l^1} + a^{\sum_{l\in P'} c_l^2} - a^{\sum_{l\in P'} c_l^1 + \sum_{l\in P'} c_l^2 - \sum_{l\in P'} c_l^3}. \end{aligned} \tag{4.81}$$

Suppose copy one of the message is sent to node $i \in S_k$ following fixed path P_1'. Similarly, copy two is sent to node $j \in S_k$ following fixed path P_2'. Define now, $n_1 = \sum_{l\in P_1'} n_l$; $n_2 = \sum_{l\in P_2'} n_l$; and $n_3 = \sum_{l\in P_1'\cap P_2'} n_l$. Or equivalently, if P' is the path in G^2 corresponding to P_1' and P_2', $n_1 = \sum_{l\in P'} c_l^1$;$n_2 = \sum_{l\in P'} c_l^2$; and $n_3 = \sum_{l\in P'} c_l^3$.

Let P_1'' and P_2'' be the paths the two copies should follow if they were to start from nodes i and j to node t, so that the probability that at least one of them arrives is maximized. Notice that the choice of these paths, the paths the two copies would follow after the k^{th} layer if they were to make it that far before encountering a failed link, depends on P_1' and P_2', the paths along which the copies were routed towards the k^{th} layer. For example, is P_1' is much more reliable than P_2', the optimal choice of P_1'' and P_2'' would typically favor P_1'', as it is more probable that copy one has actually reached the k^{th} layer. The choice of these paths depends only on n_1, n_2, and n_3.

Define

$$\nu_{ij}^1(n_1, n_2, n_3) = \sum_{l\in P_1''} n_l,$$

$$\nu_{ij}^2(n_1,n_2,n_3) = \sum_{l\in P_2''} n_l,$$

$$\nu_{ij}^3(n_1,n_2,n_3) = \sum_{l\in P_1''\cap P_2''} n_l. \tag{4.82}$$

Define $\pi_{ij}(n_1,n_2,n_3)$ as the maximum probability that at least one of these two copies reaches node t. Then, given that node $ij \in S_k^2$, we have

$$\begin{aligned}\pi_{ij}(n_1,n_2,n_3) = \max_{\langle ij,kl\rangle\in L^2}\{ & a^{n_1+c^1_{\langle ij,kl\rangle}+\nu^1_{kl}(n_1+c^1_{\langle ij,kl\rangle},n_2+c^2_{\langle ij,kl\rangle},n_3+c^3_{\langle ij,kl\rangle})} \\ + & a^{n_2+c^2_{\langle ij,kl\rangle}+\nu^2_{kl}(n_1+c^1_{\langle ij,kl\rangle},n_2+c^2_{\langle ij,kl\rangle},n_3+c^3_{\langle ij,kl\rangle})} \\ - & a^{\begin{array}{l}(n_1+c^1_{\langle ij,kl\rangle}+\nu^1_{kl}(n_1+c^1_{\langle ij,kl\rangle},n_2+c^2_{\langle ij,kl\rangle},n_3+c^3_{\langle ij,kl\rangle}) \\ +n_2+c^2_{\langle ij,kl\rangle}+\nu^2_{kl}(n_1+c^1_{\langle ij,kl\rangle},n_2+c^2_{\langle ij,kl\rangle},n_3+c^3_{\langle ij,kl\rangle}) \\ -(n_3+c^3_{\langle ij,kl\rangle}+\nu^3_{kl}(n_1+c^1_{\langle ij,kl\rangle},n_2+c^2_{\langle ij,kl\rangle},n_3+c^3_{\langle ij,kl\rangle})))\end{array}}\}\end{aligned} \tag{4.83}$$

Let $mn \in S_{k+1}^2$ be such that $\langle ij,mn\rangle$ is the link that achieves the maximum in 4.83. Then we have, for $p = 1,2,3$

$$\nu_{ij}^p(n_1,n_2,n_3) = c^p_{\langle ij,mn\rangle} + \nu^p_{mn}(n_1 + c^1_{\langle ij,mn\rangle}, n_2 + c^2_{\langle ij,mn\rangle}, n_3 + c^3_{\langle ij,mn\rangle}). \tag{4.84}$$

Equations (4.83) and (4.84) provide the means to calculate the optimal routing for two copies of a message. The maximum of $P(P_1,P_2)$, as in equation (3.36), is given by

$$P(P_1,P_2) = \pi_{ss}(0,0,0). \tag{4.85}$$

The paths can be obtained by recording the decisions taken when using equation (4.84). Notice that

$$\begin{aligned} \pi_{ss}(n_1, n_2, n_3) &= a^{n_1} + a^{n_2} - a^{n_1+n_2-n_3}, \\ \nu^i_{ss}(n_1, n_2, n_3) &= n_i, \quad i = 1, 2, 3. \end{aligned} \tag{4.86}$$

We next look at the performance of this algorithm. If M_k is an upper bound on the maximum length a path can have from node ss to a node in S_k, where the length of a link l is given by n_l, if we know the $\nu^n_{ij}(n_1, n_2, n_3)$ for $n = 1, 2, 3$, and $0 \leq n_1, n_2, n_3 \leq M_{k+1}$, and $i, j \in S_{k+1}$, i.e. $ij \in S^2_{k+1}$, then it is possible to calculate the $\nu^n_{ij}(n_1, n_2, n_3)$ for $n = 1, 2, 3$; and $0 \leq n_1, n_2, n_3 \leq M_k$; $ij \in S^2_k$.

This dynamic programming solution of the problem presents the usual problem of state space explosion. Any attempt at using it for practical application would require careful programming. In particular, one way to optimize the performance of this approach is to make the M_k as tight as possible. Notice that this approach could also be used for the case of a link-disjoint path as in section 3.2. Also, by a simple device as in the introduction, we can treat the case of *node*-disjoint paths.

Chapter 5

Conclusions and Recommendations for Further Research

In this thesis we have presented several ways of selecting two s-t-paths for simultaneous transmission of two copies of one message in a data communication network. In section 5.1 we compare how these approaches perform in practice. In section 5.2 we look at suggestions for further research.

5.1 Comparing Approaches

We have chosen three of the route selection procedures, all of them using logarithmic input. These three have been selected because they can be implemented

efficiently:

- Complementary Path Algorithm, chapter 2, (`Compath`).
- Finding two disjoint s-t-paths P_1 and P_2 such that the probability that both of them (P_1 *and* P_2) are operative is maximized, (`Twopathand`).
- Finding two disjoint s-t-paths P_1 and P_2 such that the probability that at least one of them (P_1 *or* P_2) is operative is maximized, (`Twopathor`).

These three methods have been programmed. We have some limited experience comparing their performance in actual examples. We next comment on each of them separately and then compare their behaviour.

`Compath`: The algorithm given in chapter two for this problem works on unrestricted networks, it is pseudopolynomial. But, if for every link l we take $n_l < n$, where $p_l = a^{n_l}$, and 4n4 is a positive integer, then the algorithm becomes polynomial in $|L|$ and $|N|$. Specifically, it takes $O(n|L|^2)$ operations. For a given input network, given basic path P_1, given the n_l's, and given the basic operational probability a the solution is dependent on the value of this a, as follows. The closest a is to one the more it is likely that P_2, the complementary path, be disjoint from P_1. The closest a is to zero the more likely it is that P_1 and P_2 will share links.

`Twopathand`: the algorithm given in [16] consists of two consecutive applications of Dijkstra's shortest path method with the addition of an $O(|L|)$ step between these two applications. Therefore this is a polynomial procedure. It works on unrestricted networks. The solution is independent of the basic operative probability

a. As a by-product we obtain a shortest path of the network which can be used as part of the input, for Compath, the fixed path (P_1).

Twopathor: the pseudopolynomial algorithm given in chapter three applies only to acyclic networks. The algorithm works by finding all pairs of non-dominated extreme disjoint s-t-paths (P_1, P_2). Once this set has been found it can be calculated which pair attains the optimum for the given value of a.

To test the comparative performance of these algorithms we generated acyclic networks with random values of n_l, $p_l = a^{n_l}$, for the links of the network, n_l takes the values $1, 2, 3, 4, 5$ with equal probability. The generated networks are characterized by the number of nodes they have, and by a parameter called $width$, this parameter is such that there exists a link connecting node i to node j if and only if $i < j \leq j + width$. Here we present result for six networks:

- net1: consisting of 10 nodes, with a width of 3.
- net2: consisting of 15 nodes, with a width of 4.
- net3: consisting of 20 nodes, with a width of 4.
- net4: consisting of 25 nodes, with a width of 4.
- net5: consisting of 40 nodes, with a width of 5.
- net6: consisting of 30 nodes, with a width of 6.

Tables 5.1, 5.2 and 5.3 present the value of the probability that at least one of the two messages arrives at the destination node for the pair of paths provided by

a=0.99	Twopathand	Twopathor	Compath
net1	.9910	.9910	.9910
net2	.9909	.9909	.9909
net3	.9905	.9905	.9905
net4	.9756	.9756	.9756
net5	.9754	.9754	.9754
net6	.9905	.9905	.9905

Table 5.1: Message arrival probability for $a = 0.99$

the algorithms indicated in the columns. The source node is node number 1, the destination node is the highest numbered node of the network. The three tables correspond to different values of the basic probability a. We use a most reliable path of the network as the fixed path in the input for algorithm Compath.

Notice that even though the paths found by algorithm Twopathand maximize the probability that *both* path are operative, the value shown in the tables is the probability that *at least one* of them is operative.

Our computational experience suggests that the solutions obtained by the algorithm Twopathand are very close to those given by algorithm Twopathor, in the cases shown here they happen to be always the same. This can be explained by recalling that the solution given by algorithm Twopathor is always a non-dominated extreme pair of link-disjoint paths of the network, which is also the case for algorithm Twopathand, though not necessarily the same pair, if the network is such

a=0.9	Twopathand	Twopathor	Compath
net1	.5976	.5976	.6416
net2	.5796	.5796	.5796
net3	.5752	.5752	.6123
net4	.3131	.3131	.3388
net5	.3076	.3076	.3139
net6	.5752	.5752	.5941

Table 5.2: Message arrival probability for $a = 0.9$

a=0.5	Twopathand	Twopathor	Compath
net1	.0079	.0079	.0146
net2	.0024	.0024	.0038
net3	.0040	.0040	.0048
net4	.0000	.0000	.0000
net5	.0000	.0000	.0000
net6	.0040	.0040	.0057

Table 5.3: Message arrival probability for $a = 0.5$

that there exists only one non-dominated extreme pair of link-disjoint paths the solution provided by these two algorithms is necessarily the same. From the examples here presented one could infer that the family of generated networks has this property for sufficiently large values of the number of nodes and the *width* of the networks.

The solutions given by algorithm `Twopathand` are also close to those given by algorithm `Compath`, particulary when a is close to one, see the tables for $a = 0.99$. When the value of a decreases, see table 5.3, algorithm `Compath` does produce better solutions. This suggests that in this case, 'low' a, sharing links is necessary to improve the probability that at least one of the two copies of the message arrives at the destination node.

The numerical results obtained indicate that the possible improvements to be gained by using algorithm `Twopathor` might not justify the increased computational expense. For a close to one the solutions obtained by algorithm `Twopathand` are very close to that obtained by algorithm `Compath` when, as here, a most realiable path of the network is used as the fixed path (P_1).

5.2 Suggestions for Further Research

- Some complexity issues remain unanswered:
  - Is the problem treated in chapter three (finding two disjoint s-t-paths that maximize the probability that at least on of them is operative) *NP*-complete when the network is acyclic?

- Is the problem treated in chapter four (finding two not necessarily disjoint s-t-paths that maximize the probability that at least one of them is operative) strongly *NP*-complete for an unrestricted network, i.e. a network that may contain cycles.

- From a more general perspective the subjetct of "robustness" for algorithms in data-communication networks seems to be open. The design of flow-control and routing algorithms that will behave reasonably in the presence of unreliable input data seems to be deserving of attention.

Bibliography

[1] M. O. Ball, and J.S. Provan, "Calculating Bounds on Reachability and Connectedness in Stochastic Networks," *Networks*, Vol. 13 (1983), pp. 253-278.

[2] R. Chandrasekaran, and A. Tamir, "Polynomial Testing of the Query 'Is $a^b \geq c^d$?' with Application to Finding a Minimal Cost Reliability Ratio Spanning Tree," *Discrete Applied Mathematics*, Vol. 9 (1984), pp. 117-223.

[3] E. A. Dinic, "Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation," *Soviet Mathematics Doklady*, Vol. 11 (1970), pp. 1277-1280.

[4] S. G. Finn, "Resynch Procedures and a Fail-Safe Network Protocol," *IEEE Transactions on Communications*, Vol. COM-27 (1979), pp. 840-845.

[5] S. Fortune, J. Hopcroft, and J. Wyllie, "The Directed Subgraph Homeomorphism Problem," *Theoretical Computer Science*, Vol. 10 (1980), pp. 111-121.

[6] M. R. Garey, and D. S. Johnson, "Computers and Intractability," W. H. Freeman and Company, San Francisco, 1979.

[7] A. Itai, Y. Perl, and Y. Shiloach, "The Complexity of Finding Maximum Disjoint Paths with Length Constrains," *Networks*, Vol. 12 (1982), pp. 277-286.

[8] R. M. Karp, "Reducibility among Combinatorial Problems," in R. E. Miller and J. W. Thatcher (eds.), *Complexity of Computer Computations*, pp. 85-103, Plenum Press, New York, 1972.

[9] E. L. Lawler, "Combinatorial Optimization: Networks and Matroids," Holt, Rinehard and Winston, New Yor, 1976.

[10] K. G. Murty, "Linear Programming," John Wiley and Sons, New York, 1983.

[11] J. S. Provan, and M. O. Ball, "The Complexity of Counting Cuts and of Computing the Probability that a Graph is Connected," *SIAM Journal on Computing*, Vol. 12 (1983), pp. 777-788.

[12] J. S. Provan, and M. O. Ball, "Computing Network Reliability in Time Polynomial in the Number of Cuts," *Operations Research*, Vol. 32 (1984), pp. 516-526.

[13] J. A. Roskind, "Edge Disjoint Spanning Trees and Resynchronization in Data Communication Networks," Ph.D. Thesis, M.I.T., and M.I.T. Laboratory for Information and Decision Systems Report LIDS-TH-1332, Oct. 1983.

[14] Y. Shiloach, "A Polynomial Solution to the Undirected Two Path Problem," *Journal of the ACM*, Vol. 27 (1980), pp. 445-456.

[15] J. M. Spinelli, "Broadcasting Topology and Routing Information in Computer Networks," S.M. Thesis, M.I.T., and M.I.T. Laboratory for Information and Decision Systems Report LIDS-TH-1470, May 1985.

[16] J. W. Suurballe, "Disjoint Paths in a Network," *Networks*, Vol. 4 (1974), pp. 125-145.

[17] A. S. Tanenbaum, "Computer Networks," Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.

[18] L. G. Valiant, "The Complexity of Computing the Permanent," *Theoretical Computer Science*, Vol. 8 (1977), pp. 189-201.

[19] L. G. Valiant, "The Complexity of Enumeration and Reliability Problems," *SIAM Journal on Computing*, Vol. 8 (1979), pp. 410-421.