

# **On The Inverse Shortest Path Length Problem**

A Thesis  
Presented to  
The Academic Faculty

by


**Cheng-Huang Hung**

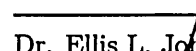
In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy

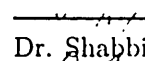
School of Industrial and Systems Engineering  
Georgia Institute of Technology  
December 2003

# On The Inverse Shortest Path Length Problem

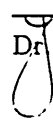
Approved by:

  
Dr. Joel S. Sokol, Committee Chair

  
Dr. Ellis L. Johnson

  
Dr. Shabbir Ahmed

  
Dr. Samer Takriti

  
Dr. Özlem Ergun

Date Approved 10/30/03

*To my Mom, Jim, Sophie, and Susan  
for their unwavering love and support*

## ACKNOWLEDGEMENTS

I first acknowledge my advisor, Joel Sokol. Joel guided me into the right direction, gave valuable feedback and comments, and was supportive during every step of this journey. He is not only my advisor in academics, but also an instructor of my career and life. Joel made this journey unforgettable and guided me through every bottleneck. His brilliant insight and inspiration helped me to explore this new frontier of research. Without Joel's patient and guidance, I could not reach the end of this journey. Joel made this impossible mission possible.

I must thank my thesis committee members, Shabbir Ahmed, Özlem Ergun, Ellis Johnson, and Samer Takriti. They looked through every word I wrote and gave me useful comment and suggestions for future research. I especially thank Shabbir and Özlem for their valuable time in numerous discussions to give me invaluable advice and supervision during this research. Their involvement made this thesis more complete and solid.

I have to thank my professors in optimization, including Ellis Johnson, George Nemhauser, Renato Monteiro, Martin Savelsbergh, and John Vande Vate. Their teaching built up my foundation in optimization. I also thank Anthony Hayter, Earl Barnes, and Craig Tovey for their support when I worked with them.

Other members of the Georgia Tech community who have been helpful include Paul Brooks, Junxia Chang, Jin-Hwa Song, Se-Kyoung Oh, Pamela Morrison, Patti Parker, Valarie DuRant-Modeste, and Gary Parker. I would like to thank CPLEX, a division of ILOG, for its software support.

I am thankful for many good people from Taiwan who helped me in Atlanta over the years, particularly Chien-Tai & Chen-Chen, James & Tina, Jack & Miawshian, Ivy & George, Wen-Chih, Ke-Hau, Po-Hsun, and I-Lin.

Special thanks go out to Anthony Hayter and his wife, Miki. Their friendship made my graduate school experience unforgettable.

Most importantly, I have to thank my family. My Mom's love and encouragement always surround me in this lonely journey. My dear wife, Susan, raised our two lovely children alone in Taiwan. She is not only the mother of my children, but also played the role of a father in these years. Her love and support accompanied with me at each step of this journey. My lovely children, Jim and Sophie, lived a life without a father around. Their sacrifice and support made this happen. If there is any contribution in this thesis, it belongs to them and not me. Without the support of my family, nothing would have been accomplished. I love them so much. I would not leave them at any moment of the rest of my life.

# TABLE OF CONTENTS

<b>DEDICATION</b> . . . . .	<b>iii</b>
<b>ACKNOWLEDGEMENTS</b> . . . . .	<b>iv</b>
<b>LIST OF TABLES</b> . . . . .	<b>ix</b>
<b>LIST OF FIGURES</b> . . . . .	<b>xi</b>
<b>SUMMARY</b> . . . . .	<b>xiii</b>
<b>I INTRODUCTION</b> . . . . .	<b>1</b>
1.1 Motivation of Inverse Problems . . . . .	1
1.2 Motivation of Inverse Shortest Path Problem . . . . .	2
1.3 Motivation and Problem Statement of Inverse Shortest Path Length Problem . . . . .	3
1.3.1 ISPL in Telecommunication Pricing . . . . .	3
1.3.2 Statement of ISPL . . . . .	4
1.4 Thesis Objective . . . . .	6
<b>II LITERATURE REVIEW</b> . . . . .	<b>7</b>
2.1 Inverse Solution Optimization Problem . . . . .	7
2.2 Methods for Solving Inverse Solution Optimization Problems . . . . .	8
2.3 Inverse Objective Value Optimization Problems and Their Complexity . . . . .	12
<b>III COMPLEXITY OF ISPL</b> . . . . .	<b>15</b>
3.1 Formal Statement . . . . .	15
3.2 Summary of Previous ISPL Complexity Results . . . . .	17
3.3 Complexity Results . . . . .	18
3.3.1 Tree and Complete Graph ISPL . . . . .	18
3.3.2 Cycle ISPL . . . . .	25
3.3.3 ISPL Complexity and the Number of Commodities . . . . .	37
3.4 Summary . . . . .	39
<b>IV ALGORITHMS</b> . . . . .	<b>41</b>
4.1 Heuristic Ideas . . . . .	41
4.1.1 Spanning Tree Enumeration Scheme . . . . .	41

4.1.2	Path Iteration Algorithm . . . . .	42
4.2	ISPL Algorithms . . . . .	45
4.2.1	Shortest Path Subproblem . . . . .	45
4.2.2	Minimization of Infeasibility . . . . .	46
4.2.3	Cost Perturbation Subproblem . . . . .	47
4.3	Description of Algorithms . . . . .	48
4.3.1	Algorithm 0 . . . . .	49
4.3.2	Algorithm 1 . . . . .	49
4.3.3	Algorithms 2 & 3 . . . . .	51
4.3.4	Algorithms 4 & 5 . . . . .	51
4.4	Properties of Algorithms . . . . .	59
4.4.1	Initialization of the Algorithm . . . . .	59
4.4.2	Termination of the Algorithm . . . . .	60
4.4.3	Perturbation Benefit . . . . .	61
4.4.4	Monotonicity Properties . . . . .	65
4.5	Worst-Case Bounds . . . . .	68
4.6	Summary . . . . .	79
<b>V</b>	<b>COMPUTATIONAL RESULTS . . . . .</b>	<b>81</b>
5.1	Random ISPL Instance Generation . . . . .	81
5.1.1	General Framework . . . . .	82
5.1.2	Random ISPL 1 . . . . .	83
5.1.3	Random ISPL 2 . . . . .	83
5.1.4	Random ISPL 3 . . . . .	84
5.1.5	Random Instance Generation Results . . . . .	85
5.2	Sample Algorithm Performance . . . . .	88
5.3	Performance of Algorithms on Random Instances . . . . .	90
5.3.1	Intermediate-difficulty Instances . . . . .	90
5.3.2	Hard Instances . . . . .	99
5.4	Telecommunication Example . . . . .	104
5.5	Summary . . . . .	109

<b>VI CONCLUSIONS AND FUTURE RESEARCH . . . . .</b>	<b>111</b>
6.1 Conclusions . . . . .	111
6.2 Future Research . . . . .	112
<b>REFERENCES . . . . .</b>	<b>114</b>
<b>VITA . . . . .</b>	<b>117</b>



## LIST OF TABLES

1	Summary of the complexity result . . . . .	39
2	Percentage of similarity in shortest path set when $(p1,p2)=(0.15,0.7)$ . . . .	86
3	Percentage of similarity in the shortest path set for switching edge type by p1	86
4	Similarity in SP path sets when $( N , E , K ,p1,p2)=(30,50,435,0.5,0.4)$ . .	87
5	Similarity in SP path sets when $( N , E , K ,p1,p2)=(30,100,435,0.5,0.4)$ . .	87
6	Similarity in SP path sets when $( N , E , K ,p1,p2)=(30,200,435,0.5,0.4)$ . .	87
7	Similarity in SP path sets when $( N , E , K ,p1,p2)=(30,300,435,0.5,0.4)$ . .	88
8	Similarity in SP path sets when $( N , E , K ,p1,p2)=(30,435,435,0.5,0.4)$ . .	88
9	Average performance for intermediate-difficulty ISPL instances (%) . . . . .	91
10	Standard deviation of the performance for intermediate-difficulty ISPL (%)	91
11	Average performance for algorithms under 95 and 99 percent certainly (%)	91
12	Lower bound of probability of solving intermediate-difficult ISPL within 3 and 5 percent . . . . .	93
13	t-value comparing the performance starting with the minimum cardinality path . . . . .	94
14	t-value comparing the performance starting with random paths . . . . .	94
15	t-value comparing two initial paths for all algorithms . . . . .	94
16	Average number of iterations for intermediate-difficulty ISPL . . . . .	95
17	Standard deviation of the number of iterations for intermediate-difficulty ISPL	95
18	Average number of iterations to converge for intermediate-difficulty ISPL .	95
19	t-value comparing the speed of convergence between different initial paths .	96
20	Algorithm performance: (ISPL instance I, the minimum cardinality paths)	97
21	Algorithm performance: (ISPL instance I, uniform random paths) . . . . .	98
22	Algorithm performance: (ISPL instance II, the minimum cardinality paths)	98
23	Algorithm performance: (ISPL instance II, uniform random paths) . . . . .	99
24	Algorithm performance: (ISPL instance III, the minimum cardinality paths)	100
25	Algorithm performance: (ISPL instance III, uniform random paths) . . . . .	100
26	Mean and standard deviation of the performance for hard ISPL (%) . . . . .	100
27	The performance of algorithms for hard ISPL (%) . . . . .	100

28	Lower bound of probability of solving hard ISPL within 3 and 5 percent . . .	101
29	t-value comparing the performance of algorithms for hard ISPL . . . . .	102
30	Algorithm performance for hard ISPL in different scenario . . . . .	102
31	Mean and standard deviation of iterations to converge for hard ISPL . . . .	103
32	Average iterations for convergence for hard ISPL under 95 and 99 percent .	103
33	Algorithm performance for Tyco (%) . . . . .	106
34	Algorithm performance for Level 3 (%) . . . . .	106
35	Algorithm performance for Global Crossing (%) . . . . .	107
36	Two infeasibilities of Telecom ISPL instances . . . . .	108
37	Adjusted performance by infeasibility bounds . . . . .	109

# LIST OF FIGURES

1	Distance graph must be a Euclidean graph . . . . .	19
2	Partition of vertices in a complete graph without edge $(u, v)$ . . . . .	22
3	If $K_n \setminus \{(u, v)\}$ is feasible, then $\exists P_{uv}$ such that $ P_{uv}  = 2$ . . . . .	22
4	Original Cycle ISPL to Induced Cycle ISPL . . . . .	26
5	Two commodities have disjoint paths on the general cycle ISPL instance . .	28
6	Logic graph of general cycle ISPL . . . . .	29
7	Restricted cycle has 3 disjoint paths . . . . .	30
8	Logic graph of restricted cycle ISPL . . . . .	30
9	Logical graph with clique of size 5 has unique shortest path set . . . . .	31
10	A has no two disjoint paths . . . . .	32
11	Largest restricted cycle ISPL with no 3 disjoint paths . . . . .	33
12	All commodities have 3 disjoint paths . . . . .	34
13	Decompose cycle nodes into two sets . . . . .	36
14	Matching origins and destinations for commodities . . . . .	37
15	Perfect matching of origins and destinations with no cross edges . . . . .	37
16	The union of two commodities path exists cycle . . . . .	38
17	Only one spanning tree is feasible . . . . .	42
18	Example of the unique feasible solution is a cycle . . . . .	42
19	Example with exponentially many constraints . . . . .	43
20	Constraint generating scheme flow chart . . . . .	44
21	Algorithm 0 . . . . .	50
22	Algorithm 1 . . . . .	52
23	Algorithm 2 . . . . .	54
24	Algorithm 3 . . . . .	55
25	Algorithm 4 . . . . .	57
26	Algorithm 5 . . . . .	58
27	Perturb the cost to reach a better solution . . . . .	62
28	Linear function of length of a path . . . . .	63
29	Shortest path length function for one commodity is piecewise linear . . . .	64

30	Number of constraints generated is neither convex nor concave . . . . .	64
31	Example of cost perturbation does not work . . . . .	65
32	Feasible cost of ISPL . . . . .	65
33	Inner loop and outer loop . . . . .	66
34	Feasible ISPL instance with 2 commodities . . . . .	71
35	Feasible ISPL solution and shortest paths . . . . .	71
36	Worst solution for 2-commodity ISPL . . . . .	72
37	The performance bound on this ISPL with 3 commodities is tight at $Z_3+Z_2-Z_1$	73
38	The performance bound on this ISPL with 3 commodities is tight at $2Z_3-Z_2-Z_1$	74
39	A feasible solution for 2-commodity ISPL worst case . . . . .	75
40	A feasible solution for 3-commodity ISPL worst case 1 . . . . .	76
41	A feasible solution for 3-commodity ISPL worst case 2 . . . . .	76
42	The worst performance of the algorithms is $d= K -1$ . . . . .	78
43	ISPL example 1 . . . . .	89
44	ISPL example 2 . . . . .	89
45	ISPL example 3 . . . . .	90
46	Performance of algorithms starting at the minimum cardinality paths . . .	92
47	Performance of algorithms starting at uniform random paths . . . . .	92
48	Iterations for convergence when starting with minimum cardinality paths .	96
49	Iterations for convergence when starting with uniform random paths . . . .	96
50	Hard ISPL performance of algorithms . . . . .	101
51	Iterations for convergence in the worst case . . . . .	103
52	Tyco network . . . . .	104
53	Level 3 network . . . . .	105
54	Global Crossing network . . . . .	105

# SUMMARY

The Inverse Shortest Path Length Problem (ISPL) is to find the vector of cost coefficients to satisfy given shortest path length constraints in a network. Given a graph and target shortest path lengths for some origin and destination pairs, we want a cost vector such that the shortest path length for each given pair will be equal to its target.

The objective of this dissertation is to explore the complexity, algorithms and applications of the Inverse Shortest Path Length Problem (ISPL). Prior researchers showed that ISPL is *NP*-complete. We explore the complexity of ISPL for special cases. We also introduce some heuristic algorithms and analyze their worst case performance. We test the algorithms on a set of randomly-generated instances, and observe that the results are usually within 3% of optimality. Finally, we present an application of ISPL to telecommunications bandwidth pricing, and test our heuristics on real-world data.

# CHAPTER I

## INTRODUCTION

### *1.1 Motivation of Inverse Problems*

When solving an optimization problem, we usually assume that all parameters are known exactly and we try to find an optimal solution. In practice, this assumption is often not correct. Sometimes, we are unsure about the parameters of the system but we have the optimization solutions. For example, assume there is an outcome of the system that we want, or that we know is optimal. We need to find a cost vector for the given system so that the desired outcome will be the system's optimal solution. Sometimes, the parameters (cost, capacity, travel time, etc.) are difficult to calculate exactly. The best thing we can do is to estimate them as closely as possible. The goal of inverse optimization is to find values for the parameters in order to make the desired solutions optimal. Often, we have an initial set of parameter values and our goal is to perturb them as little as possible when making our desired solutions optimal.

Inverse optimization problems can be divided into two categories: inverse solution optimization and inverse objective value optimization. The inverse solution optimization problem is to perturb the cost vector to make the given solutions optimal. There are numerous inverse combinatorial optimization problems that have been studied in this category, e.g., inverse shortest path problem (ISPP), inverse minimum spanning tree, inverse min cost flow, etc. The second category, inverse objective value optimization, has drawn less attention than the first one. In an inverse objective value optimization problem, we do not have the desired solution in advance, but we do have desired values for the objective functions. The problem is to find a cost vector which makes each piece of the optimal objective value equal to its desired value. The inverse shortest path length problem (ISPL) addressed in this thesis belongs to this category. In general, the problems in the first category are comparatively easier than the problems in the second. Ahuja and Orlin [3] proved that many

inverse solution optimization problems share the following property: if the original problem is polynomially solvable, then the corresponding inverse problem is also polynomially solvable. In Chapter 3, we show that ISPL is polynomially solvable if there is only one desired part of the objective function or two desired parts of the objective function. When the number of the desired parts of the objective function is greater than two, then ISPL becomes hard.

## 1.2 *Motivation of Inverse Shortest Path Problem*

One example of an inverse problem of the first type is the traffic assignment problem. Given an O-D (origin-destination) trip matrix, the traffic assignment problem is to assign traffic onto a network so that each driver uses his shortest path. In a user equilibrium, if two paths are used by the same O-D drivers, then the travel times of these two paths are equal. Moreover, nobody can decrease his travel time by changing his path independently. However, calculating the real travel times is difficult since the delay due to congestion is hard to estimate and is not linear in the traffic on the road. A reasonable and easy way to obtain estimates for travel time in the uncongested state is the arc length. Moreover, we can identify the shortest paths of some specific O-D pairs by investigating real driver routes. Hence, the actual travel time on each arc can be recovered by perturbing the estimated time as little as possible and making sure that these specific paths are shortest for the given O-D pairs. This is an Inverse Shortest Path Problem (ISPP). Another important application of ISPP from the geophysical sciences concerns predicting the movement of an earthquake.

The ISPP problem can be formulated as follows. Let  $\bar{c}$  denote the vector of estimated costs on the arcs, and let  $P_j$  be the desired shortest path for origin/destination pair  $(o_j, d_j)$ . The objective is to perturb the cost vector minimally while making sure that the identified paths  $P_j$  are the shortest ones with respect to the perturbed cost vector. In other words, if  $Q_j$  is the set of all paths from  $o_j$  and  $d_j$ , then every path  $q \in Q_j$  must be at least as long as  $P_j$ .

$$\min_c \|c - \bar{c}\| \quad (1.1)$$

$$s.t. \sum_{k|a_k \in q} c_k \geq \sum_{k|a_k \in P_j} c_k, \quad (j = 1, \dots, K, q \in Q_j) \quad (1.2)$$

$$c_k \geq 0 \quad (1.3)$$

The number of constraints is equal to the total number of paths for each  $(o_j, d_j)$ . There could be a huge number of constraints. However, we do not have to deal with all the constraints at the same time. Since we can find the shortest path in polynomial time, we can identify a violated constraint in polynomial time also. This allows us to handle many fewer constraints at each iteration and solve the problem in polynomial time.

### ***1.3 Motivation and Problem Statement of Inverse Shortest Path Length Problem***

In some situations, we do not have the desired shortest paths in advance, but we have the desired shortest path lengths. The objective is to recover the network costs to satisfy specific O-D shortest path length constraints. This is called the Inverse Shortest Path Length Problem (ISPL).

#### **1.3.1 ISPL in Telecommunication Pricing**

ISPL has an important application in telecommunication pricing. There has been unprecedented growth in the demand of bandwidth capacity over the past few years. Chiu and Crametz [11] point out that the deregulation of the telecommunication industry and the explosive growth of the internet have changed the supply-demand landscape for telecom capacity. Both supply and demand of bandwidth have increased dramatically.

RateXlab [25], which monitors the bandwidth market, has observed pricing inconsistency in the market. Pricing inconsistency means the pricing violates the triangular inequality. RateXlab examines pricing inconsistencies resulting in two types of apparent arbitrage opportunities: geographical and time. For example, RateXlab reports the price between New York and certain European cities can be higher than buying two segments connecting New



York and the destination city via London. This is an example of geographical arbitrage. The second type of arbitrage opportunity results from the total price of a long term contract costing less than that of a contract with shorter length. RateXlab found that one can purchase a 60-month contract and pay less than the total cost of 36-month contract. RateXlab point out these two kinds of arbitrage opportunities are not instant risk-free profit because most of the prices used in the analysis are not the actual trading price, but posted offers. Nevertheless, the pricing inconsistency draws our attention and suggests the need more careful pricing scheme. In this thesis, we focus on eliminating the more complex arbitrage opportunity, geographic arbitrage.

Chiu and Crametz [12] show that when two no-arbitrage relationships in bandwidth pricing are combined, opportunities for arbitrage may still exist. Therefore, instead of decomposing into small networks, we have to deal with the whole telecommunication network when trying to find consistent pricing. Since the product (bandwidth) is not storable and the market is extremely competitive, pricing of the bandwidth becomes an important issue for a company to survive and generate revenue[24].

We define a pricing strategy as the desired cost of bandwidth on a set of O-D pairs. A company may decide its pricing strategy for some specific O-D pairs after considering its competitor's price. To achieve the pricing strategy, the company must select a price for each segment so that the shortest path cost of each specific O-D pair will equal to the desired price.

### 1.3.2 Statement of ISPL

Formally, given an undirected network  $G := (N, E)$ , and desired shortest path length set (pricing strategy)  $Z = \{((o_r, d_r), z_r) : r = 1, 2, \dots, K\}$ , ISPL requires us to find costs  $c_{ij}$  for each  $(i, j)$  in  $E$  to satisfy the following condition: If  $P_r$  is the shortest path between  $o_r$  and  $d_r$  under cost coefficients  $c_{ij}$ , then  $\sum_{(i,j) \in P_r} c_{ij} = z_r$  for  $r = 1, \dots, k$ .

Before we introduce the formulation of ISPL, we first define  $Q_r(c)$  as the shortest path value function of commodity  $r$  with respect to cost  $c$ . Let  $G := (N, E)$  be a directed network,  $c = (c_1, c_2, \dots, c_{|E|})$  be the cost (length) of each arc, and  $\delta^+(n)$  denote the set of

incoming arcs and  $\delta^-(n)$  denote outgoing arcs at vertex  $n$ . Then  $Q_r(c) : R_{+,0}^{|E|} \mapsto R_{+,0}$  is a function defined as follows:

$$Q_r(c) = \min_y \sum_{a \in E} c_a y_a \quad (1.4)$$

$$s.t. \sum_{a \in \delta^+(n)} y_a - \sum_{a \in \delta^-(n)} y_a = -1 \quad \text{if } n = o_r \quad (1.5)$$

$$\sum_{a \in \delta^+(n)} y_a - \sum_{a \in \delta^-(n)} y_a = 0 \quad \text{if } n \in V \setminus \{o_r, d_r\} \quad (1.6)$$

$$\sum_{a \in \delta^+(n)} y_a - \sum_{a \in \delta^-(n)} y_a = 1 \quad \text{if } n = d_r \quad (1.7)$$

$$y_a \geq 0 \quad (1.8)$$

Constraints (1.5,1.6,1.7) define a path for commodity  $r$ , and the formulation above defines an arc-based shortest path formulation for commodity  $r$ .

The formulation of ISPL is the following. Let  $z_r$  be the desired shortest path length, and  $Q_r(c)$  be the shortest path value function of commodity  $r$  with respect to cost  $c$ . Then, (1.9)-(1.11) is an implicit formulation for ISPL. The major difference between ISPL and ISPP is that here we do not have each desired shortest path as in (1.2). The ISPL is then

$$\min_c f(c) \quad (1.9)$$

$$s.t. \quad Q_r(c) = z_r, \quad (r = 1, \dots, K), \quad (1.10)$$

$$c \geq 0, \quad (1.11)$$

where  $f$  is an arbitrary objective function.

Actually, ISPL is really a feasibility problem. No matter what kind of objective function we have, what we want is to find a feasible solution to satisfy (1.10) and (1.11). One possible way to pick between multiple feasible solutions is to minimize  $f(c) = \sum c_{ij}$  over all  $(i, j)$ . We now introduce another two possible objective functions. Let  $I_r(z_r, c)$  be an indicator of a violated constraint of  $(o_r, d_r)$  under some cost vector  $c$  by setting

$$I_r(z_r, c) = \begin{cases} 1, & \text{if } \sum_{(i,j) \in P_r} c_{ij} \neq z_r, \\ 0, & \text{otherwise.} \end{cases}$$

Because there is no specific objective function in ISPL, we can either minimize the sum of violations over all the given O-D pairs or minimize the number of violated paths. The different objectives lead to different schemes to approach the solution. The two possible objectives are

$$\text{Min} \left( \sum_{r=1}^K \left\| \sum_{(i,j) \in P_r} c_{ij} - z_r \right\| \right) \text{ or } \text{Min} \sum_{r=1}^K I_r(z_r, c).$$

## 1.4 Thesis Objective

The objective of this work is to explore the Inverse Shortest Path Length Problem (ISPL), including complexity, algorithms, and applications.

Some prior research has discussed the complexity of Inverse Shortest Path Length Problem (ISPL) and identified some special cases for which ISPL can be solved to optimality in polynomial time. The general ISPL is NP-hard and cannot be solved optimally in polynomial time. We will discuss the complexity of more complex special cases in an effort to identify when the problem becomes intractable. We also develop a family of new heuristics for ISPL and evaluate their performance on random problems and on data sets based on real world telecommunications data.

The remainder of this dissertation is organized as follows. Chapter 2 contains a literature review of inverse solution optimization problems, methods for solving inverse solution optimization problems, and inverse objective value optimization problems. It also summarizes previous research on the complexity of ISPL. Chapter 3 discusses our complexity results for some special cases of ISPL. In Chapter 4, we introduce different solution approaches for ISPL and propose some solution algorithms. Chapter 5 presents the telecommunications application in detail and reports the performance of the proposed algorithms of Chapter 4. Chapter 6 contains conclusions and proposes future research directions.

## CHAPTER II

### LITERATURE REVIEW

We give a literature review in this chapter. The content is organized as follows. In Section 2.1, we discuss the inverse solution optimization problem. Methods used for solving inverse solution optimization problems are introduced in Section 2.2. Section 2.3 covers inverse objective value optimization problems. We will focus on the inverse shortest path length problem and the corresponding complexity results in this section.

#### *2.1 Inverse Solution Optimization Problem*

Inverse optimization problems were first investigated by Burton and Toint [6]. They studied the inverse shortest path problem in 1992. After that, many inverse optimization problems were considered by different research groups. Cai and Li [8] considered inverse minimum spanning tree, inverse weighted matching, and inverse weighted matroid intersection. They showed that these problems are equivalent to the min-cost flow problem and hence can be solved in polynomial time. Liu and Zhang [23] considered other inverse combinatorial optimization problems. They showed that the inverse maximum matching perfect matching problem can be solved in polynomial time under an  $l_1$  norm.

Heuberger [20] showed that most polynomially solvable problems will have polynomially solvable inverse solution optimization problems. This is true for a large class of problems where the objective function of the original problem is linear, e.g.  $f(c, x) = cx$ . However, this is not true for all inverse problems. Cai, Yang, and Zhang [10] showed that the minmax inverse center location problem is *NP*-hard even though the original problem is polynomially solvable.

The inverse shortest path problem (ISPP) is polynomially solvable. Given a graph  $G(N, E)$ , costs  $c_{ij}$  for each  $(i, j) \in E$ , and a set of paths  $P_r$  (one for each O-D pair), we can perturb the costs on the edges such that the given paths  $P_r$  are the shortest paths for

their corresponding origins and destinations. Burton and Toint ([6],[7],[4]) formulate this as a quadratic programming problem where the objective is to minimize the perturbations with respect to an  $l_2$  norm. They solve it by using Goldfarb and Idnani's algorithm [18]. Ahuja and Orlin [3] use linear programming duality to solve the problem. Zhang, Ma, and Yang [41] use a column generation method to solve the unconstrained case under a unit weight  $l_1$  norm. Cai and Yang [9] propose a combinatorial algorithm to solve the constrained multiple objective inverse optimization problem under an  $l_1$  norm with asymmetric weights. Hochbaum [21] proposes a more efficient algorithm for the inverse spanning tree problem by formulating the problem as the dual of an assignment problem.

Day [13] and Day, Nemhauser, and Sokol [14] apply the inverse shortest path problem to the management of railroad impedances for shortest path-based routing.

Heuberger [20] gives a detailed survey of inverse optimization problems (both inverse solution optimization and inverse objective value optimization problems), which includes descriptions, methods and state of the art results for different problems.

## 2.2 *Methods for Solving Inverse Solution Optimization Problems*

In this section, we introduce some general methods used for solving inverse solution optimization problems. We take the classification of the methods from Heuberger [20]. The methods can be divided into the following categories:

### 1. Inverse linear programming method

Zhang and Liu [39] first introduced linear programming methods to solve inverse problems. Ahuja and Orlin [3] considered the dual of the inverse problem. They showed that the solution of the inverse problem is very similar to the original one and can be solved by using the associated dual solution. For example, the dual of the inverse shortest path problem is itself a shortest path problem. Their approach is the following:

Let  $A \in R^{m \times n}$ ,  $c \in R^n$ ,  $b \in R^m$ ,  $I := \{1, \dots, m\}$ ,  $J := \{1, \dots, n\}$ . Consider the linear

program

$$\begin{aligned}
& \min cx \\
& \text{s.t. } Ax \geq b, \\
& l \leq x \leq u.
\end{aligned} \tag{2.1}$$

The dual problem will be

$$\begin{aligned}
& \max \pi b + \rho l - \sigma u \\
& \text{s.t. } \pi A + \rho - \sigma = c, \\
& \rho \geq 0, \sigma \geq 0, \pi \geq 0.
\end{aligned} \tag{2.2}$$

The inverse optimization problem will be equivalent to solving the following system (2.3).

$$\begin{aligned}
& \min \|c - \widehat{c}\| \\
& \text{s.t. } \widehat{c}\widehat{x} = \min\{\widehat{c}x : Ax \geq b, l \leq x \leq u\}
\end{aligned} \tag{2.3}$$

Define corresponding constraints to ( 2.1 ) by setting  $B := \left\{ i \in I : \sum_{j \in J} a_{ij} \widehat{x}_j = b_i \right\}$ ,  
 $R := \left\{ i \in I : \sum_{j \in J} a_{ij} \widehat{x}_j < b_i \right\}$ ,  $S := \left\{ i \in I : \sum_{j \in J} a_{ij} \widehat{x}_j > b_i \right\}$ ,  $L := \{j \in J : \widehat{x}_j = l_j\}$ ,  
 $U := \{j \in J : \widehat{x}_j = u_j\}$  and  $T := J \setminus (L \cup U) = \{j \in J : l_j < \widehat{x}_j < u_j\}$ .

By the complementary slackness conditions,  $\widehat{x}$  is optimal for ( 2.1 ) if and only if the corresponding dual variables  $\pi, \rho$ , and  $\sigma$  satisfy

$$\begin{aligned}
& \sum_{i \in B} a_{ij} \pi_i + \rho_j = \widehat{c}_j, j \in R, \\
& \sum_{i \in B} a_{ij} \pi_i - \sigma_j = \widehat{c}_j, i \in S, \\
& \sum_{i \in B} a_{ij} \pi_i = \widehat{c}_j, i \in T,
\end{aligned} \tag{2.4}$$

$$\pi \geq 0, \rho \geq 0, \sigma \geq 0.$$

Therefore, the inverse linear programming problem can be reformulated as

$$\min\{\|c - \hat{c}\| : \exists \pi, \rho, \sigma \text{ such that the system (2.4) holds}\}. \quad (2.5)$$

Defining  $\alpha$  and  $\beta$  to be the vectors of decreases and increases in  $c$ , and  $w$  to be a vector of objective value weights on each change, we can write (2.5) as

$$\begin{aligned} & \max -w\alpha - w\beta \\ \text{s.t. } & \sum_{i \in B} a_{ij}\pi_i + \rho_j - \alpha_j = c_j, j \in R, \\ & \sum_{i \in B} a_{ij}\pi_i - \sigma_j + \beta_j = c_j, i \in S, \\ & \sum_{i \in B} a_{ij}\pi_i - \alpha_j + \beta_j = c_j, i \in T, \\ & \pi \geq 0, \rho \geq 0, \sigma \geq 0, \alpha \geq 0, \beta \geq 0. \end{aligned} \quad (2.6)$$

Then we consider the dual of (2.6) with associated dual variables  $\tilde{y}_j$  and take a linear transformation by setting  $\tilde{y}_j = y_j - \hat{x}_j$ , to get

$$\begin{aligned} & \min cy \\ \text{s.t. } & A_B y \geq b_B \\ & l_j \leq y_j \leq l_j + w_j, & \text{for } j \text{ such that } l_j = \hat{x}_j \\ & u_j - w_j \leq y_j \leq u_j, & \text{for } j \text{ such that } \hat{x}_j = u_j \\ & \hat{x}_j - w_j \leq y_j \leq \hat{x}_j + w_j, & \text{for } j \text{ such that } l_j \leq \hat{x}_j \leq u_j \\ & y \in R^n. \end{aligned} \quad (2.7)$$

In addition to a straightforward simplex solution of (2.7), other methods have been used. The idea of using column generation methods to solve inverse shortest path problems was proposed by Zhang, Ma, and Yang [41]. Yang and Zhang [35] use this method to solve an inverse combinatorial optimization problem (according to the input parameters, the problem could either be an inverse shortest path problem or an inverse minimum spanning tree problem).

Because we can formulate these inverse optimization problems as linear programs, the ellipsoid method can be used to prove polynomial solvability.

## 2. Duality for inverse problems

Sokkalingam [27] produces a duality theory of inverse linear programming under general norms using duality results from convex analysis. Sokkalingam, Ahuja, and Orlin [28] consider the duality of the inverse problem under an  $l_1$  norm and show that the inverse spanning tree problem can be formulated as the dual of an assignment problem on a bipartite network and can be solved in  $O(n^3)$  time. They also show that the weighted inverse spanning tree problem can be formulated as the dual of a transportation problem and can be solved in  $O(n^2 m \log(nC))$  time, where  $C$  is the largest arc cost in the network.

## 3. Newton type methods

Zhang and Liu [40] propose a general optimization model which includes most inverse optimization problems as its special cases (e.g., inverse minimum spanning tree problem, inverse minimum cost flow problem, inverse maximum perfect matching problem, etc). They also propose a Newton type method to solve these inverse combinatorial optimization problems in a uniform way under an  $l_\infty$  norm.

## 4. Direct combinatorial methods

Ahuja and Orlin [2] and Sokkalingam, Ahuja, and Orlin [28] both propose combinatorial algorithms to solve the inverse spanning tree problem. Hochbaum [21] proposes a more efficient algorithm for the inverse spanning tree problem by formulating the problem as the dual of an assignment problem.

## 5. Others

Tong and Lam [29] use an embedded connection list approach to solve inverse shortest path problems. Zhang and Ishikawa [37] uses a neural network approach to solve inverse optimization problems. Amico, Maffloi and Malucelli [15] propose a base-matroid idea to solve inverse combinatorial optimization problems.



## 2.3 Inverse Objective Value Optimization Problems and Their Complexity

Burton and Toint [5] were the first to discuss a problem similar to inverse objective value optimization. They examine the complexity of the inverse shortest path problem with upper bounds on shortest path costs and prove that obtaining a globally optimal solution to this problem is *NP*-complete. They show that ISPL is also *NP*-complete by reducing the 3-satisfiability problem to it. They show that the feasible set is not convex and propose a local search algorithm. The algorithm is based on quadratic programming and uses the algorithm of Burton and Toint [6] as a subroutine.

Heuberger [20] defined “reverse optimization problems” to be those problems with given desired objective values, but not desired solutions (e.g., ISPL). We will adopt Burton’s terminology and use “inverse optimization” to “reverse optimization”.

Ahmed and Guan [1] show that the inverse optimal value problem of linear programming is *NP*-complete. They provide sufficient conditions under which the problem can be solved in polynomial time. Ahmed and Guan also give an algorithm to solve the general case problem based on solving linear and bi-linear programming problems. Yang and Zhang [36] study the inverse optimal value problem for minimum spanning tree, show the problem can be formulated as a combinatorial linear program, and present a combinatorial strongly polynomial algorithm for it.

Zhang and Lin [38] consider an inverse shortest path problem of trying to shorten the edge lengths as little as possible such that the distance between specific origins and destination is inside some desired bounds. Burton and Toint [5] already showed the general case of this problem is *NP*-complete. Zhang and Lin show that the problem is polynomially solvable when restricted to the cases where the demand graph is a forest or a single terminal. Polynomial-time algorithms are also presented.

Faragó, Szentesi and Szviatovszki [16] study the inverse shortest path problem and apply it to an ATM (asynchronous transfer mode) telecommunications network. Their problem is a form of the inverse optimal value problem. There is no initial cost or length on edges; hence, the objective is not to minimize the perturbation of the cost vector. However, they

still have desired shortest paths between some specific nodes. They define three types of tasks applied in the ATM network and use linear programming to solve these three tasks in polynomial time. They also indicate the application to the high-speed telecommunication network.

Fekete, Hochstättler, Kromberg, and Moll [17] investigate the complexity of ISPL based on characteristics of the distance graph. Let  $G = (N, E)$  be an undirected graph. The distance graph  $G_d = (N, E_d, z_d)$  is defined by a weighted function  $z_d : E_d \mapsto R_{|E_d|}^+$  describing the shortest distances between specific node pairs  $E_d$ . A cost function  $c : E \mapsto R_{|E|}^+$  for  $G$  is called a  $G_d$  – *satisfying* function if the shortest path length for each edge of  $E_d$  in  $G(N, E, c)$  is equal to the corresponding value of  $z_d$ . ISPL is identical to the question: “given  $G$  and  $G_d$ , is there a  $G_d$  – *satisfying* function  $c$  on  $G$ ?”

Fekete, Hochstättler, Kromberg, and Moll [17] show ISPL is *NP*-complete by reducing the vertex-disjoint paths problem to ISPL. They indicate that ISPL is polynomially solvable under very restricted cases. They also show that ISPL becomes intractable even when the distance graph is only a little more complex than a polynomially solvable instance. Their major complexity results are the following:

1. ISPL is *NP*-complete.
2. ISPL is *NP*-complete, even if restricted to instances  $G, G_d$  for which  $G + G_d, (N, E \cup E_d)$ , is planar.
3. ISPL is polynomially solvable if the distance graph is the union of complete stars. (A set of edges  $S$  is called a star if all edges are incident to one vertex  $s$ , i.e.,  $S \subseteq s \times N$ . If  $S$  covers all vertices of the graph,  $S$  is a complete star).
4. ISPL is *NP*-complete, even when restricted to instances where the edges of the distance graph can be covered by two stars.
5. ISPL for digraphs is *NP*-complete, even restricted to instances where the distance graph has only 3 edges.

For the telecommunications application, complexity with respect to the structure of the underlying physical network  $G = (N, E)$  is more relevant, so those are the complexity results we focus on.

## CHAPTER III

### COMPLEXITY OF ISPL

Prior researchers have already shown that ISPL is *NP*-complete. We show that some special cases of ISPL are polynomially solvable, and discuss the complexity of ISPL under different scenarios to identify the ranges dividing ISPL instances into polynomially solvable and *NP*-complete.

This chapter is organized as follows: In Section 3.1, we give a formal statement of ISPL. A summary of previous research on ISPL's complexity is included in Section 3.2. Section 3.3 contains our complexity results for ISPL. We propose some research directions for ISPL complexity in Section 3.4.

#### 3.1 Formal Statement

The Inverse Shortest Path Length problem (ISPL) can be solved on undirected or directed networks. We define ISPL on an undirected network. The only difference between the undirected version and the directed version is in the path formulation. To avoid ambiguity, the network that we consider in the following content is undirected unless otherwise specified.

In ISPL, we are given an undirected network  $G := (N, E)$  and desired shortest path length set  $Z = \{((o_r, d_r), z_r) : r = 1, 2, \dots, K\}$ . Each origin and destination  $(o_r, d_r)$  can be treated as a commodity traveling on a shortest path from  $o_r$  to  $d_r$  where we want the shortest path distance to be equal to  $z_r$ . Define  $Q_r(c)$  to be the shortest path value function of commodity  $r$  with respect to cost  $c$ . Let  $c = (c_1, c_2, \dots, c_{|E|})^t$  be the cost (length) of each edge. Let  $Q_r(c) : R_+^{|E|} \mapsto R_+$  by setting it as the following:

$$Q_r(c) = \min_y \sum_{a \in E} c_a y_a \quad (3.1)$$

$$\text{s.t. } \sum_{a \in \delta^+(n)} y_a - \sum_{a \in \delta^-(n)} y_a = -1 \quad \text{if } n = o_r \quad (3.2)$$

$$\sum_{a \in \delta^+(n)} y_a - \sum_{a \in \delta^-(n)} y_a = 0 \quad \text{if } n \in V \setminus \{o_r, d_r\} \quad (3.3)$$

$$\sum_{a \in \delta^+(n)} y_a - \sum_{a \in \delta^-(n)} y_a = 1 \quad \text{if } n = d_r \quad (3.4)$$

$$y_a \geq 0 \quad (3.5)$$

where  $\delta^+(n)$  denotes the set of incoming edges to node  $n$  and  $\delta^-(n)$  denotes the set of outgoing edges from  $n$ . The formulation above identifies a shortest path for commodity  $r$ .

Let  $z_r$  be the desired shortest path length, and  $Q_r(c)$  be the shortest path value function, of commodity  $r$  with respect to cost  $c$ . We want to find nonnegative costs  $c_{ij}$  for each  $(i, j) \in E$  to satisfy  $Q_r(c) = z_r$  for  $r = 1, \dots, K$  under some objective function. The formulation of ISPL is following.

$$\min_c f(c) \quad (3.6)$$

$$Q_r(c) = z_r, \quad (r = 1, \dots, K) \quad (3.7)$$

$$c \geq 0 \quad (3.8)$$

Notice that ISPL is actually a feasibility problem. The objective function  $f$  could be arbitrary. Given an ISPL instance, we are not trying to find the optimal solution. We just want to find a feasible solution to satisfy (3.7) and (3.8).

**Definition 3.1.** *Given an ISPL instance, we say that the instance is feasible if and only if there exists a cost vector  $\bar{c} \in R_{+,0}^{|E|}$  such that the following conditions are satisfied:*

$$Q_r(\bar{c}) = z_r \quad \text{for } r = 1, \dots, K. \quad (3.9)$$

Checking whether the given ISPL instance is feasible or not is a *NP*-complete problem in general. Moreover, the implicit equation (3.7) makes ISPL difficult to solve directly. Hence, we try to solve this feasibility problem, ISPL, by solving a different optimization problem.

For example, we can solve the optimization problem formulated as (3.10)-(3.12). The objective function is to minimize the sum of shortest path length violation of all commodities when restricted such that the shortest path length of each commodity must be greater than or equal to the desired value.

$$\min f(c) = \sum_{r=1}^K (Q_r(c) - z_r) \quad (3.10)$$

$$Q_r(c) \geq z_r, \quad (r = 1, \dots, K) \quad (3.11)$$

$$c \geq 0 \quad (3.12)$$

Alternatively, we can remove (3.11) and choose an objective function like the following.

$$f(c) = \sum_{r=1}^K \|Q_r(c) - z_r\| \quad (3.13)$$

Then the problem becomes to minimize the *norm* between  $Q_r(c)$  and  $z_r$ . Any norm can be used. In Chapter 4, we discuss a family of heuristics for ISPL based on these ideas.

### 3.2 Summary of Previous ISPL Complexity Results

To our knowledge, there are only two previous studies that discuss the complexity of the inverse shortest path length problem. Burton and Toint [5] first investigated the complexity of ISPL and proved ISPL is *NP*-complete by reducing the 3-satisfiability problem to it. Fekete, Hochstättler, Kromberg, and Moll [17] used the concept of the distance graph (demand graph) to discuss the complexity of ISPL. They showed that ISPL is *NP*-complete by reducing the vertex-disjoint-path problem to it. Fekete, Hochstättler, Kromberg, and Moll also indicated that ISPL is polynomially solvable in very restricted cases, and that ISPL instances of these polynomially solvable classes become intractable when the underlying  $O - D$  structure varies slightly.

Unlike Fekete, Hochstättler, Kromberg, and Moll who discuss complexity in terms of the demand graph, we identify additional classes of ISPL that are polynomially solvable by considering the physical network. If the underlying physical network is a tree, a forest, or a complete graph, then ISPL is polynomially solvable (see Section 3.3). Since both the most simple and the most complicated structures are polynomially solvable, and ISPL in general is not, we also identify ranges dividing the ISPL instances into polynomially solvable and intractable.

### 3.3 Complexity Results

In this section, we discuss the complexity of ISPL on special graphs (for example, different underlying network structures and numbers of commodities). In the first subsection, we show complexity results for ISPL when the underlying network is a tree, complete graph, and some modifications of the complete graph. The complexity of 2-commodity ISPL is also discussed in Section 3.3.1. The ISPL complexity result when the underlying network is a cycle is discussed in Section 3.3.2.

#### 3.3.1 Tree and Complete Graph ISPL

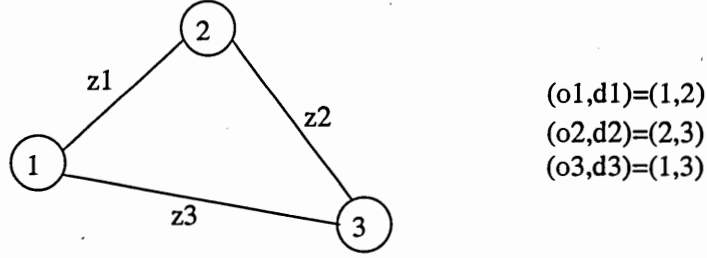
First, we consider two special cases of underlying networks: trees and complete graphs. We will not discuss the case of forests separately since the proof is the same as for multiple trees.

**Theorem 3.1.** *ISPL is polynomially solvable when the underlying network is a tree.*

*Proof.* Suppose that the underlying network is a tree. Then there exists a unique path  $p_r$  for each commodity  $r$ . Hence, the shortest paths used to calculate  $Q_r(c)$  in (1.10) are defined uniquely for each commodity. Therefore, the problem becomes the following linear system of equalities:

$$\sum_{(i,j) \in p_r} c_{ij} = z_r, r = 1, \dots, K \quad (3.14)$$

Both solving and checking feasibility can be done in polynomial time.  $\square$



**Figure 1:** Distance graph must be a Euclidean graph

Before we prove that ISPL is polynomially solvable when the underlying network is a complete graph, we need the following lemma.

**Lemma 3.1.** *Given an instance of ISPL, an undirected network  $G := (N, E)$ , and desired shortest path length set  $Z = \{((o_r, d_r), z_r) : r = 1, 2, \dots, K\}$ , let  $G_d = (N, E_d, z_r)$  be the distance graph defined as follows:*

$E_d = \{(o_r, d_r) : r = 1, 2, \dots, K\}$  and  $z_r$  is the desired shortest path length from  $o_r$  to  $d_r$ .

*If the ISPL instance is feasible, then  $G_d$  must be a Euclidean graph (i.e.,  $G_d$  satisfies the triangular inequality).*

*Proof.* We prove this lemma by contradiction.

Suppose that  $G_d$  is not a Euclidean graph, i.e., it violates triangular inequality. Without loss of generality, we may assume there exists a triangle with  $z_1 + z_2 < z_3$  (see Figure 1). Since  $G_d$  is a graph that defines the shortest path lengths, the shortest path length between nodes 1 and 3 must be equal to  $z_3$ . Since  $1 - 2 - 3$  is a path between node 1 and node 3, we require  $z_1 + z_2 \geq z_3$ , which is a contradiction. Thus, the ISPL instance is infeasible. Therefore, if the ISPL instance is feasible, then  $G_d$  must be a Euclidean graph.  $\square$

**Theorem 3.2.** *ISPL is polynomially solvable when the underlying network is a complete graph.*

*Proof.* Suppose that the underlying undirected network is a complete graph. We can obtain a set of vertex disjoint paths for the  $O - D$  pairs by using the edges between the origin and destination of each commodity (since  $G$  is complete). Let  $c_{ij} = z_r$  if  $o_r = i$  and  $d_r = j$  for some commodity  $r$ , and  $c_{ij} = M$  otherwise ( $M$  is a big number larger than any  $z_r$ ). Thus



we have one path length (namely, the length of the direct edge) equal to the desired length for each commodity. We now show that either this solution is feasible, or else  $G_d$  is not Euclidean.

If  $c$  is feasible, Lemma 3.1 guarantees that  $G_d$  is Euclidean.

Suppose the solution  $c$  is not feasible. Then there must be some commodity  $r$  for which  $Q_r(c) < z_r = c_{o_r, d_r}$ . Since  $M > z_r$ , the shortest path for commodity  $r$  must not include any edges  $(i, j)$  that are not in the set of  $O - D$  pairs. Therefore, the shortest path from  $o_r$  to  $d_r$  consists of nodes  $o_r = i_1, i_2, \dots, i_k = d_r$  such that each  $(i_{m-1}, i_m)$  is an  $O - D$  pair, for  $m = 2, 3, \dots, k$ . By definition, the length of this path is equal to the length of these commodities' desired shortest paths. Since  $Q_r(c) < z_r$ , these  $O - D$  pairs form a shortcut of commodity  $r$ , and thus  $G_d$  is not Euclidean.

Checking feasibility can be done by solving the some-to-some shortest path problem. It is  $O(n^3)$  when using the Floyd-Warshall algorithm, and thus can be done in polynomial time.  $\square$

We now have shown that ISPL is polynomially solvable both when the underlying network is a tree and when it is a complete graph. Since two extreme cases are polynomially solvable and the general ISPL is *NP*-complete, the remaining question is when ISPL becomes hard. We first discuss modifications of the complete graph here.

Before we prove the theorem, we need the following lemmas:

**Lemma 3.2.** *Given an instance of ISPL, if*

- (1) *there is a vertex disjoint path set for commodities  $(o_r, d_r)$  ( $r = 1, \dots, K$ ), or*
- (2) *there is an internally vertex disjoint path set for commodities  $(o_r, d_r)$  ( $r = 1, \dots, K$ )*

*and the distance graph  $G_d$  is an Euclidean Graph (Lemma 3.1),*

*then the instance must be feasible.*

*Proof.* Case (1) is trivial. Let  $P = \{p_r : p_r \text{ is an } o_r - d_r \text{ path, } r = 1, \dots, K\}$  be the vertex disjoint path set and let  $|p_r|$  denote the number of edges of  $p_r$ . Assign  $c_{ij} = \frac{z_r}{|p_r|}$  for  $(i, j) \in p_r$ ,  $r = 1, \dots, K$  and  $c_{ij} = M$ , where  $M$  is a large number, for all other edges. Then we have a feasible solution of the ISPL instance. Hence, the instance is feasible.

When case (2) holds, assign edge costs  $c_{ij}$  as in case (1). We have a path for each commodity  $r$  with length equal to  $z_r$ . We must show that there is no shorter path for any commodity under the cost vector  $c_{ij}$  we just assigned. Every path for commodity  $r$  using edges other than those in  $P$  will have longer length than  $z_r$  because it contains an edge of length  $M > z_r$ . Thus, any possible path of shorter length must use only edges in  $P$ . However, by the fact that each  $p_r$  is internally vertex disjoint from the others and because the distance graph satisfies the triangular inequality (by Lemma 3.1), we can guarantee that there is no shorter path for each commodity. Hence, we have a feasible solution of the ISPL instance.

Thus, if either case (1) or case (2) is true, then the ISPL instance must be feasible.  $\square$

We now return to studying the complexity of ISPL in the case of dense graphs.

Suppose that we remove one edge from the complete graph. We now show that ISPL is still polynomially solvable. Consider an ISPL instance  $(G = (N, E), Z)$ . Let  $(u, v)$  be the edge removed from the complete graph. Consider the graph  $G' = G \setminus \{(u, v)\}$ . Then the node set in  $G'$  can be partitioned into  $A$ ,  $B$ , and  $C$  as follows (see Figure 2):

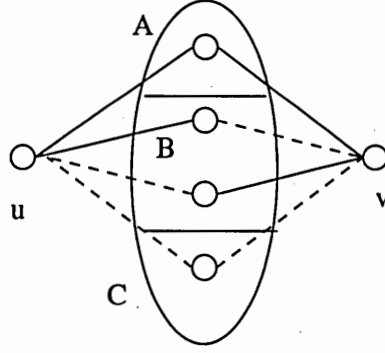
$$\begin{aligned} A &: \{w \in N \setminus \{u, v\} \text{ such that both } (u, w), (v, w) \in Z\} \\ B &: \{w \in N \setminus \{u, v\} \text{ such that either } (u, w) \text{ or } (v, w) \in Z \text{ but not both}\} \\ C &: \{w \in N \setminus \{u, v\} \text{ such that neither } (u, w) \text{ nor } (v, w) \in Z\} \end{aligned} \quad (3.15)$$

Note that  $A \cap B = B \cap C = A \cap C = \emptyset$  and  $A \cup B \cup C = E \setminus \{u, v\}$ .

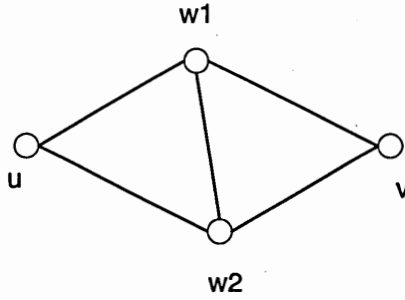
To show that ISPL is polynomially solvable when the underlying network is a complete graph with exactly one edge removed, we first prove the following lemma.

**Lemma 3.3.** *Let  $((u, v), z_{uv}) \in Z$  be a commodity in an ISPL instance on a graph  $K_n \setminus \{(u, v)\}$ . If the instance is feasible, then there exists a feasible solution in which the shortest path  $P_{uv}$  from  $u$  to  $v$  contains exactly 2 edges.*

*Proof.* Since  $G = K_n \setminus \{uv\}$ , we have  $|P_{uv}| \geq 2$ .



**Figure 2:** Partition of vertices in a complete graph without edge  $(u, v)$



**Figure 3:** If  $K_n \setminus \{(u, v)\}$  is feasible, then  $\exists P_{uv}$  such that  $|P_{uv}| = 2$

Suppose  $|P_{uv}| > 2$  for all shortest paths from  $u$  to  $v$  in some feasible solution. Without loss of generality, we may assume  $|P_{uv}| = 3$ . Let  $P_{uv} = u \rightarrow w_1 \rightarrow w_2 \rightarrow v$  be the shortest path, and  $|P_{uv}| = 3$  (see Figure 3). By Lemma 3.1, we have

$$c_{w_1 w_2} + c_{w_2 v} \geq c_{w_1 v} \text{ and } c_{u w_1} + c_{w_1 w_2} \geq c_{u w_2}.$$

Therefore, paths  $u \rightarrow w_1 \rightarrow v$  and  $u \rightarrow w_2 \rightarrow v$  are at least as short as  $u \rightarrow w_1 \rightarrow w_2 \rightarrow v$ . This is a contradiction.

□

**Theorem 3.3.**  $K_n \setminus \{(u, v)\}$  is polynomially solvable.

*Proof.* Let  $K_n$  be an undirected complete graph and  $(u, v)$  be the only missing edge from  $K_n$ . We may assume there is a commodity between  $(u, v)$ . Otherwise, the problem can be solved as a complete graph (see Theorem 3.2) since we allocate cost  $M$  on all edges without a commodity on them.

Consider the vertex sets  $A, B, C$  as defined in (3.15).

(1) If  $C \neq \phi$ , then we can pick any  $w \in C$  and set the  $u, v$  path to be  $u \rightarrow w \rightarrow v$  and the rest of commodities' paths to be their direct edges. All other edges get a large cost. Obviously, the paths set are internally vertex-disjoint since both  $(u, w), (v, w) \notin Z$ . By Lemma 3.2, this is a feasible solution (obviously obtained in polynomial time).

(2) If  $C = \phi$  and  $B \neq \phi$ , then there exists a node  $w \in B$  such that  $w$  is adjacent to exactly one vertex of  $u$  and  $v$ . When we assign the shortest paths, if any  $w \in B$  such that  $z_{uv} > z_{uw}$ , and  $(u, w) \in Z$  is feasible when  $c_{wv} = z_{uv} - z_{uw}$  and  $p_{uv} = \{(u, w), (w, v)\}$ , then it is a feasible solution. We can check the feasibility by solving the some-to-some shortest path problem. The maximum number of choices of  $w$  is  $n - 2$ , which is  $O(n)$ . Therefore, it is polynomially solvable.

(3) If  $C = \phi$ , and case (2) contains no feasible solution, then

(i) if  $\exists w \in A$  such that  $z_{uw} + z_{wv} = z_{uv}$ , then  $u \rightarrow w \rightarrow v$  is the desired path;

(ii) if not, then the problem instance is infeasible by Lemma 3.3.  $\square$

Since ISPL on a complete graph with one edge removed is polynomially solvable, we now consider modifying the complete graph further. Specially, we consider the case when there are  $k$  edges omitted from the complete graph  $K_n$ .

**Theorem 3.4.** *Suppose  $k$  edges are missing from the complete graph  $K_n$  ( $k \leq n - 2$ ) in an instance of ISPL. Let  $P_{uv}$  be the shortest path between  $u$  and  $v$  with the smallest number of edges and  $|P_{uv}|$  be the number of edges used by  $P_{uv}$ . If the instance is feasible, then  $|P_{uv}| \leq k + 1$ . Moreover, if  $|P_{uv}| = k + 1$ , then both endpoints of each of the missing edges are in  $P_{uv}$ .*

*Proof.* We prove this theorem by induction.

Let  $c_0$  denote the cost vector that solves the complete graph ISPL instance as desired in Theorem 3.2. By the theorem, it must be feasible.

If  $k = 1$ , Lemma 3.3 requires that the shortest path is direct for all commodities that correspond to edges in  $G$ , and  $|P_{uv}| = 2$  for the missing edge  $(u, v)$ . Because there is no  $(u, v)$  edge, both endpoints  $u$  and  $v$  of the missing edge must belong to  $P_{uv}$ . We can get a

feasible cost vector  $c_1$  from  $c_0$  by keeping all edge costs unchanged except for  $(u, v)$  and the two edges used by  $P_{uv}$ .

Suppose the claim is true for  $k = m$ . Let  $c_m$  be a feasible solution where  $P_{uv}$  is a minimal edge path for each commodity  $(u, v)$  and  $|P_{uv}| \leq m + 1$ .

Remove one more edge  $e = (r, s)$  from the graph.

If the missing edge  $e$  is not part of any shortest path  $P_{uv}$ , then  $P_{uv}$  is still a feasible shortest path for new graph; thus  $|P_{uv}| \leq m + 1$ . In this case,  $c_{m+1} = c_m$  (for all edges other than  $(r, s)$ ) must be a feasible solution.

If  $e = (r, s) \in P_{uv}$ , then  $P_{uv}$  is no longer feasible. Consider the induced subgraph  $G' = (N', E')$  such that

$$N' = N \setminus \{n \in N : n \in P_{uv}, n \neq r, s\}$$

$$E' = \{(i, j) \in E : i \in N', j \in N'\}$$

Then  $G'$  is a complete graph missing one edge  $(r, s)$ . Then the shortest path between  $r, s$  is a path with 2 edges without using any vertices in  $P_{uv}$  other than  $r, s$ . Thus the shortest length path  $P_{uv}$  with  $|P_{uv}| = (m + 1) - 1 + 2 = m + 2$  are available. To show the path is feasible, let  $z_1, z_2, z_3$  be the cost for  $P_{ur}, P_{rs}, P_{sv}$  respectively. Since the path is feasible when  $k = m$ , we have  $z_{uv} = z_1 + z_2 + z_3$ . Since  $(r, s) \in K$ , have  $z_2 = z_{rs}$ . Hence the new path is feasible. Then we can get  $c_{m+1}$  by changing  $c_m$  on  $(r, s)$  and two edges used by  $P_{rs}$ .

□

**Theorem 3.5.** *An instance of ISPL on  $K_n$  with some constant  $k$  edges removed is polynomially solvable.*

*Proof.* By Theorem 3.4, the number of edges in the shortest path for commodity  $r$  should be less than or equal to  $k + 1$ . An upper bound on the number of possible paths of commodity  $r$  when  $k$  edges are missing ( $0 \leq k \leq n - 2$ ) is  $\sum_{m=0}^k \binom{n-2}{m} \cdot m!$ .

$$\sum_{m=0}^k \binom{n-2}{m} \cdot m! = \sum_{m=0}^k \frac{(n-2)!}{(n-2-m)!} \leq \sum_{m=0}^k n^m \leq \sum_{m=0}^k n^k = (k+1)n^k = O(kn^k)$$

Hence, there are  $O(kn^k)$  possible paths for each commodity  $r$  such that  $(o_r, d_r) \notin E$ .

Since there are at most  $k$  commodities for which we have to enumerate possible paths, the maximum number of combinations is  $O(kn^k)^k = O(k^k n^{k^2})$ .

Since  $k$  is a constant, we can enumerate all possible combinations of paths in polynomial time. Therefore, an instance of ISPL on  $K_n$  with  $k$  edges removed is polynomially solvable.  $\square$

### 3.3.2 Cycle ISPL

In the previous section, we studied the complexity of ISPL when the physical network is dense. Now we study ISPL on sparse graphs. Theorem 3.1 shows that ISPL on a tree is polynomially solvable. We now discuss the complexity of ISPL when the underlying network is a simple cycle.

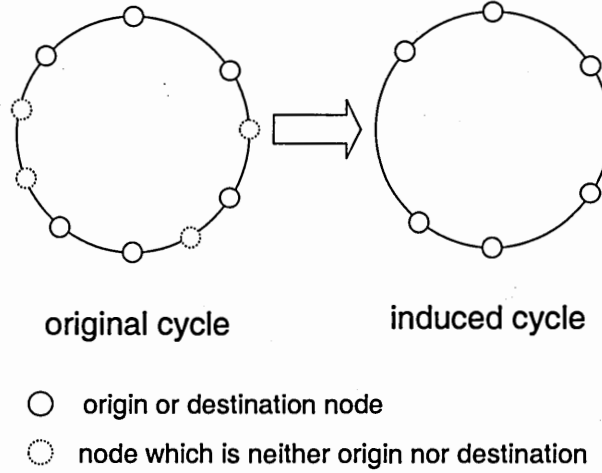
Given an ISPL instance on a cycle, there are exactly two possible paths for each commodity. Since we consider an undirected cycle, we can reverse the commodity origin and destination without affecting the feasibility. We can also ignore those nodes in the cycle which are neither origin nor destination for any commodity (see Figure 4). It is trivial that the cycle ISPL is feasible if and only if the ISPL on such an induced cycle is feasible. Each path using an ignored node in the original cycle must also use the two adjacent nodes since the ignored node is neither origin nor destination for any commodity. Therefore, only the sum of the two adjacent edges is important. Without loss of generality, the cycle ISPL that we discuss in the following sections is assumed to be an ISPL instance such that the underlying network is the induced cycle.

For each commodity of cycle ISPL, we can count how many edges are contained in each of its two paths. Let  $PC_i$  denote the path which contains smaller number of edges for commodity  $i$ .

Before we discuss the complexity of cycle ISPL, we introduce some definitions.

#### 3.3.2.1 Restricted Cycle ISPL

We define a cycle ISPL in which all the commodities have the same desired shortest path length  $z$  to be a *restricted cycle ISPL*. Let  $SP$  denote the chosen shortest path set of all commodities,  $p_{k1}$  denote the chosen shortest path of commodity  $k$ , and  $p_{k2}$  be the other



**Figure 4:** Original Cycle ISPL to Induced Cycle ISPL

path for commodity  $k$ . Consider the linear programming formulation of restricted cycle ISPL with the objective function set to minimize the total cost on the cycle.

$$\text{Min } \sum_{e \in C} c_e \quad (3.16)$$

$$\text{s.t. } \sum_{e \in p_{k1}} c_e = z \quad \text{for } k = 1, 2, \dots, K \quad (3.17)$$

$$\sum_{e \in p_{k2}} c_e \geq z \quad \text{for } k = 1, 2, \dots, K \quad (3.18)$$

$$c_e \geq 0 \quad (3.19)$$

The dual of the restricted cycle ISPL formulation will be the following:

$$\text{Max } \sum_{k,i=1,2} y_{p_{ki}} z \quad (3.20)$$

$$\sum_{e \in p_{k1} \text{ or } e \in p_{k2}} y_{p_{ki}} \leq 1 \quad \text{for } \forall e \in C \quad (3.21)$$

$$y_{p_{k2}} \geq 0 \quad \text{for } k = 1, 2, \dots, K \quad (3.22)$$

$$y_{p_{k1}} \text{ unrestricted for } k = 1, 2, \dots, K \quad (3.23)$$

The dual problem is feasible since there is a trivial solution ( $y_{p_{k1}} = y_{p_{k2}} = 0$  for all

$k$ ). Hence, the restricted cycle ISPL formulation is infeasible if and only if the dual is unbounded. Hence, if we can find an optimal  $y$  that guarantees the dual problem is not unbounded, then the restricted cycle ISPL must be feasible.

Define three sets  $S^+$ ,  $S^-$  and  $S^0$  as following:

$$S^+ = \{y_{p_{ki}} | y_{p_{ki}} \rightarrow \infty\}$$

$$S^- = \{y_{p_{ki}} | y_{p_{ki}} \rightarrow -\infty\}$$

$$S^0 = \{y_{p_{ki}} | y_{p_{ki}} \notin S^+ \cup S^-\}$$

Let  $w_{p_{ki}}$ ,  $v_{p_{ki}}$  and  $u_{p_{ki}}$  be 0 – 1 variables representing the following relations.

$$w_{p_{ki}} = 1 \Leftrightarrow y_{p_{ki}} \in S^+$$

$$v_{p_{ki}} = 1 \Leftrightarrow y_{p_{ki}} \in S^-$$

Consider the following induced linear programming problem.

$$\text{Max} \sum_{k,i} w_{p_{ki}} - \sum_{k,i} v_{p_{ki}} \quad (3.24)$$

$$\sum_{k,p_{ki}, e \in p_{ki}} v_{p_{ki}} \geq \sum_{k,p_{ki}, e \in p_{ki}} w_{p_{ki}} \text{ for } \forall e \in C \quad (3.25)$$

$$v_{p_{ki}} + w_{p_{ki}} \leq 1 \text{ for each } k, i \quad (3.26)$$

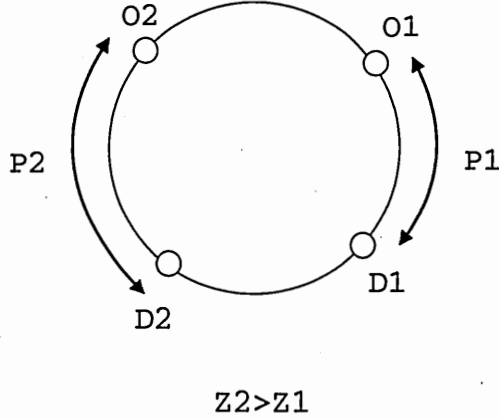
$$v_{p_{ki}}, w_{p_{ki}} \in \{0, 1\} \quad (3.27)$$

$$v_{p_{k2}} = 0 \text{ for all } k \quad (3.28)$$

Then answering the question of whether the restricted cycle ISPL is feasible is equivalent to answering the following question: Is there a path set  $SP$ , one path  $p_k$  for each commodity  $k$ , such that the induced linear programming problem ( 3.24-3.28) is feasible and the optimal objective value is equal to 0? If we can answer this question in polynomial time, then the restricted cycle ISPL can be solved in polynomial time also.

Since the complexity of ISPL has strong relationship with finding disjoint paths, we discuss how disjoint paths impact the cycle ISPL. Suppose there are two commodities that have disjoint paths for general cycle ISPL as in Figure 5, with  $z_2 > z_1$ . Then the only





**Figure 5:** Two commodities have disjoint paths on the general cycle ISPL instance

possible path that commodity 1 can choose is  $P_1$  because the other path must contain  $P_2$  and the length of  $P_2$  is at least  $z_2$  which is strictly greater than  $z_1$ . Hence, detecting disjoint paths on a cycle can reduce the size of the possible shortest path set  $SP$ .

Let  $G_L = (V_L, E_L)$  be a logical graph such that each vertex  $v$  represents a path of a commodity. Two vertices are adjacent if choosing these two paths to be the shortest for their commodities will not directly result in infeasibility. The logical graph for a cycle is shown in Figure 6. It is trivial that there are two paths for each commodity. In the example Figure 6, we can find that there is only one possible path for commodities 1 and 3. Suppose we just consider the first 3 commodities. The number of possible path sets we have to enumerate reduces from 8 to 2. More specifically, the only possible path set for ISPL is a simple path on the logic graph passing through each commodity exactly once. Hence, the number of path sets we have to enumerate is the same as the number of paths which pass through each commodity with length equal to  $K - 1$ . If there is no such path, then the instance of ISPL must be infeasible.

The benefits to solving the restricted cycle ISPL resulting from disjoint paths are different from the general cycle case. Since all the  $z$  are equal, we cannot erase any possibility by considering two disjoint paths as we did in the general cycle ISPL. Instead of two disjoint paths, we consider the relation between three disjoint paths.

**Lemma 3.4.** *If there are 3 commodities which have disjoint paths in a restricted cycle*

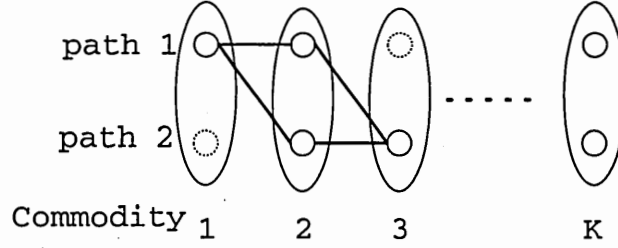


Figure 6: Logic graph of general cycle ISPL

ISPL, then every feasible solution must make these disjoint paths shortest for the three commodities.

*Proof.* Suppose there are 3 commodities which have disjoint paths  $P_1$ ,  $P_2$ , and  $P_3$  as in Figure 7. Let  $\overline{P}_i$  be the other path of commodity  $i$  (so  $\overline{P}_i = C \setminus P_i$ ) and let  $l(P_i)$  denote the length of  $P_i$ . Consider all the possible choices of shortest path sets.

- (1)  $(P_1, P_2, P_3)$  : It is trivial that this is feasible.
- (2)  $(\overline{P}_1, P_2, P_3)$  : Since  $P_2$  and  $P_3$  are disjoint and both are contained in  $\overline{P}_1$ ,  $l(\overline{P}_1) \geq l(P_2) + l(P_3) = 2z$ . Therefore, it is infeasible.
- (3)  $(\overline{P}_1, \overline{P}_2, P_3)$  : Since  $l(P_3) = z$  and  $\overline{P}_1$  contains  $P_2$  and  $P_3$ ,  $l(\overline{P}_1) = z$  implies  $l(P_2) = 0$ . Therefore, it is infeasible.
- (4)  $(\overline{P}_1, \overline{P}_2, \overline{P}_3)$  : We know that  $l(P_1), l(P_2), l(P_3) \geq z$ . Since  $\overline{P}_1$  contains  $P_2$  and  $P_3$  which are disjoint, we have  $l(\overline{P}_1) \geq l(P_2) + l(P_3) \geq 2z$ . Therefore, it is infeasible.

Therefore, if there are 3 disjoint paths on the restricted cycle ISPL, the only possible shortest path set for these 3 commodities are the disjoint paths.

□

Based on Lemma 3.4, we can construct a logic graph for the restricted cycle ISPL as in Figure 8. The vertex set is the same as in the general case logic graph, but the edge set is different. Two vertices are adjacent if the two paths are disjoint. Hence, for each triangle on the logic graph of the restricted cycle ISPL, all these paths must be included in the shortest path set. We can construct the logic graph and identify all triangles in polynomial time.

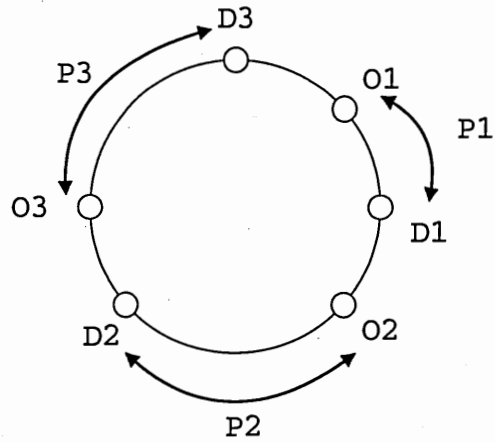


Figure 7: Restricted cycle has 3 disjoint paths

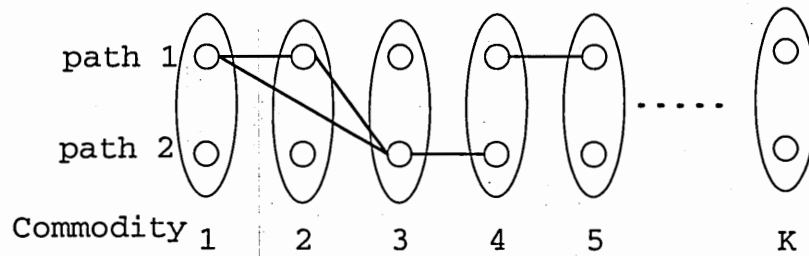


Figure 8: Logic graph of restricted cycle ISPL

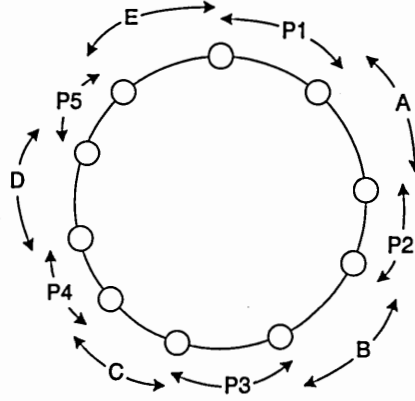


Figure 9: Logical graph with clique of size 5 has unique shortest path set

**Lemma 3.5.** *Suppose a restricted cycle ISPL is feasible and its logical graph has a clique of size 5. Then there is a unique shortest path for each commodity.*

*Proof.* Suppose a restricted cycle ISPL is feasible and its logical graph ( $G_L$ ) has a clique of size 5. Since there is a clique of size 5 in  $G_L$ , we know there are 5 commodities whose paths are disjoint from each other. By Lemma 3.4, we know the only feasible shortest paths for these commodities are these disjoint paths. Without loss of generality, we may assume these 5 commodities and their shortest paths as shown in Figure 9.

Suppose there is commodity  $i$  such that  $o_i \in P_1$ .

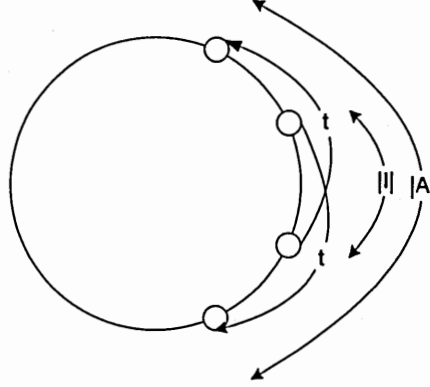
(1) If  $d_i \in P_1 \cup P_2 \cup P_5 \cup A \cup B \cup D \cup E$ , then there are 3 disjoint paths for commodities  $i$ , 3, and 4. By Lemma 3.4, if ISPL is feasible, then there is unique choice of path for commodity  $i$ .

(2) If  $d_i \in P_3$ , then there are disjoint paths for commodities  $i$ , 4, and 5. By Lemma 3.4, there is unique path for commodity  $i$  if the ISPL instance is feasible. The same argument can be applied when  $d_i \in P_4$ .

(3) If  $d_i \in C$ . Since there are two commodities whose paths are disjoint and totally contained in either path of commodity  $i$ , the ISPL instance must be infeasible.

Similar arguments can be applied when  $o_i \in A$ .

Since we picked 5 commodities which have disjoint paths arbitrarily, we know if the restricted cycle ISPL is feasible, there is a unique shortest path set for each commodity.  $\square$



**Figure 10:**  $A$  has no two disjoint paths

In order to discuss the relationship between disjoint paths and cycle size, we need the following definition.

**Definition 3.2.** A cycle ISPL is  $t$ -max if

$$t = \max \{|PC_i|, i = 1, 2, \dots, K\}$$

**Lemma 3.6.** If a restricted cycle ISPL is  $t$ -max with cycle size  $|C| \geq 4t + 1$ , then there must exist 3 disjoint paths on the cycle. It is equivalent to say that there is at least one triangle in the logic graph.

*Proof.* Consider a  $t$ -max restricted cycle ISPL instance on  $C = (N, E)$ . Define a node set  $A$  as following:

$$A = \{(a_1, a_2, \dots, a_m) | a_i \in N, (a_i, a_{i+1}) \in E \text{ for } i = 1, \dots, m - 1\}$$

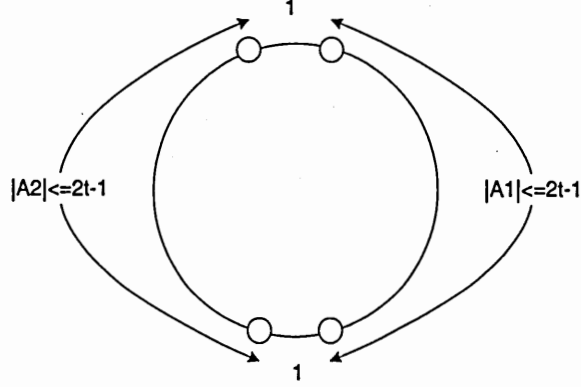
Choose  $A$  such that no two commodities that belong to  $A$  have disjoint paths. Then the intersection of paths of all commodities in  $A$ , say  $I$ , must be a path which is nonempty (see Figure 10). Let  $|A|$  and  $|I|$  denote the number of edges in  $A$  and  $I$ . Then we have

$$|A| \leq 2t - |I|.$$

Since  $|I| \geq 1$ ,  $|A| \leq 2t - 1$ .

Hence, the largest  $t$ -max restricted cycle ISPL instance which has no 3 disjoint paths must include  $A_1$  and  $A_2$  such that  $A_1 \cap A_2 = \phi$  (see Figure 11).

$$|C| \leq 2(2t - 1) + 2 = 4t$$



**Figure 11:** Largest restricted cycle ISPL with no 3 disjoint paths

Therefore, there must exist 3 disjoint paths when  $|C| \geq 4t + 1$ .  $\square$

**Lemma 3.7.** *Suppose a restricted cycle ISPL is  $t$ -max with cycle size  $|C|$ . If  $|C| \geq 5t + 1$ , then the only feasible solution of ISPL is to use the minimum cardinality paths for all commodities.*

*Proof.* Consider a restricted cycle ISPL which is  $t$ -max with cycle size  $|C| \geq 5t + 1$ . If we can show that for every commodity  $i$ , there are two commodities  $j$  and  $k$  whose minimum cardinality paths are disjoint and are both contained in the longer edge path of commodity  $i$ , then by Lemma 3.4, the lemma is proved.

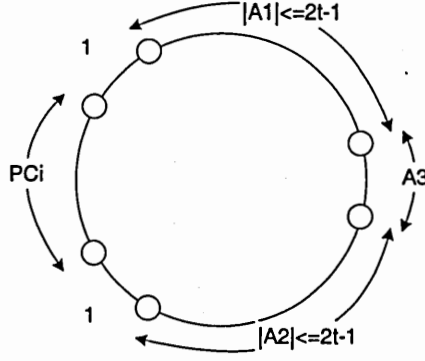
Consider any commodity  $i$ .  $PC_i$  and  $\overline{PC_i}$  denote the minimum cardinality path and the other (longer edge) path for commodity  $i$ . Since the problem is  $t$ -max and  $|C| \geq 5t + 1$ , we have

$$|PC_i| \leq t \text{ and } |\overline{PC_i}| \geq 4t + 1.$$

Let  $A$  be a set which contains all the nodes in  $\overline{PC_i}$  except for  $o_1$  and  $d_i$ .  $A$  satisfies the node set definition in the proof in Lemma 3.6 (see Figure 12). Since  $|\overline{PC_i}| \geq 4t + 1$ ,  $|A| \geq 4t - 1$ . Then  $A$  must include three subsets  $A_1$ ,  $A_2$  and  $A_3$  as in Figure , such that

$$|A_1| = |A_2| = 2t - 1$$

$$|A_3| = |A| - |A_1| - |A_2| \geq 1$$



**Figure 12:** All commodities have 3 disjoint paths

By the same argument in the proof of Lemma 3.6, there must exist two commodities  $j$  and  $k$  such that  $PC_j$  and  $PC_k$  are contained in  $A_1$  and  $A_2$ , respectively.

Since we pick commodity  $i$  arbitrarily, every commodity must use its shortest edge path by Lemma 3.4. Therefore, the only feasible solution of a  $t$ -max restricted cycle ISPL with  $|C| \geq 5t + 1$  is to use the minimum cardinality paths for all commodities.  $\square$

Now, we consider a special class of the cycle ISPL which has a restriction on the commodity distance. We refer to this concept as regularity.

**Definition 3.3.** *A ISPL instance is regular if the number of edges in the minimum cardinality path for all commodities are equal.*

**Definition 3.4.** *A ISPL instance is  $t$ -regular if ISPL is regular and the number of edges in each minimum cardinality path is equal to  $t$ .*

The definitions above are defined on the general ISPL without considering the underlying network. For example, ISPL on the complete graph is  $1$ -regular because the minimum cardinality paths for each commodity have exactly one edge.

For each commodity of cycle ISPL, let  $|PC_i|$  denote the number of edges contained in the smaller path for commodity  $i$ .

**Definition 3.5.** *A cycle ISPL is regular if the number of edges used in the smaller edge path for each commodity are all equal, i.e.,*

$$|PC_1| = |PC_2| = \dots = |PC_K|.$$

We now discuss the complexity of the restricted regular cycle ISPL.

**Lemma 3.8.** *If a restricted cycle ISPL is regular, then this ISPL instance must be feasible.*

*Proof.* We may assume that the restricted cycle ISPL is  $t$ -regular. Set the cost vector of the cycle equal to  $(\frac{z}{t}, \frac{z}{t}, \dots, \frac{z}{t})$ . Then we have a feasible solution since the shortest path length for all commodities is equal to  $z$ .  $\square$

**Lemma 3.9.** *If no two commodities in a cycle ISPL have disjoint paths, then the cycle ISPL must be regular.*

*Proof.* We will prove the lemma by contradiction.

Suppose the cycle ISPL is not regular. There must exist one commodity  $i$  such that  $|PC_i| \geq |PC_k|$  for all  $k = 1, 2, \dots, K$  and  $k \neq i$ . Moreover, there must exist at least one commodity  $j$  such that  $|PC_i| > |PC_j|$ . Without loss of generality, we may assume that  $|PC_1| \geq |PC_k|$  for all  $k = 2, \dots, K$ .

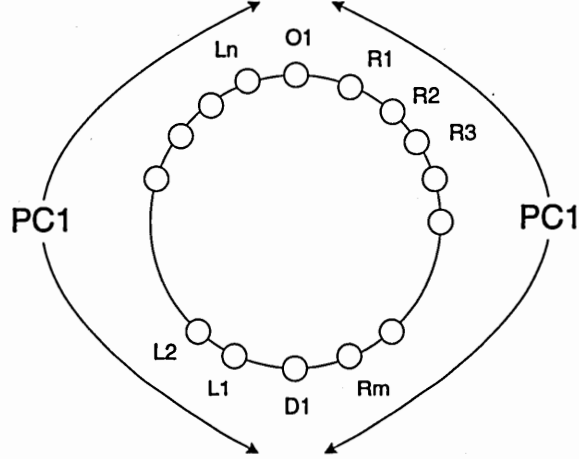
Recall that all the nodes in the cycle must be an origin or destination for at least one commodity. Let  $R$  denote all the nodes in  $PC_1$  except for  $O_1$  and  $D_1$ . Let  $L$  denote the node set on  $\overline{PC_1}$ . Then all the nodes in the cycle can be divided into two disjoint sets,  $R$ ,  $L$  by  $O_1$  and  $D_1$  (see Figure 13). Since  $R$  is the node set that is used by the smaller edge path for commodity 1, we have  $|R| = |PC_1| - 1$  and  $|R| \leq |L|$ .

If there is another commodity, say  $j$ , that shares either  $O_1$  or  $D_1$  as its origin or destination, then we can create a disjoint path set for commodities 1 and  $j$ . We may assume that the commodities share  $O_1$ .

- (1) if  $D_j = D_1$ , it is trivial that  $R$  and  $L$  are two disjoint path sets.
- (2) if  $D_j \in R$ , by choosing commodity 1 along  $L$  and commodity  $j$  along  $R$ , disjoint path sets are created.
- (3) if  $D_j \in L$ , by choosing commodity 1 along  $R$  and commodity  $j$  along  $L$ , disjoint path sets exist also.

Hence, no commodity can share either origin or destination with commodity 1. Moreover, the more general result that no two commodities can share either origin or destination can be reached by the same arguments.





**Figure 13:** Decompose cycle nodes into two sets

Since all the nodes in the cycle are an origin or destination for some commodity, if there exists a commodity whose origin and destination both belong to  $R$ , then a disjoint path set exists by argument (2). If they both belong to  $L$ , same result is reached by argument (3). Hence, origin and destination of any commodity other than 1 must be split: one in  $R$  and the other in  $L$ .

Since no two commodities can share an origin or destination, all the nodes in the cycle should be an endpoint for exactly one commodity. Also, the endpoints must lie in the different node sets  $R$  and  $L$ . Hence,  $|R| = |L|$ . Consider commodities  $2, \dots, K$ . Their origins and destinations must form a perfect matching as Figure 14.

Suppose the matching edges cross as in Figure 14, then we can construct a disjoint shortest path set for these two commodities. Since the matching cannot cross, the only possible matching should be like Figure 15. Then all  $|PC_i|$  should be equal, i.e., the cycle ISPL is regular. This is a contradiction.

Therefore, if no two commodities have disjoint paths, then the cycle ISPL must be regular. □

In this section, we use the concepts of disjoint path and logical graph to explore the complexity of cycle ISPL. We showed that restricted cycle ISPL is polynomially solvable when it satisfies one of the following conditions:

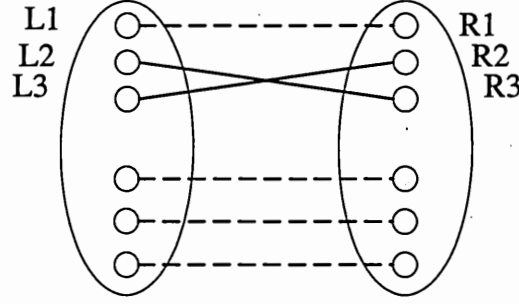


Figure 14: Matching origins and destinations for commodities

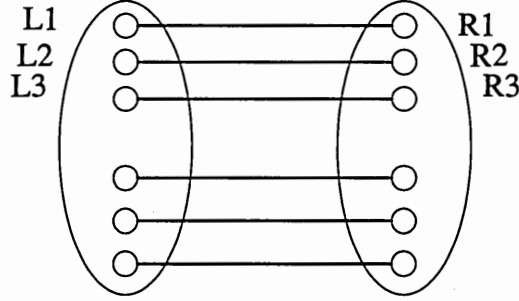


Figure 15: Perfect matching of origins and destinations with no cross edges

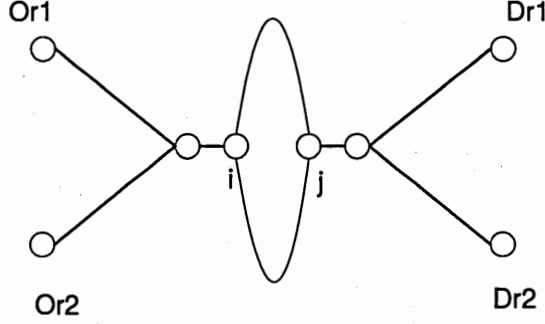
1.  $|C| \geq 5t + 1$  where  $t$  is the maximum number of edges of the minimum cardinality path for all commodities, or
2. the instance is  $t$ -regular.

### 3.3.3 ISPL Complexity and the Number of Commodities

We also study the complexity of a special case of the number of commodities. If there is only one commodity, ISPL can be trivially solved. The ISPL instance becomes a problem checking whether the given origin and destination are (strongly) connected on the (directed) graph, since we can put appropriate costs on the path if there is one (and infinite costs on all other edges). The following theorem shows that ISPL is also polynomially solvable when there are only two commodities.

**Theorem 3.6.** *The two-commodity ISPL is polynomially solvable.*

*Proof.* Suppose that there exist two commodities  $r_1$  and  $r_2$  with desired shortest path lengths  $z_{r_1}$  and  $z_{r_2}$ . Without loss of generality, we may assume that  $z_{r_1} \leq z_{r_2}$ . We may also



**Figure 16:** The union of two commodities path exists cycle

assume that there exists a subgraph  $G' \subseteq G$  such that  $o_{r_i}$  and  $d_{r_i}$  are connected on  $G'$  for  $i=1,2$  (where  $o_{r_i}$  and  $d_{r_i}$  are the origin and destination of commodity  $r_i$ ). Otherwise, the instance is infeasible. We can check this using breadth-first search (BFS) originated at each  $o_{r_i}$  separately, to find a path  $P_i$  from  $o_{r_i}$  and  $d_{r_i}$ . The running time of BFS is  $O(|N| + |E|)$ . If there is no such a path, then the instance is infeasible.

If  $P_1 \cap P_2 = \phi$  (so  $P_1$  and  $P_2$  are internally disjoint), then Lemma 3.2 shows how to find a solution in polynomial time.

Suppose that  $P_1 \cap P_2 \neq \phi$ . Without loss of generality, we may assume that  $P_1 \cap P_2 = P_3$  where  $P_3$  is a simple path. If  $P_3$  is not a simple path, then there exists at least one cycle in  $P_1 \cup P_2$  as in Figure 16. Since both  $P_1$  and  $P_2$  are shortest paths, the distance between  $i$  and  $j$  along  $P_1$  and  $P_2$  must be the same. Hence, we can set all the edge costs on this cycle equal to 0 without affecting the feasibility of ISPL. Thus, we can remove the cycles so  $P_3$  becomes some simple paths. Let  $c(P_i)$  denote the length of path  $P_i$ . Since  $P_3 \subseteq P_1$ ,  $0 \leq c(P_3) \leq z_{r_1}$ , and  $c(P_1 - P_3) = z_{r_1} - c(P_3) \geq 0$ . Similarly,  $c(P_2 - P_3) = z_{r_2} - c(P_3) \geq 0$ . Moreover,  $P_1 - P_3, P_2 - P_3, P_3$  are internally vertex disjoint ( $P_1 - P_3$ ,  $P_2 - P_3$ , and  $P_3$  may not be a single path; each might be the union of several internally vertex disjoint paths). The only infeasibility can happen when  $z_{r_1} < z_{r_2}$  and all  $P_1$  path contains a path for commodity 2. We can check the feasibility by doing BFS for commodity 1 on the induced subgraph with  $N - o_{r_2}$  and  $N - d_{r_2}$ . Hence, the two-commodity ISPL is polynomially solvable.

□

**Table 1:** Summary of the complexity result

	Type of network		
	Tree	$K_n$ with $k$ edges removed	$K_n$
undirected	polynomial	polynomial	polynomial
directed	polynomial	polynomial	polynomial

We now discuss the situation where the number of commodities is very large. The extreme case is that there is a commodity for each node pair  $(u, v)$ , i.e., the distance graph is a complete graph.

**Lemma 3.10.** *ISPL is polynomially solvable when the distance graph is complete.*

*Proof.* Suppose we are given an ISPL for which the distance graph is complete. Consider any edge  $e = (u, v) \in E$ . Suppose commodity  $i$  has origin/destination pair  $(u, v)$ . By Lemma 3.1, we have

$$c_e = z_i.$$

Since there is a commodity for each edge in the underlying graph, there is a unique cost vector  $c$  for this ISPL. The instance is feasible if and only if the shortest path length for all commodities is equal to the desired  $z_k$ . It can be checked in polynomial time. Therefore, ISPL is polynomially solvable when the distance graph is complete.  $\square$

### 3.4 Summary

Table 1 summarizes the results on the complexity of ISPL. We prove that ISPL is polynomially solvable whenever the underlying network is a tree, complete graph, or a complete graph with any constant  $k$  edges removed. These arguments are true for both directed and undirected graphs.

We also show that some the following classes of cycle ISPL are polynomially solvable.

1. Regular restricted cycle ISPL is polynomially solvable.
2. Restricted cycle ISPL with  $|C| \geq 5t+1$  is polynomially solvable when  $t$  is the maximum number of edges in any shortest edge path for any commodity on the induced cycle.

By Lemma 3.7 and Lemma 3.9, we know some restricted cycle ISPL are polynomially solvable. However, the complexity of some restricted ISPL and general cycle ISPL are still unknown. We discuss some disjoint path related properties that we can use to detect an infeasible shortest path set or decrease the number of choices of shortest path sets when we search for feasible solutions.

## CHAPTER IV

### ALGORITHMS

In this chapter, we propose some algorithms to solve the Inverse Shortest Path Length problem approximately. Since ISPL is NP-complete, we cannot find a polynomial algorithm for the exact solution except in some special cases (see Chapter 3, for example). The algorithms are mainly based on a path iteration scheme using constraint generation to deal with the problem of having exponentially many constraints. Three core subproblems of the ISPL algorithms are minimization of infeasibility, shortest path, and cost perturbation. The properties of these algorithms are discussed in this chapter. We also provide worst case analyses of these algorithms.

#### *4.1 Heuristic Ideas*

Two types of ideas will be covered in this section. The first one is a spanning tree enumeration scheme and the second is a path iteration algorithm. The spanning tree enumeration scheme tries to solve ISPL by taking advantage of the complexity of ISPL on trees (see Chapter 3). The path iteration scheme uses constraint generation to deal with the problem of having exponentially many constraints.

##### **4.1.1 Spanning Tree Enumeration Scheme**

We have proved that ISPL is polynomially solvable if the underlying network is a tree or a forest (Theorem 3.1). The direct way to apply this idea is to pick a spanning tree ( $T$ ), set it to be the solution, and check its feasibility. We can solve this restricted ISPL in polynomial time, and obtain costs  $\tilde{c}$ . If  $G' := (N, T, \tilde{c})$  is feasible, then  $G := (N, E, \tilde{c})$  is feasible for ISPL (because we can set the cost of all edges in  $E \setminus T$  to some large value  $M$ ). If  $G' := (N, T, \tilde{c})$  is not feasible, we have to choose another spanning tree to be a solution.

The problem with this heuristic method is that the performance mainly depends on the number of spanning trees in the network. The number of spanning trees may grow

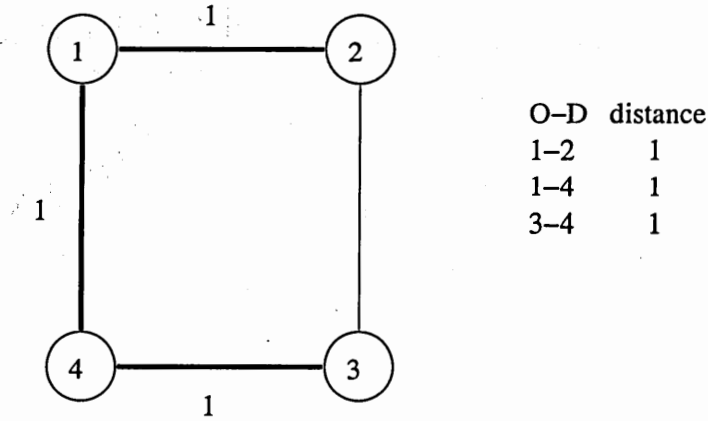


Figure 17: Only one spanning tree is feasible

exponentially and we cannot identify an infeasible instance without enumerating all possible spanning trees. For example, in Figure 17 there are 4 spanning trees on graph  $G$ , and only one of them is feasible.

The other problem is that a feasible ISPL instance may have a non-tree-type solution. For example, suppose  $z_{(1,3)} = z_{(2,4)} = z_{(3,5)} = z_{(4,1)} = z_{(5,2)} = 2$  as in Figure 18. In this case, the only feasible solution is a cycle and total enumeration of spanning trees cannot solve ISPL. Hence, we consider path iteration algorithms to solve ISPL.

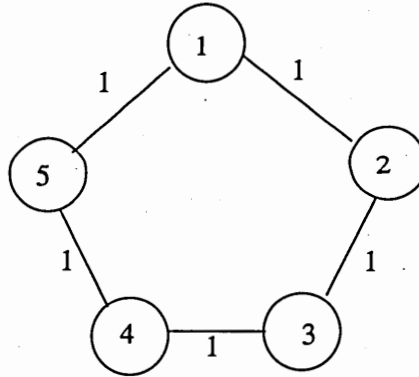
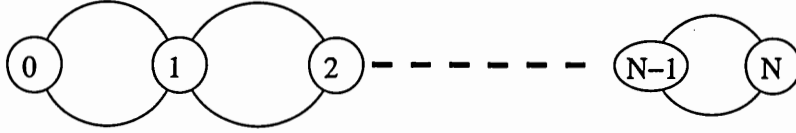


Figure 18: Example of the unique feasible solution is a cycle

#### 4.1.2 Path Iteration Algorithm

The main difficulties of ISPL are the non-convex feasible region and the exponential number of constraints. Even a simple graph could have an exponential number of constraints (see



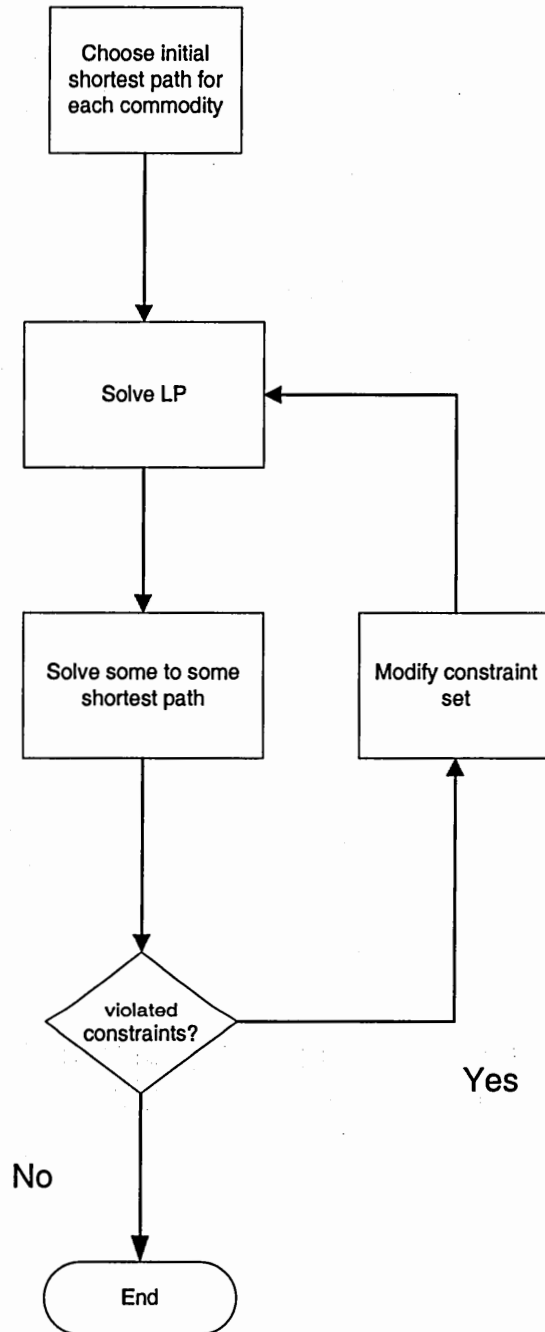
**Figure 19:** Example with exponentially many constraints

Figure 19, where the number of paths from node 0 to node  $N$  is  $2^N$ ). As we mentioned in Chapter 1, (1.10) is an implicit form. We do not only need one path for each commodity to satisfy the equality constraint (1.10), but also we must make sure no other path for the same O-D pair has shorter length. The number of constraints here depends on how many paths there are for each commodity. Moreover, since each path of each commodity could be the shortest one, the possible combinations of equality constraints could be  $|p_1| \cdot |p_2| \cdot \dots \cdot |p_K|$  (where  $|p_r|$  denotes the number of paths for commodity  $r$ ). A reasonable idea is to handle some specific constraints but not all at each iteration. Suppose we select a path for each commodity and set it to be the shortest path. The problem becomes identical to the original Inverse Shortest Path problem, and it can be solved in polynomial time.

First, we choose one path for each commodity and set it to be the shortest. Then we have an ISPP constraint set and solve the problem by solving a linear system. After every iteration, we solve the shortest path problem under the current costs, and identify violated constraints (the shortest path length for some commodity may be less than what we want). We add these constraints to the constraint set. Since the number of constraints is finite, this constraint generating scheme will terminate. Also, because violated constraints can be found in polynomial time, Khachiyan's theorem guarantees that we can optimize in polynomial time in spite of having exponentially many constraints. Figure 20 shows the flow chart for this algorithm.

**Theorem 4.1.** (*Khachiyan's theorem*) *Systems of rational linear inequalities, and linear programming problems with rational data, can be solved in polynomial time if a violated inequality can be found in polynomial time [26].*





**Figure 20:** Constraint generating scheme flow chart

---

**Constraint Generating Scheme Algorithm**

Step 0: Choose an initial shortest path for each commodity.

Step 1: Solve edge cost  $\tilde{c}$  by solving the system of linear equalities (equation 3.14) or simple linear programming.

Step 2: Solve for the shortest path for each commodity under the network  $G = (N, E, \tilde{c})$ . Find the violated constraints (paths), which have path length less than  $z_r$ .

Step 3: If there are no violated constraints, stop.

Step 4: Modify the constraint set by adding the violated constraints. Go to Step 1.

---

## 4.2 ISPL Algorithms

We now propose several heuristic algorithms to exploit the advantages of the constraint generating algorithm. Our algorithms contain three core subproblems: minimization of infeasibility, shortest path calculation, and cost perturbation. The main idea is that we add one slack variable for each commodity to represent the positive difference between the current and desired shortest path length.

$$\sum_{e \in SP_{o_i d_i}} c_e - \delta_i = z_i, \quad \text{for } i = 1, \dots, K \quad (4.1)$$

As before, we choose the shortest path set for each commodity in each iteration. Moreover, we check the shortest path set and make modifications at the end of each iteration, then re-optimize the linear program based on the previous iteration's result. Solving a some-to-some shortest path problem, minimizing the slack by linear programming, and perturbing the cost vector by linear programming are the core subproblems for our iterative re-optimization algorithms. We first discuss these three core subproblems.

### 4.2.1 Shortest Path Subproblem

As we mentioned in Section 4.1.2, there may exist exponentially many paths for each commodity. Finding all these paths and putting them into the constraint set of ISPL is time consuming and many of them may be redundant. Hence, we try to begin with only a few constraints, and add violated constraints by the constraint generating scheme. To do this, we have to solve the shortest path problem for each commodity and check whether the shortest path is better than the path in the shortest path set or is a violated constraint. Suppose each commodity  $r$  has O-D pair  $(o_r, d_r)$ . The shortest path subproblem for commodity  $r$

can be formulated as (4.2)-(4.4).

$$\text{Min} \quad \sum_{e=(i,j) \in E} c_{ij} x_{ij} \quad (4.2)$$

$$\sum_{\{j:(i,j) \in E\}} x_{ij} - \sum_{\{j:(j,i) \in E\}} x_{ji} = \begin{cases} 1 & \text{for } i = o_r \\ -1 & \text{for } i = d_r \\ 0 & \text{otherwise} \end{cases} \quad (4.3)$$

$$x_{ij} \geq 0 \quad \text{for all } (i, j) \in E \quad (4.4)$$

Many researchers have worked on the shortest path problem. Our version is the some-to-some problem in which we simultaneously find shortest path for several origin/destination pairs. Wang [30] provides a new polynomial algorithm to solve the some-to-some shortest path problem, and gives a detailed survey of previous work in multiple pairs shortest paths. The most well known algorithms, like Dijkstra's algorithm for one-to-all shortest path and the Floyd-Warshall algorithm for all-to-all shortest paths, can be used to solve our some-to-some shortest path subproblems in polynomial time. We use the Floyd-Warshall algorithm in our implementation.

#### 4.2.2 Minimization of Infeasibility

In Chapter 1, we noted that the objective function (1.9) could take any form since the goal of ISPL is to achieve feasibility. The main difficulty is to handle the implicit constraint set in (1.10), i.e., choosing a shortest path set to satisfy the desired shortest path lengths simultaneously. Hence, we choose the objective function to minimize the difference between the current and desired shortest path length of all commodities. Let  $\delta_i$  denote the positive slack difference for commodity  $i$ . The objective we address here is to minimize the infeasibility of the chosen shortest path set. The reason why we do not take the absolute value of slack is that we would like the solution to have shortest path lengths for each commodity that are at least  $z_k$ . Because our telecommunication application treats these  $z_k$  as prices, we assume the price must be set to allow some minimum profit level. The objective function is the following:

$$\text{Min} \sum_{i=1}^{|K|} \delta_i \quad (4.5)$$

Let  $P_{o_i d_i}$  be the path set of commodity  $i$  and  $SP_{o_i d_i}$  denote the shortest path of commodity  $i$  in this iteration. As we mentioned in Section 4.1.2, there might exist exponentially many possible paths for each commodity. So, we use constraint generation to increase the path set for each commodity in iterations when a violated path is found. This is the same as our constraint generation scheme. Hence, the constraint set will be the following:

$$\sum_{e \in SP_{o_i d_i}} c_e - \delta_i = z_i, \quad \text{for } i = 1, \dots, K \quad (4.6)$$

$$\sum_{e \in P} c_e \geq z_i, \quad \text{for } i = 1, \dots, K, \quad P \in P_{o_i d_i} \quad (4.7)$$

$$c \geq 0, \delta \geq 0 \quad (4.8)$$

It is trivial that the objective is always greater than or equal to zero. The optimal solution for re-optimization is zero. If we cannot find any path which length is smaller than the desired shortest path length for some commodity under the current cost vector, then the algorithm terminates.

#### 4.2.3 Cost Perturbation Subproblem

The third core subproblem in our algorithm is to perturb the cost vector without increasing the infeasibility. This can help us to generate more possible path sets to get a better solution. Moreover, the perturbation of the cost vector is a scheme to escape from a local optimal solution. Since the feasible region of ISPL is not convex, we need some heuristic scheme to escape from local optima. We use the cost perturbation scheme to do this.

The main idea of the perturbation scheme is to modify the costs by solving a linear programming problem when no further improvement is possible with constraint generation.

$$\text{Min } \sum_e r_e c_e \quad (4.9)$$

$$\sum_{e \in SP_{o_i d_i}} c_e = z_i + \delta_i, \quad \text{for } i = 1, \dots, K \quad (4.10)$$

$$\sum_{e \in P} c_e \geq z_i, \quad \text{for } i = 1, \dots, K, \quad P \in P_{o_i d_i} \quad (4.11)$$

$$c \geq 0 \quad (4.12)$$

We first randomly generate a perturbing coefficient,  $r_e$ , between 0 and 1 for each edge. The objective function is to minimize the summation of inner product the perturbation vector and cost vector (4.9). We also have to make sure that we will not increase the infeasibility after we perturb the cost vector. We use the slack solution  $\delta$  from (4.5)-(4.8) to change the right hand side of the first constraint sets (4.10). The second part of the constraint set guarantees that the perturbed cost vector will maintain feasibility for all the possible paths in hand (4.11). We know the constraint set (4.10)-(4.12) is consistent because the initial cost vector without any perturbation is a feasible solution. Since the objective value of (4.5) remains the same and we randomly generate the perturbing coefficient in each iteration, it is reasonable to say that randomly perturbing the costs can generate more possible paths without increasing the infeasibility.

### 4.3 Description of Algorithms

After finishing the discussion of the three core subproblems, we now introduce the detail of how the pieces of our iterative ISPL algorithms work together.

The basic idea is to begin with a set of paths, one for each commodity, and find the nearest-to-feasible solution using those paths. Then, we perturb the costs to create new shortest paths and iterate.

There are several variations we can use on this algorithm.

First, we can modify the shortest path set (SP set) for each commodity either whenever there is violated constraint found or at the end of each iteration. Second, we can use one of three cost perturbation schemes:

1. Do not perturb at all.
2. Perturb when there is no violated constraint found.
3. Perturb only when we reach a local optimum.

Based on the choices of changing the shortest path set and perturbing the cost, we propose 6 polynomial-time approximation algorithms to solve ISPL.

#### 4.3.1 Algorithm 0

In Algorithm 0, we modify the shortest path set only when there is no violated constraint found by solving shortest path subproblem, and do not perturb the cost vector. The detailed steps are shown in Algorithm 0 and Figure 21.

---

##### Algorithm 0

Step 0: Initialization. Choose one path for each commodity to be shortest.

Step 1: Solve infeasibility minimization subproblem.

- (1) Modify the constraint sets in (4.6) and (4.7) according to the shortest path set.
- (2) Optimize the linear programming problem (4.5)-(4.8) to get cost vector  $c$  and infeasibility variable  $\delta$ .
- (3) If the optimal value  $=0$ , i.e., sum of the infeasibilities are 0, stop: solution is optimal.

Step 2: Solve shortest path subproblem for each commodity  $i$ .

- (1) If there exist shortest path length of commodity  $i$  which is less than the desired length  $z_i$ , violated constraint found. Go to Step 1.

Step 3: Termination criteria test.

- (1) If any termination criteria is satisfied, stop.

Step 4: Modify shortest path set.

- (1) For every commodity  $i$ . If the shortest path length calculated in Step 2 is strictly less than the length of path in the shortest path set, put the shortest path for commodity  $i$  found in Step 2 into the shortest path set. Remove the old path constraint from (4.6) to (4.7).
  - (2) If any shortest path was changed, go to Step 1.
- 

#### 4.3.2 Algorithm 1

In Algorithm 1, we also do not perturb the cost vector. The difference between Algorithms 1 and 0 is that we modify the shortest path set at the end of each iteration. At the end of each iteration, we set the shortest paths of all commodities to be the shortest path set. The

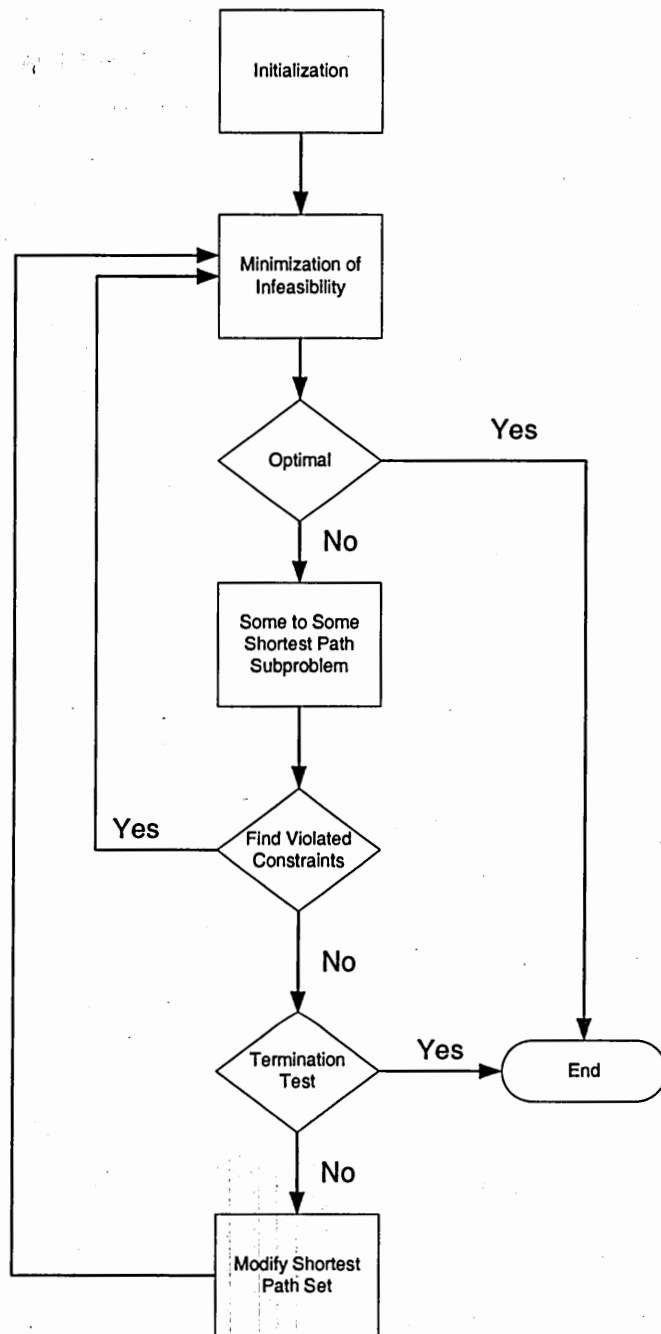


Figure 21: Algorithm 0

key idea to modify the shortest path at the end of each iteration is to save the iterations of running inner loop (find violated constraint and solve minimization of infeasibility). If we do not modify the shortest path set at each iteration, algorithm exits for the inner loop first time when all paths for each commodity are greater than desired length  $z_i$ . Alternative way to avoid too many running iterations of the inner loop is just to remove it by modifying the shortest path set at the end of each iteration. The detailed steps are shown in Algorithm 1 and Figure 22.

---

**Algorithm 1**

Step 0: Initialization. Choose one path for each commodity to be shortest.

Step 1: Solve infeasibility minimization subproblem.

- (1) Modify the constraint sets in (4.6) and (4.7) according to the shortest path set.
- (2) Optimize the linear programming problem (4.5)-(4.8) to get cost vector  $c$  and infeasibility variable  $\delta$ .
- (3) If the optimal value  $=0$ , i.e., sum of the infeasibilities are 0, stop: solution is optimal.

Step 2: Solve shortest path subproblem for each commodity  $i$ .

Step 3: Termination criteria test.

- (1) If any termination criteria is satisfied, stop.

Step 4: Modify shortest path set.

- (1) For every commodity  $i$ . If the shortest path length calculated in Step 2 is strictly less than the length of path in the shortest path set, put the shortest path for commodity  $i$  found in Step 2 into the shortest path set. Remove the old path constraint from (4.6) to (4.7).
  - (2) If any shortest path was changed, go to Step 1.
- 

### 4.3.3 Algorithms 2 & 3

We do not use cost perturbation subproblem in both Algorithms 0 and 1. Algorithm 2 and Algorithm 3 are the modifications of Algorithms 0 and 1 respectively by adding the cost perturbation subproblem. The benefit of applying perturbing cost vector will be discussed in Section 4.4.3. The detailed statements of Algorithm 2 are as shown in Algorithm 2 and Figure 22. Algorithm 3 is shown in Algorithm 3 and Figure 24.

### 4.3.4 Algorithms 4 & 5

Algorithms 4 and 5 are another modifications of Algorithms 0 and 1. In Algorithms 4 and 5, we still adopt the cost perturbation subproblem. But instead of perturbing the cost



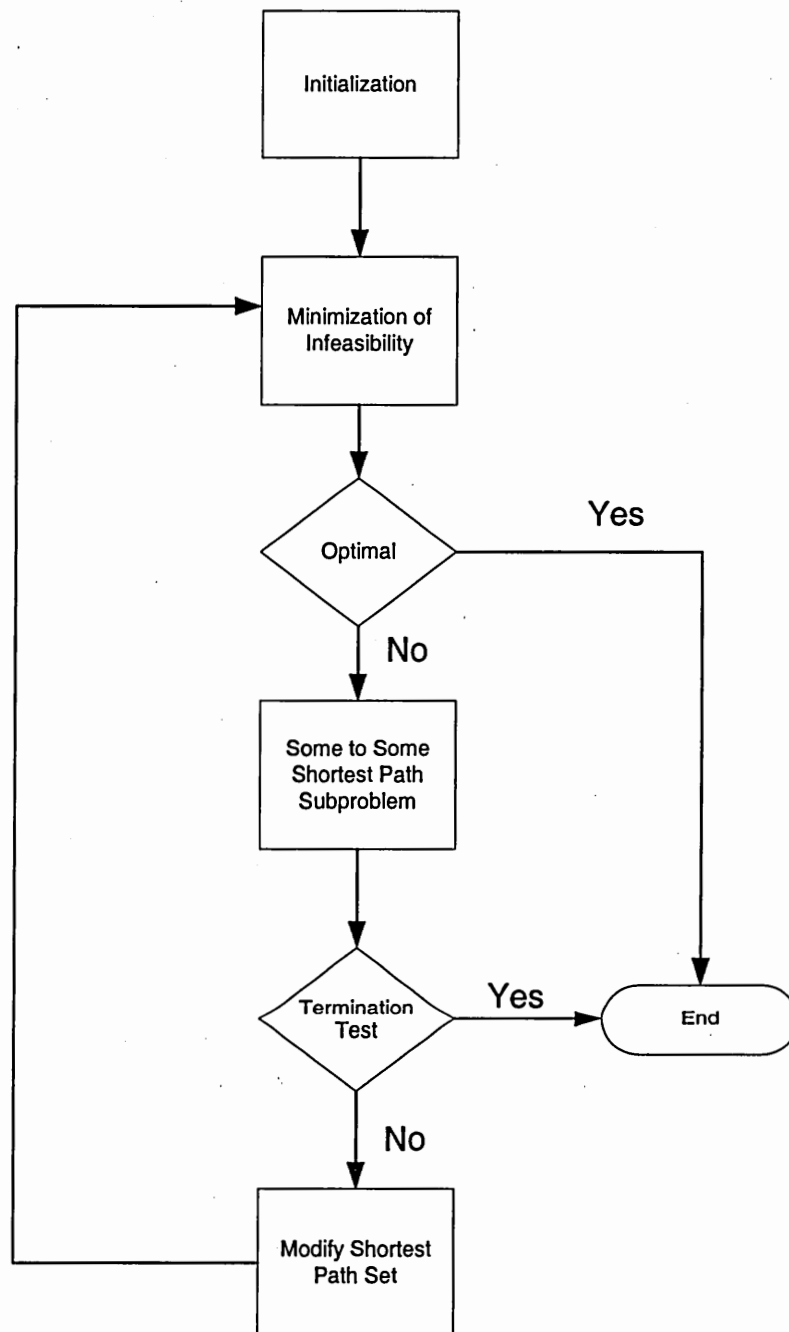


Figure 22: Algorithm 1

---

**Algorithm 2**

Step 0: Initialization. Choose one path for each commodity to be shortest.

Step 1: Solve infeasibility minimization subproblem.

- (1) Modify the constraint sets in (4.6) and (4.7) according to the shortest path set.
- (2) Optimize the linear programming problem (4.5)-(4.8) to get cost vector  $c$  and infeasibility variable  $\delta$ .
- (3) If the optimal value  $=0$ , i.e., sum of the infeasibilities are 0, stop: solution is optimal.

Step 2: Solve shortest path subproblem for each commodity  $i$ .

- (1) If there exist shortest path length of commodity  $i$  which is less than the desired length  $z_i$ , violated constraint found. Go to Step 1.

Step 3: Termination criteria test.

- (1) If any termination criteria is satisfied, stop.

Step 4: Modify shortest path set.

- (1) For every commodity  $i$ . If the shortest path length calculated in Step 2 is strictly less than the length of path in the shortest path set, put the shortest path for commodity  $i$  found in Step 2 into the shortest path set. Remove the old path constraint from (4.6) to (4.7).

Step 5: Solve cost perturbation subproblem.

- (1) Generate random perturbing coefficients  $r$ .
  - (2) Optimize the linear programming problem (4.9)-(4.12) to get cost vector  $c'$ .
  - (3) Update cost vector to be  $c'$ .
  - (4) Go to Step 1.
- 

---

**Algorithm 3**

Step 0: Initialization. Choose one path for each commodity to be shortest.

Step 1: Solve infeasibility minimization subproblem.

- (1) Modify the constraint sets in (4.6) and (4.7) according to the shortest path set.
- (2) Optimize the linear programming problem (4.5)-(4.8) to get cost vector  $c$  and infeasibility variable  $\delta$ .
- (3) If the optimal value  $=0$ , i.e., sum of the infeasibilities are 0, stop: solution is optimal.

Step 2: Solve shortest path subproblem for each commodity  $i$ .

Step 3: Termination criteria test.

- (1) If any termination criteria is satisfied, stop.

Step 4: Modify shortest path set.

- (1) For every commodity  $i$ . If the shortest path length calculated in Step 2 is strictly less than the length of path in the shortest path set, put the shortest path for commodity  $i$  found in Step 2 into the shortest path set. Remove the old path constraint from (4.6) to (4.7).

Step 5: Solve perturbation subproblem.

- (1) Generate random perturbing coefficients  $r$ .
  - (2) Optimize the linear programming problem (4.9)-(4.12) to get cost vector  $c'$ .
  - (3) Update cost vector to be  $c'$ .
  - (4) Go to Step 1.
-

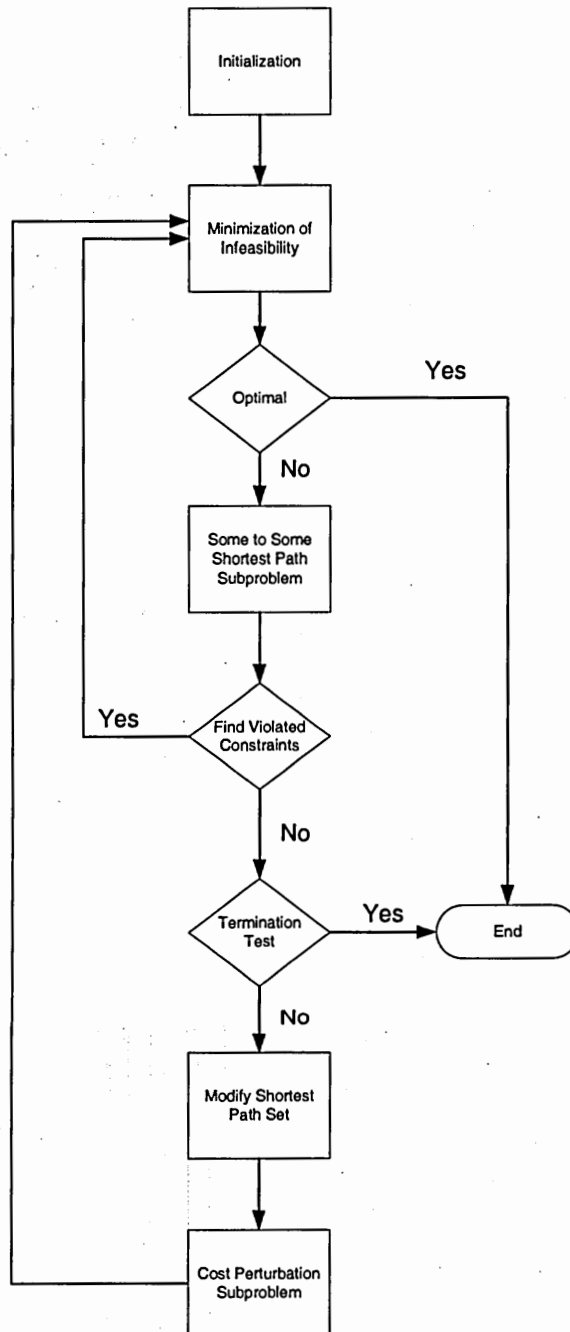
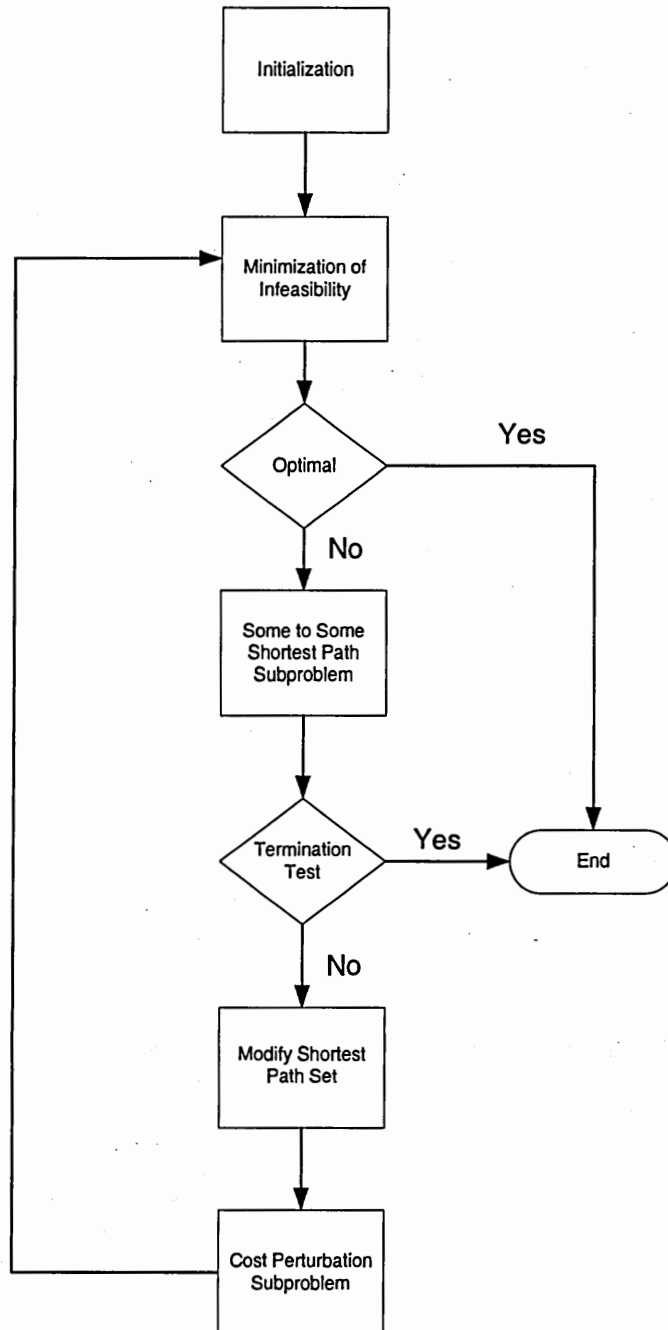


Figure 23: Algorithm 2



**Figure 24: Algorithm 3**

vector at the end of each outer iteration, Algorithms 4 and 5 perturb the cost only when Algorithms 0 and 1 cannot find better path to modify the shortest path set, i.e., Algorithms 0 and 1 reach a local optimal solution.

We perturb the cost until one of the following conditions is reached:

1. Find better path to modify the shortest path set.
2. Find violated constraint.
3. Reach the perturbation iteration limit or time limit.

Algorithms 4 and 5 are shown in Algorithm 4, Algorithm 5, Figure 25, and Figure 26.

---

#### Algorithm 4

Step 0: Initialization. Choose one path for each commodity to be shortest.

Step 1: Solve infeasibility minimization subproblem.

- (1) Modify the constraint sets in (4.6) and (4.7) according to the shortest path set.
- (2) Optimize the linear programming problem (4.5)-(4.8) to get cost vector  $c$  and infeasibility variable  $\delta$ .
- (3) If the optimal value  $=0$ , i.e., sum of the infeasibilities are 0, stop: solution is optimal.

Step 2: Solve shortest path subproblem for each commodity  $i$ .

- (1) If there exist shortest path length of commodity  $i$  which is less than the desired length  $z_i$ , violated constraint found. Go to Step 1.

Step 3: Termination criteria test.

- (1) If any termination criteria is satisfied, stop.

Step 4: Modify shortest path set.

- (1) For every commodity  $i$ . If the shortest path length calculated in Step 2 is strictly less than the length of path in the shortest path set, put the shortest path for commodity  $i$  found in Step 2 into the shortest path set. Remove the old path constraint from (4.6) to (4.7).
- (2) If any shortest path was changed, go to Step 1.

Step 5: Solve cost perturbation subproblem.

- (1) Generate random perturbing coefficients  $r$ .
  - (2) Optimize the linear programming problem (4.9)-(4.12) to get cost vector  $c'$ .
  - (3) Update cost vector to be  $c'$ .
  - (4) Go to Step 2.
-

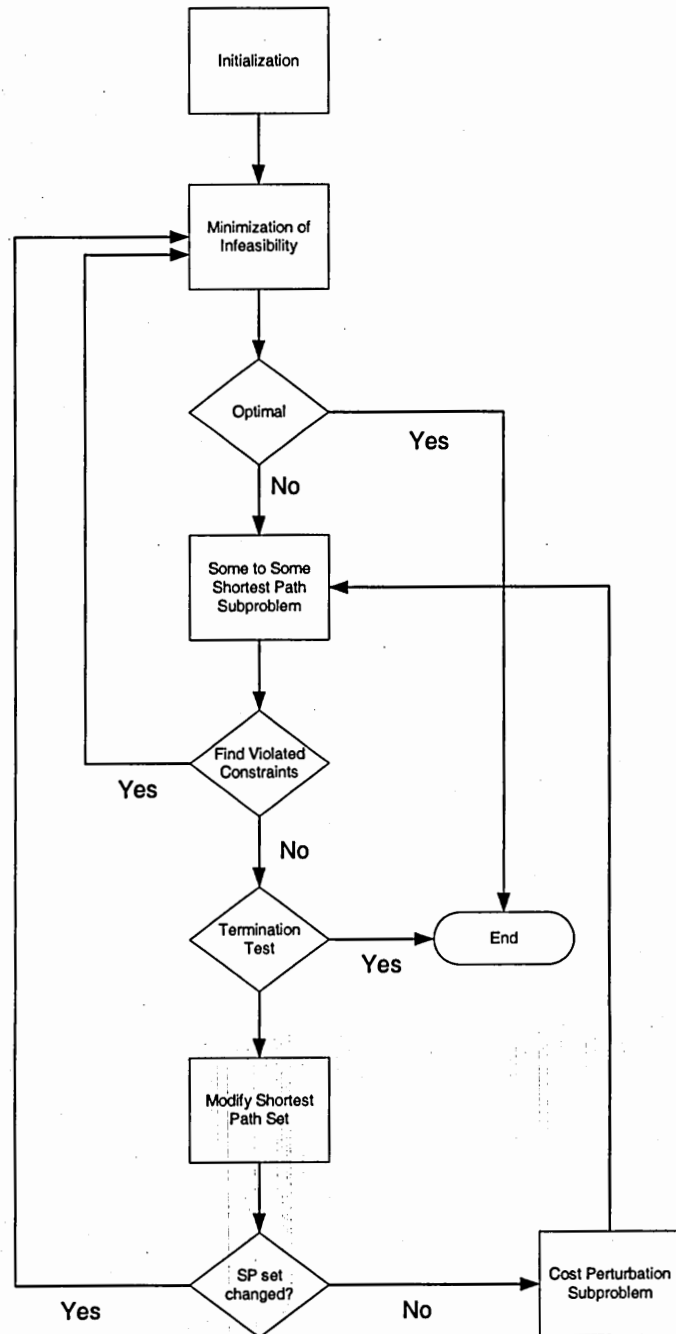


Figure 25: Algorithm 4

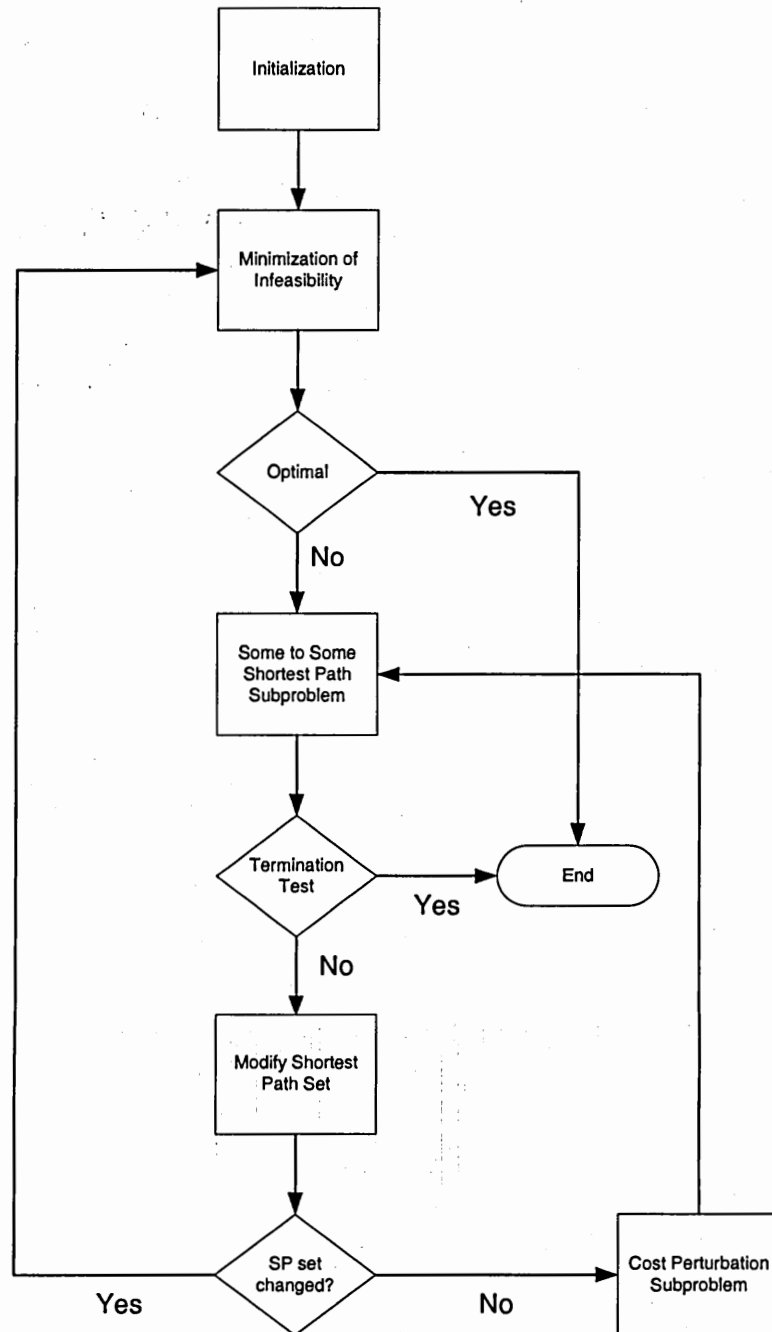


Figure 26: Algorithm 5

---

**Algorithm 5**

Step 0: Initialization. Choose one path for each commodity to be shortest.

Step 1: Solve infeasibility minimization subproblem.

- (1) Modify the constraint sets in (4.6) and (4.7) according to the shortest path set.
- (2) Optimize the linear programming problem (4.5)-(4.8) to get cost vector  $c$  and infeasibility variable  $\delta$ .
- (3) If the optimal value  $=0$ , i.e., sum of the infeasibilities are 0, stop: solution is optimal.

Step 2: Solve shortest path subproblem for each commodity  $i$ .

Step 3: Termination criteria test.

- (1) If any termination criteria is satisfied, stop.

Step 4: Modify shortest path set.

- (1) For every commodity  $i$ . If the shortest path length calculated in Step 2 is strictly less than the length of path in the shortest path set, put the shortest path for commodity  $i$  found in Step 2 into the shortest path set. Remove the old path constraint from (4.6) to (4.7).
- (2) If any shortest path was changed, go to Step 1.

Step 5: Solve cost perturbation subproblem.

- (1) Generate random perturbing coefficients  $r$ .
  - (2) Optimize the linear programming problem (4.9)-(4.12) to get cost vector  $c'$ .
  - (3) Update cost vector to be  $c'$ .
  - (4) Go to Step 2.
- 

## **4.4 Properties of Algorithms**

### **4.4.1 Initialization of the Algorithm**

To initialize the algorithm, we select one path for each commodity to be the initial shortest path. We consider two possible ways to do this job. One is to choose initial shortest path set randomly and the other is to choose the minimum cardinality path for each commodity.

We adopt both schemes to initialize the algorithms and test their performance in Section 5.3. The reason we use the minimum cardinality path is the following. Since the complexity of finding a feasible solution of ISPL is the same as finding disjoint path for each commodity, it seems a reasonable direction to reach this goal by choosing the minimum cardinality path for each commodity. Moreover, we can think of our path iteration algorithms for ISPL as starting with all edge costs equal to a very small number  $\epsilon$ . Hence, the first shortest path for each commodity should be the first violated constraint, the path using the smallest number of edges.

After initialization, the algorithm's performance depends on finding good paths. In



order to get better performance, the key is to increase the size of path sets to enlarge the feasible region. At the same time, we must avoid including redundant paths to increase complexity and computation time.

#### 4.4.2 Termination of the Algorithm

We will discuss some termination criteria in this section. Since our path iteration algorithm is a heuristic algorithm, it cannot be guaranteed to reach the optimal solution of ISPL. We set termination criteria to guarantee that the algorithm will terminate.

1. If the minimum infeasibility is equal to 0 when solving (4.5)-(4.8), the ISPL feasible solution is reached. The algorithm reports a feasible cost vector for the ISPL instance.
2. If the minimum infeasibility is less than some  $\epsilon$ , the solution is very close to optimal. The algorithm reports the cost vector and the tolerance level.
3. We also set the following termination criteria.
  - (a) Time. Some ISPL instances might be too large to solve in a reasonable amount of time. Hence, we set a time limit for the algorithm. If the algorithm reaches the time limit  $T$ , it terminates and reports the solution with termination status. Based on the size of ISPL and the heuristic ISPL solution, we can adjust the parameter  $T$ .
  - (b) Iteration count. For those algorithms that contain the cost perturbation scheme, we need an iteration limit to avoid falling into an infinite loop. Also, for Algorithms 4 and 5, we keep perturbing the cost vector to improve the current solution when Algorithms 0 and 1 terminate. Hence, we need two iteration limits. The first limit  $I$  is the maximum consecutive unchanged iteration. When the cost vector has not changed for  $I$  iterations of perturbation, the algorithms terminate. The second is the perturbation iteration limit  $M$ . If the minimum infeasibility is unchanged for  $M$  consecutive cost perturbations, we terminate the algorithm. Since we randomly perturb the cost vector in each sub-iteration, the cost vector itself might not be the same.  $M$  must be large enough that the

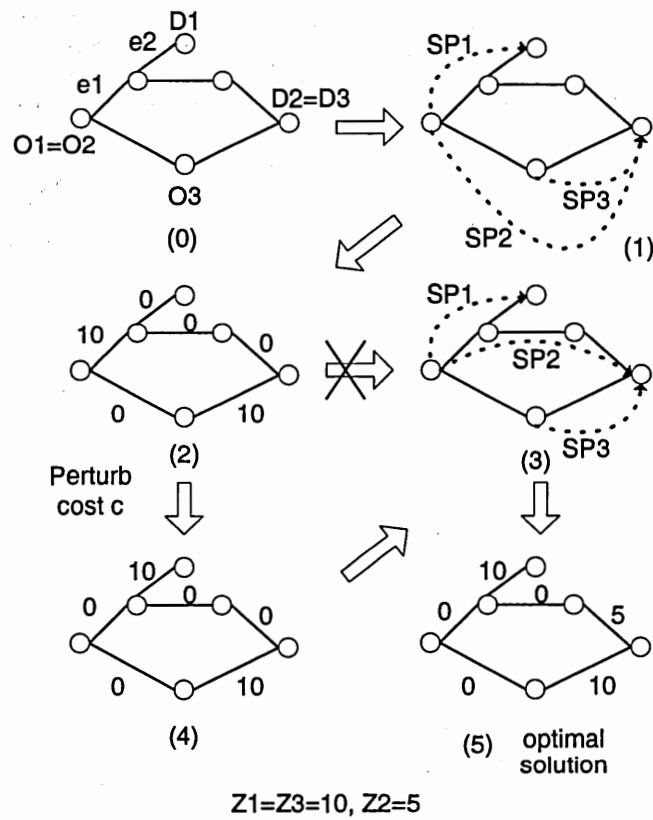
algorithm can improve the current solution. On the other hand,  $M$  should not be too big since we might have no chance to improve current solution. Hence, the parameter  $M$  can effect the algorithm performance in terms of both solution and computation time. We will explain the detail in Section 4.4.3 and Section 4.4.4.

Using the above termination criteria, the algorithm is guaranteed to terminate.

#### 4.4.3 Perturbation Benefit

We will discuss why we need the perturbation subproblem in this section. We can see the benefit clearly in Figure 27. There are three commodities in the ISPL instance in stage (0). Suppose we initialize the algorithm by choosing each minimum cardinality path as the shortest path. Then the shortest path for each commodity is shown in stage (1). We minimize the summation of positive slack and obtain the solution shown in stage (2). Suppose we did not perturb the cost vector. Then the algorithm would terminate because there are no violated path constraints. However, if we perturb the cost from stage (2) to (4), then a violated constraint for commodity 2 will be found and the shortest path set will be recalculated as in stage (3). So, we can reach the optimal solution of (5) by re-optimization of the sum of positive slacks in this instance.

Some questions need to be answered since perturbing the cost vector properly can provide a better ISPL solution. Does there exist a consistent way to perturb the cost vector to guarantee reaching a better solution? Unfortunately, we cannot find a perturbation scheme that can guarantee reaching a better solution for every ISPL instance because the complexity of ISPL depends on the structure of the underlying network and the corresponding desired shortest path lengths. Moreover, the complexity of general ISPL is  $NP$ -complete. Based on the fact that all the subproblems in our algorithms can be solved in polynomial time and the inner step of the algorithm terminates in a polynomial number of re-optimization iterations, if there exists a consistent scheme to improve the solution using a polynomial number of perturbation iterations, the general ISPL can be solved in polynomial time. Hence, it seems impossible to find a consistent perturbing scheme for all ISPL instances. Therefore,



**Figure 27:** Perturb the cost to reach a better solution

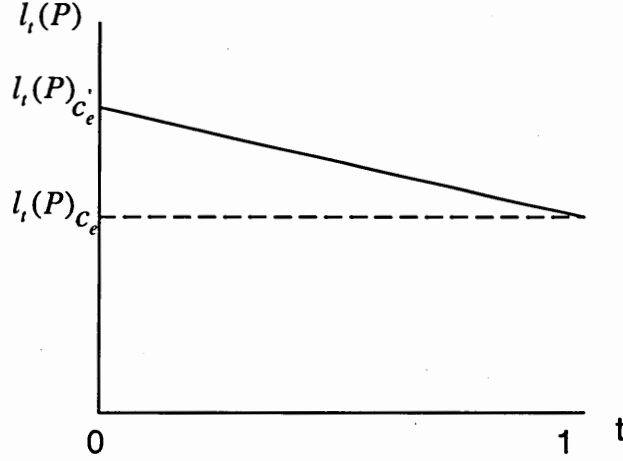


Figure 28: Linear function of length of a path

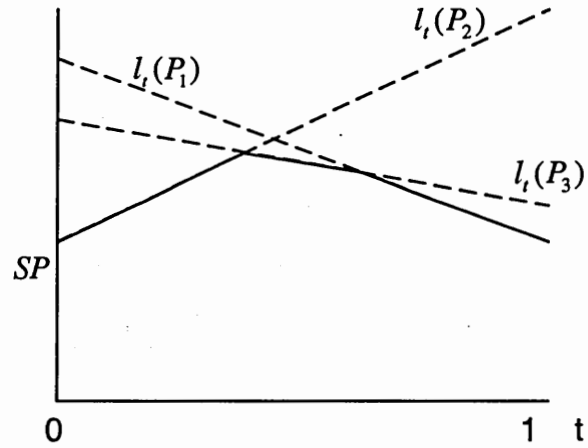
we choose a random perturbing scheme to perturb the cost vector in each sub-iteration.

Let  $c$  and  $c'$  be the two cost vectors found from solving the minimization of the sum of positive slacks and from perturbing, respectively. Notice that  $tc + (1 - t)c'$  is still feasible for both linear programming problems when  $0 \leq t \leq 1$ . We now consider whether there is a line search method to find a good  $t$  to generate better paths.

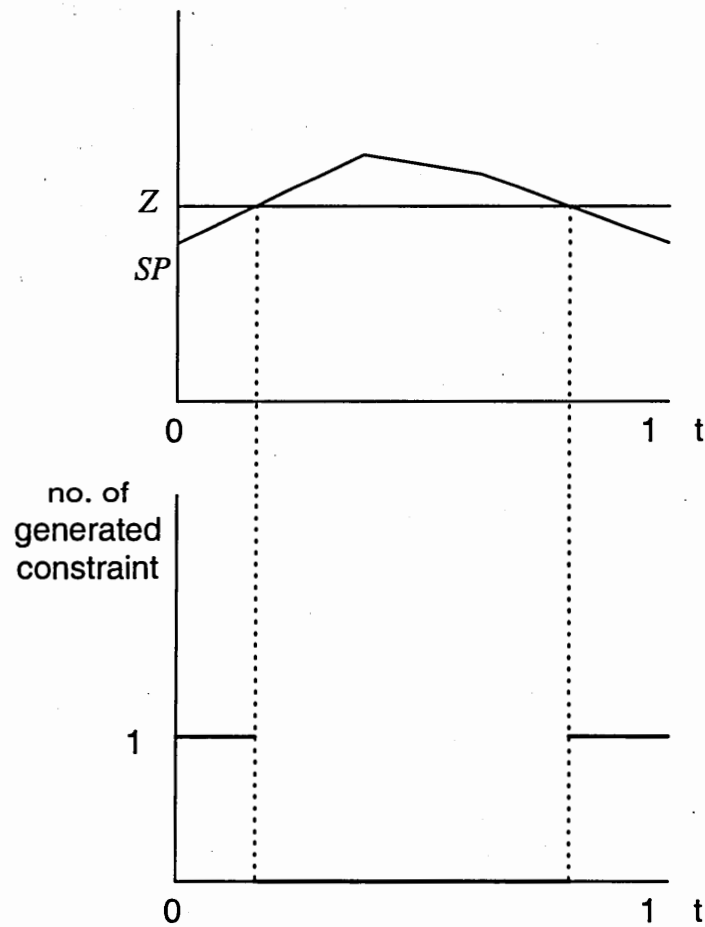
Let  $P$  denote a path and let  $l_t(P)$  be the length of  $P$  under  $tc + (1 - t)c'$ .

$$l_t(P) = \sum_{e \in P} (tc_e + (1 - t)c'_e) = \sum_{e \in P} c'_e + t \sum_{e \in P} (c_e - c'_e) \quad (4.13)$$

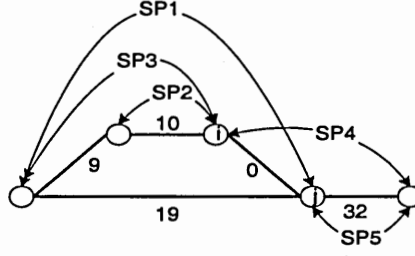
The path length is linear function of  $t$  as in Figure 28. Consider any commodity. The shortest path length function under cost vector  $tc + (1 - t)c'$  is a piecewise linear concave function (Figure 29). In order to generate a path for this commodity, the length of the new path should be strictly less than the desired  $z$ . In Figure 30, we can see clearly that the range of  $t$  for which a new path constraint is generated is not convex, and the indicator function for the new path generation is neither convex nor concave. Consequently, the sum of these functions for all commodities is neither convex nor concave. Hence, the only way to find the  $t$  that generates the most new paths is total enumeration. Therefore, we instead use the random perturbing scheme to get  $c'$  as the cost vector for next iteration and do not adopt a line search scheme in the algorithm.



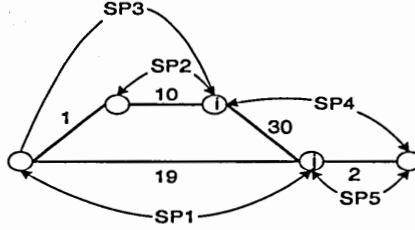
**Figure 29:** Shortest path length function for one commodity is piecewise linear



**Figure 30:** Number of constraints generated is neither convex nor concave



**Figure 31:** Example of cost perturbation does not work



**Figure 32:** Feasible cost of ISPL

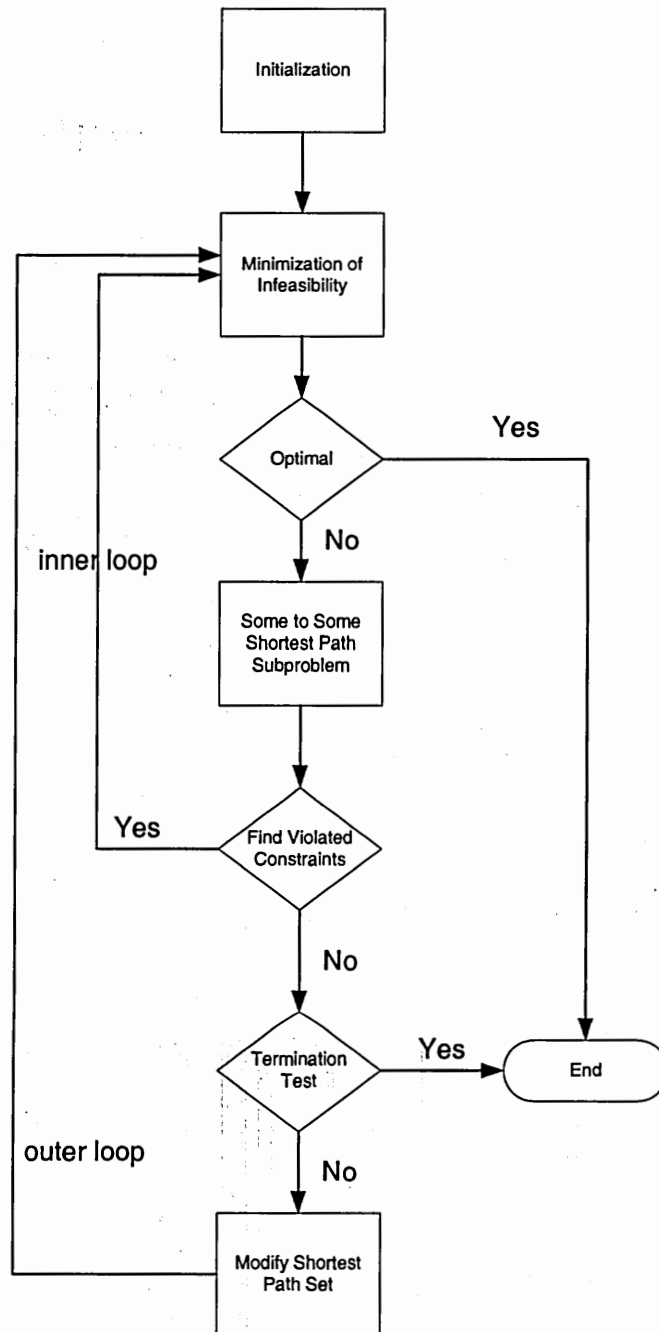
Cost perturbation allows the algorithms to escape from a local optimal solution (see Figure 27). The other important property of the cost perturbation is that it may not guarantee to escape from a local optimal solution. In some instances, our algorithms cannot escape from a local optimal solution no matter how many cost perturbation iterations we use. For example, there is a 4-commodity ISPL in Figure 31 with desired shortest path length  $z = (19, 10, 11, 32, 2)$ . The ISPL instance is feasible because we can set the cost as in Figure 32.

#### 4.4.4 Monotonicity Properties

We discuss some properties of our ISPL algorithms in this section.

Consider Algorithms 0, 2, and 4 in which we have an inner loop and an outer loop as identified in Figure 33.

First, the objective value in (4.5) is non-decreasing in the inner loop. The reason is that the inner loop is finding violated constraints in (4.7). Hence, the feasible region of next iteration in the inner loop (4.6)-(4.8) is strictly smaller than the one for the previous iteration. Since the shortest path set remains the same ((4.6) does not change) and the



**Figure 33: Inner loop and outer loop**

feasible region becomes smaller, the objective value must be at least as big as the one in the previous iteration. Therefore, the objective value is non-decreasing in the inner loop.

For the outer loop, consider two consecutive infeasibility minimization problems (P1) and (P2). Let  $SP_{o_i d_i}^1$  and  $SP_{o_i d_i}^2$  be the specified shortest paths for commodity  $i$  in problems (P1) and (P2). Let  $J$  be the set of commodities that switch specified shortest paths from (P1) to (P2).

(P1)

$$\begin{aligned} w_1 &= \min \sum_{i=1}^{|K|} \delta_i \\ \sum_{e \in SP_{o_i d_i}^1} c_e - \delta_i &= z_i, \text{ for } i = 1, \dots, |K| \\ \sum_{e \in P} c_e &\geq z_i, \text{ for } i = 1, \dots, |K|, P \in P_{o_i d_i} \\ c_e, \delta_i &\geq 0 \end{aligned}$$

(P2)

$$\begin{aligned} w_2 &= \min \sum_{i=1}^{|K|} \delta_i \\ \sum_{e \in SP_{o_i d_i}^2} c_e - \delta_i &= z_i, \text{ for } i = 1, \dots, |K|, i \notin J \\ \sum_{e \in SP_{o_i d_i}^2} c_e - \delta_j &= z_j, \text{ for } j \in J \\ \sum_{e \in P} c_e &\geq z_i, \text{ for } i = 1, \dots, |K|, P \in P_{o_i d_i} \\ c_e, \delta_i &\geq 0 \end{aligned}$$

Let  $(c^1, \delta^1)$  and  $(c^2, \delta^2)$  be the optimal solutions for (P1) and (P2).

**Lemma 4.1.** *If  $J \neq \emptyset$ , then  $w_2 < w_1$ .*

*Proof.* Let  $\delta_i^* = \sum_{e \in SP_{o_i d_i}^2} c_i^1 - z_i$  for every commodity  $i$ . If  $i \notin J$ , then  $SP_{o_i d_i}^1 = SP_{o_i d_i}^2$ , so  $\delta_i^* = \delta_i^1$ . If  $i \in J$ , then it must be that  $\sum_{e \in SP_{o_i d_i}^2} c_e < \sum_{e \in SP_{o_i d_i}^1} c_e$ . Therefore,

$$\delta_i^* = \sum_{e \in SP_{o_i d_i}^2} c_i^1 - z_i < \sum_{e \in SP_{o_i d_i}^1} c_i^1 - z_i.$$



Since  $J \neq \phi$ ,

$$\sum_{i=1}^{|K|} \delta_i^* < \sum_{i=1}^{|K|} \delta_i^1 = w_1.$$

Also, since  $(c^1, \delta^*)$  is feasible for  $(P2)$ ,

$$\sum_{i=1}^{|K|} \delta_i^* \geq \sum_{i=1}^{|K|} \delta_i^2 = w_2.$$

Thus,

$$w_2 \leq \sum_{i=1}^{|K|} \delta_i^* < w_1.$$

□

**Lemma 4.2.** *The objective of the outer loop is strictly decreasing.*

*Proof.* This follows directly from Lemma 4.1. □

Lemma 4.2 provides us with an important piece of information: these algorithms will not cycle. Since the objective value of the outer loop is strictly decreasing, there is no way for Algorithms 2 and 3 to fall into a cycle. Hence, we do not need any cycle prevention scheme.

However, some of the algorithms do have a portion that can cycle in some specific circumstances. When we try to escape from a local optimal solution by cost perturbation in Algorithms 4 and 5, the solution might remain the same or jump back and forth from iteration to iteration. To guarantee the algorithms' termination, we have already set several criteria in Section 4.4.2.

## 4.5 Worst-Case Bounds

We propose several heuristic algorithms to solve ISPL. In this section, we give bounds on the worst case performance of the algorithms. Since Algorithms 2, 3, 4 and 5 contain random perturbation subproblems, it is difficult to evaluate the bounds of these algorithms precisely. However, because these 4 algorithms are based on Algorithms 0 and 1, we can evaluate the worst case bounds for Algorithms 0 and 1 as the base case.

There are two subproblems in Algorithms 0 and 1, minimization of infeasibility and shortest path. The shortest path problem is used for finding violated constraints or changing

the shortest path set. It will not affect the worst case bound. Hence, we will focus on the worst case analysis of the minimization of infeasibility problem.

Consider the minimization of infeasibility subproblem (4.5)-(4.8). Let  $\pi_1, \dots, \pi_{|K|}$  be the dual variables for (4.6) and  $\rho_{k_j}$  denote dual variable for the  $j$ -th path constraint of commodity  $k$  in (4.7). Then, the dual formulation of minimization of infeasibility is the following:

$$Max \sum_{k=1}^{|K|} \pi_k z_k + \sum_{k=1}^{|K|} \sum_{j=1}^{|P_k|} \rho_{k_j} z_k \quad (4.14)$$

$$-\pi_k \leq 1, \quad k = 1, 2, \dots, |K| \quad (4.15)$$

$$\sum_{k: e \in SP_k} \pi_k + \sum_{k=1}^{|K|} \sum_{j: e \in P_{k_j}} \rho_{k_j} \leq 0, \quad \text{for every } e \in E \quad (4.16)$$

$$\pi_1, \pi_2, \dots, \pi_{|K|} \text{ unrestricted, all } \rho_{k_j} \geq 0 \quad (4.17)$$

Finding an upper bound on the minimization of infeasibility (4.5)-(4.8) is equivalent to finding a lower bound for its dual problem (4.14)-(4.17). Since for any feasible solution  $(\pi, \rho)$  to (4.14)-(4.17), the solution  $(\pi, 0)$  is also feasible and has an objective function no greater. Any lower bound on (4.14)-(4.17) is also a lower bound on an unrestricted form of the dual with no  $\rho_{k_j}$ . Therefore, the bounding problem can be reduced to the following:

$$Max \sum_{k=1}^{|K|} \pi_k z_k \quad (4.18)$$

$$-\pi_k \leq 1, \quad k = 1, 2, \dots, |K| \quad (4.19)$$

$$\sum_{\{k: e \in SP_k\}} \pi_k \leq 0, \quad \text{for every } e \in E \quad (4.20)$$

$$\pi_1, \pi_2, \dots, \pi_{|K|} \text{ unrestricted} \quad (4.21)$$

Since the objective value of the primal is non-negative, the dual objective value must be non-negative as well. If the initial shortest paths are totally disjoint, then we have an optimal solution where all  $\pi_k = 0$  by (4.19) and (4.20). Hence, the optimal value when minimizing infeasibility should be zero, and ISPL is solved. Therefore, if the optimal objective value is strictly greater than zero, then the paths in the shortest path set must be not disjoint.

We first discuss absolute performance bounds, and then discuss relative bounds. Before finding a general worst-case bound, we first consider two special cases.

Lemma 4.3 and Lemma 4.4 are the worst case absolute bounds when solving the minimization of infeasibility problem for an instance with two and three commodities.

**Lemma 4.3.** *Given an instance of ISPL with 2 commodities ( $z_1 \leq z_2$ ), the worst case performance bound of our algorithms when minimizing infeasibility is the following:*

- (1) 0, when  $z_1 = z_2$ ,
- (2)  $z_2 - z_1$ , when  $z_2 > z_1$ .

*Proof.* Suppose we are given an instance of ISPL with 2 commodities and the desired lengths,  $z_1 \leq z_2$ . Let  $SP_1$  and  $SP_2$  denote the chosen shortest paths for commodity 1 and 2. Let  $P = SP_1 \cap SP_2$ . There are 3 possible situations:

- (1)  $P = \phi$ .
- (2)  $P \neq \phi$ .  $P \neq SP_1$  and  $P \neq SP_2$ .
- (3)  $P \neq \phi$ .  $P = SP_1$  or  $P = SP_2$ .

In case (1) and (2), there are edges that belong only to  $SP_1$  and edges that belong only to  $SP_2$ . By (4.20), we have

$$\pi_1 \leq 0 \text{ and } \pi_2 \leq 0 \text{ for these edges.}$$

Hence, the optimal objective value (4.18) must be equal to 0. In other words, solving the minimization of infeasibility problem will get a feasible solution for ISPL.

Suppose case (3) happens. If  $SP_1 \subset SP_2$ , then our algorithm will find a feasible solution to ISPL: put a total of  $z_1$  on the arcs of  $SP_1$ ,  $z_2 - z_1$  on the arcs of  $SP_2 \setminus SP_1$ , and large costs on all other arcs. Therefore, to construct the worst case, we assume  $SP_2 \subset SP_1$ . By (4.20), we have

$$\pi_1 + \pi_2 \leq 0 \text{ for } e \in P$$

$$\pi_1 \leq 0 \text{ for } e \in SP_1 \setminus P.$$

- (1) Suppose  $z_1 = z_2$ . Since  $\pi_1 + \pi_2 \leq 0$ , the objective function (4.18) will be following:

$$\pi_1 z_1 + \pi_2 z_2 = (\pi_1 + \pi_2) z_1 \leq 0.$$

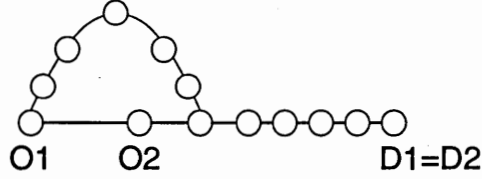


Figure 34: Feasible ISPL instance with 2 commodities

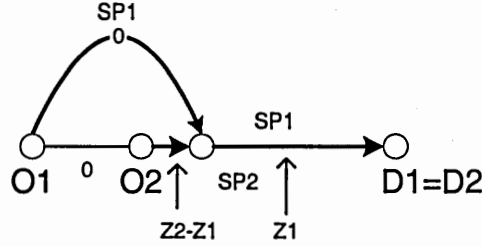


Figure 35: Feasible ISPL solution and shortest paths

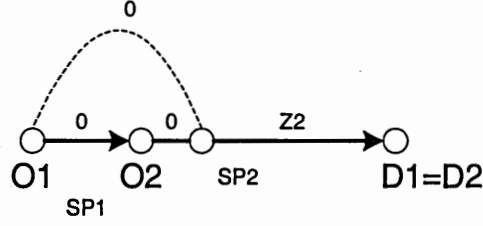
Hence, the optimal objective value of (4.18) is 0. Therefore, the worst case for  $z_1 = z_2$  is 0.

(2) Suppose  $z_1 < z_2$ . By (4.19), we have  $-1 \leq \pi_1 \leq 0$ . Then the objective function (4.18) is following:

$$\begin{aligned} \pi_1 z_1 + \pi_2 z_2 &\leq \pi_1 z_1 - \pi_1 z_2 \\ &= \pi_1 (z_1 - z_2) \\ &\leq z_2 - z_1. \end{aligned}$$

Hence, the worst case performance bound for 2-commodity ISPL is  $z_2 - z_1$ .  $\square$

Consider a 2-commodity ISPL as in Figure 34. This ISPL is feasible since we can choose the shortest paths and set costs as in Figure 35. Suppose we choose the shortest path for each commodity by using the shortest number of edges. Then solving the minimization of infeasibility problem might give costs as in Figure 36. We cannot improve the solution since there is no shorter path for each commodity (if we do not perturb the cost). Hence, the bound of Lemma 4.3 is tight.



**Figure 36:** Worst solution for 2-commodity ISPL

**Lemma 4.4.** *Suppose we are given an instance of ISPL with 3 commodities ( $z_1 \leq z_2 \leq z_3$ ). Then the worst case performance bound of our algorithm for minimizing infeasibility is the following:*

$$\text{Max } \{z_3 + z_2 - z_1, 2z_3 - z_2 - z_1\}.$$

*Proof.* Let  $(\pi_1, \pi_2, \pi_3)$  be the variables for commodities 1, 2, and 3 in (4.18)-(4.21). For each edge in the graph, there are 4 possibilities of how many commodities' shortest paths are on it: no commodity, only one commodity, exactly two commodities or all three commodities. We only have to consider the edges which are used by some commodities.

(1) One commodity, say  $i$ : Then we have

$$-1 \leq \pi_i \leq 0. \quad (4.22)$$

(2) Exactly two commodities, say  $i$  and  $j$ : We have

$$-1 \leq \pi_i \quad (4.23)$$

$$-1 \leq \pi_j \quad (4.24)$$

$$\pi_i + \pi_j \leq 0. \quad (4.25)$$

(4.23)-(4.25) imply

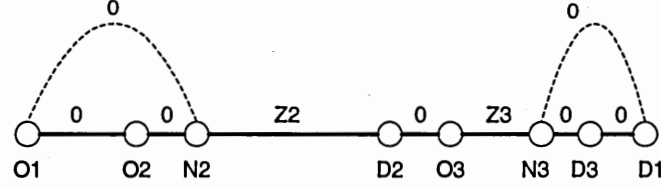
$$\pi_i, \pi_j \leq 1. \quad (4.26)$$

(3) All 3 commodities: Then we have all  $\pi_i \geq -1$  and

$$\pi_1 + \pi_2 + \pi_3 \leq 0. \quad (4.27)$$

Then we also have

$$\pi_i \leq 2 \text{ for } i = 1, 2, 3. \quad (4.28)$$



**Figure 37:** The performance bound on this ISPL with 3 commodities is tight at  $Z_3 + Z_2 - Z_1$

Since  $z_1 \leq z_2 \leq z_3$ , the worst case can happen only at one of the following two combinations of  $\pi$ :  $(-1, -1, 2)$  or  $(-1, 1, 1)$ . therefore, the worst case for 3-commodity ISPL must  $\max \{2z_3 - z_2 - z_1, z_3 + z_2 - z_1\}$ .  $\square$

Both bounds in Lemma 4.4 are tight.

The first bound is tight as illustrated in Figure 37. In Figure 37, the shortest path length for commodities 1, 2, and 3 are  $z_2 + z_3$ ,  $z_2$ , and  $z_3$ . All four paths of commodity 1 have the same length,  $z_2 + z_3$ . The algorithms will terminate because there is no violated constraint or shorter path for commodity 1 to modify the shortest path set. The difference between optimal and our solution is

$$(z_2 + z_3 - z_1) + (z_2 - z_2) + (z_3 - z_3) = z_2 + z_3 - z_1.$$

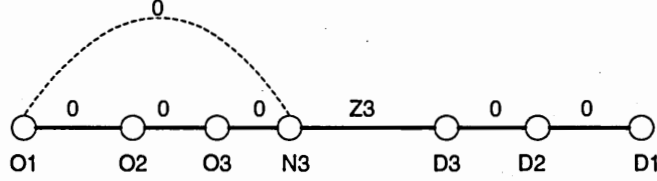
ISPL in Figure 37 can be solved optimally if we can generate the path constraint for commodity 1 which uses both upper curves.

The second bound is shown to be tight in Figure 38. In Figure 38, the shortest path lengths for all commodities are equal to  $z_3$ . Hence, the difference will be

$$(z_3 - z_1) + (z_3 - z_2) + (z_3 - z_3) = 2z_3 - z_2 - z_1.$$

Since all paths for commodities 1 and 2 have the same length,  $z_3$ , no path constraint can be generated. Algorithms terminate at this local optimal solution. The ISPL in Figure 38 can be solved optimally when we choose paths using the upper curve to be the shortest for commodities 1 and 2.

Because of the way the performance bounds for two and three commodities relies on the desired shortest path lengths  $z_i$ , our algorithm's performance can be arbitrarily bad as



**Figure 38:** The performance bound on this ISPL with 3 commodities is tight at 2Z3-Z2-Z1

some  $z_i$  gets arbitrarily large.

**Lemma 4.5.** *Given an instance of ISPL with  $|K|$  commodities  $(z_1, \dots, z_{|K|})$ , the worst case performance bound of our algorithms for minimizing infeasibility is  $\sum_{k=1}^{|K|} (|K| - 1)z_k$ .*

*Proof.* Suppose we are given an ISPL instance with  $|K|$  commodities. Let  $\pi = (\pi_1, \dots, \pi_{|K|})$  be the corresponding dual variables defined in (4.18)-(4.21).

By (4.20), we have

$$\sum_{\{k: e \in SP_k\}} \pi_k \leq 0, \quad \text{for every } e \in E.$$

Hence,

$$\pi_i \leq \sum_{\{k: e \in SP_k, k \neq i\}} (-\pi_k), \quad \text{for every commodity } i \text{ and } e \text{ such that } e \in SP_i.$$

Let  $|\omega_e|$  denote the number of commodities such that  $e \in SP_k$ . Since

$$-\pi_k \leq 1, \quad k = 1, 2, \dots, |K|,$$

we have

$$\pi_i \leq \sum_{\{k: e \in SP_k, k \neq i\}} (-\pi_k) \leq |\omega_e| - 1.$$

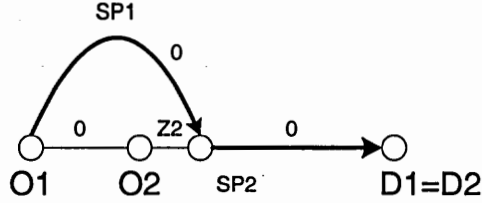
Because  $|\omega_e| \leq |K|$ ,

$$\pi_i \leq |K| - 1.$$

The dual objective value (4.18) provides a bound on the minimum infeasibility. Therefore, the bound for minimizing infeasibility can be no worse than

$$\sum_{k=1}^{|K|} \pi_k z_k \leq \sum_{k=1}^{|K|} (|K| - 1) z_k.$$

□



**Figure 39:** A feasible solution for 2-commodity ISPL worst case

The bound in Lemma 4.5 is not tight. Since all  $\pi_k$  cannot be equal to  $|K| - 1$  simultaneously, it violates the constraint (4.20). But this bound could be arbitrarily close to the minimum infeasibility. We will introduce such an example later (see Figure 42) when we discuss the relative performance of algorithms. By Lemma 4.5, we know the worst case for solving the minimization of infeasibility could be arbitrarily bad when  $z_i \rightarrow \infty$  for some commodity  $i$ .

We note that these performance bounds assume that the perturbation step is not used; in fact, we use the perturbation specifically to allow our algorithm to escape from bad local optima.

Suppose we adopt cost perturbation when we reach the bad local optimal solutions as in Figure 36, Figure 37, and Figure 38. The 2-commodity ISPL worst case in Figure 36 can be solved to optimality when the cost is perturbed as in Figure 39. Cost perturbation might shift the costs in Figure 37 to Figure 40. Then the desired shortest path for commodity 1 can be generated (see Figure 40) and our algorithms will solve this ISPL to optimality. For the ISPL in Figure 38, the desired paths for commodities 1 and 2 will be generated when costs after perturbation are as in Figure 41. Therefore, algorithms with perturbation have chances to solve the previous three worst case ISPLs to optimality.

We now discuss the worst case relative performance of our algorithms. We evaluate the performance of the algorithms by the ratio  $d$  between a current solution and the optimal solution. It is equal to sum of all current shortest path lengths divided by the sum of all



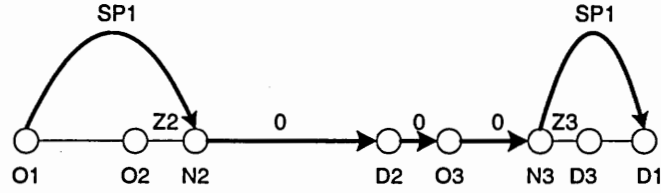


Figure 40: A feasible solution for 3-commodity ISPL worst case 1

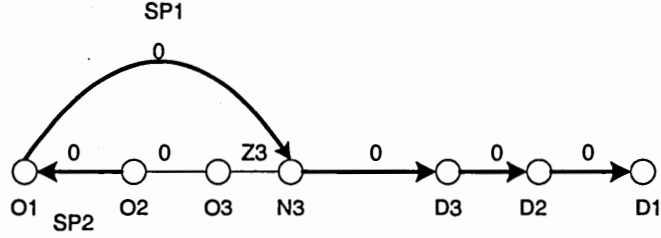


Figure 41: A feasible solution for 3-commodity ISPL worst case 2

desired shortest path lengths (4.29).

$$d = \frac{\sum_{i=1}^{|K|} (l(SP_i) - z_i)}{\sum_{i=1}^{|K|} z_i} = \frac{\sum_{i=1}^{|K|} \delta_i}{\sum_{i=1}^{|K|} z_i} \quad (4.29)$$

**Lemma 4.6.**  $d \leq |K| - 1$ .

*Proof.* Suppose a feasible ISPL has  $|K|$  commodities with desired shortest path lengths

$z_1, z_2, \dots, z_{|K|}$ .

By Lemma 4.5, we have

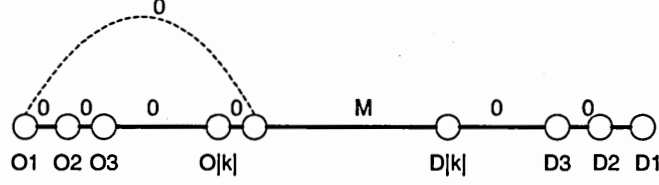
$$\begin{aligned}
d &= \frac{\sum_{i=1}^{|K|} \delta_i}{\sum_{i=1}^{|K|} z_i} \\
&\leq \frac{\sum_{k=1}^{|K|} (|K| - 1) z_k}{\sum_{i=1}^{|K|} z_i} \\
&= \frac{(|K| - 1) \left( \sum_{i=1}^{|K|} z_i \right)}{\sum_{i=1}^{|K|} z_i} \\
&= |K| - 1.
\end{aligned}$$

□

This performance ratio demonstrates that even as the problem data  $z_i$  get large, our algorithms will achieve a  $O(|K|) = O(|N|^2)$  approximation ratio. We now show that the bound on this ratio is tight.

Consider the ISPL in Figure 42. Let  $(z_1, z_2, \dots, z_{|K|-1}, z_{|K|}) = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_{|K|-1}, M)$  satisfy

$$\begin{aligned}
\varepsilon_1 &\leq \varepsilon_2 \leq \dots \leq \varepsilon_{|K|-1} \ll M, \\
\frac{\varepsilon_i}{M} &\approx 0 \text{ for all } i = 1, \dots, |K| - 1.
\end{aligned}$$



**Figure 42:** The worst performance of the algorithms is  $d=|K|-1$

Then the worst performance of algorithms will be

$$\begin{aligned}
 d &= \frac{|K| M - \sum_{i=1}^{|K|-1} \varepsilon_i - M}{\sum_{i=1}^{|K|-1} \varepsilon_i + M} \\
 &= \frac{|K| - \sum_{i=1}^{|K|-1} \frac{\varepsilon_i}{M} - 1}{\sum_{i=1}^{|K|-1} \frac{\varepsilon_i}{M} + 1} \\
 &\approx |K| - 1.
 \end{aligned}$$

Hence, the bound in Lemma 4.6 is tight.

We know it is possible to escape from a bad local optimal solution by cost perturbation. We discuss the probability of escaping from a local optimal solution in the worst case. Consider commodity 2 of the ISPL instance in Figure 40. After cost perturbation, there is exactly one edge, with the smallest perturbing coefficient, whose cost is equal to  $z_2$  and costs on all the other edges of the shortest path for commodity 2 are equal to 0. Let  $|N_i N_j|$  denote the number of edges between  $N_i$  and  $N_j$ . Then the probability of the positive cost edge being  $(O_2, N_2)$  should be  $\frac{|O_2 N_2|}{|O_2 D_2|}$ . It means we have  $\frac{|O_2 N_2|}{|O_2 D_2|}$  probability of finding a better path for commodity 1. There are two possible paths for commodity 1 that could be that found in this situation: one will improve the minimum infeasibility solution from  $z_2 + z_3 - z_1$  to  $z_3 - z_1$ , and the other will get a feasible ISPL solution directly. By the same argument, we know that there is  $\frac{|N_3 D_3|}{|O_3 D_3|}$  probability for commodity 3 to shift  $z_3$  to some edge on  $(N_3, D_3)$ . The minimum infeasibility could be improved from  $z_2 + z_3 - z_1$  either to  $z_2 - z_1$  or to 0.

Hence, the probability of escaping from this local optimum is

$$\begin{aligned} p_{\text{escape}} &= \frac{|O_2 N_2|}{|O_2 D_2|} \left(1 - \frac{|N_3 D_3|}{|O_3 D_3|}\right) + \frac{|N_3 D_3|}{|O_3 D_3|} \left(1 - \frac{|O_2 N_2|}{|O_2 D_2|}\right) + \frac{|O_2 N_2|}{|O_2 D_2|} \times \frac{|N_3 D_3|}{|O_3 D_3|} \\ &= \frac{|O_2 N_2|}{|O_2 D_2|} + \frac{|N_3 D_3|}{|O_3 D_3|} - \frac{|O_2 N_2|}{|O_2 D_2|} \times \frac{|N_3 D_3|}{|O_3 D_3|}. \end{aligned}$$

Moreover, the probability to escape from this local optimum directly to global optimum is

$$p_{\text{escape}} = \frac{|O_2 N_2|}{|O_2 D_2|} \times \frac{|N_3 D_3|}{|O_3 D_3|}.$$

Since we can reach the optimal solution after perturb the costs as in Figure 40, we can calculate the expected value of the solution after one perturbation as the following:

$$\begin{aligned} E \left( \sum_{i=1}^{|K|} \delta_i \right) &= \frac{|O_2 N_2|}{|O_2 D_2|} \left(1 - \frac{|N_3 D_3|}{|O_3 D_3|}\right) (z_3 - z_1) + \frac{|N_3 D_3|}{|O_3 D_3|} \left(1 - \frac{|O_2 N_2|}{|O_2 D_2|}\right) (z_2 - z_1) \\ &\quad + \left(1 - \frac{|O_2 N_2|}{|O_2 D_2|} \times \frac{|N_3 D_3|}{|O_3 D_3|}\right) \times (z_2 + z_3 - z_1). \end{aligned}$$

Notice that the escape probability could be very small when  $\frac{|O_2 N_2|}{|O_2 D_2|} \rightarrow 0$  and  $\frac{|N_3 D_3|}{|O_3 D_3|} \rightarrow 0$ .

We can do the same analysis for the ISPL instance in Figure 41. The escaping probability and expectation after one perturbation will be the following:

$$\begin{aligned} p_{\text{escape}} &= \frac{|O_3 N_3|}{|O_3 D_3|}, \\ E \left( \sum_{i=1}^{|K|} \delta_i \right) &= \left(1 - \frac{|O_3 N_3|}{|O_3 D_3|}\right) \times (2z_3 - z_2 - z_1). \end{aligned}$$

The probability to escape from a local optimal solution ( $p_{\text{escape}}$ ) can be used to determine a limit on the number of cost perturbation iterations. Let  $p_{\text{escape}}$ ,  $p$ , and  $I$  denote two probabilities and the iteration limit of cost perturbation. Using (4.30), we can calculate the number of iterations  $I$  we need such that there is probability  $p$  to escape from a local optimum that has escaping probability  $p_{\text{escape}}$ :

$$\sum_{i=0}^{I-1} (1 - p_{\text{escape}})^i p_{\text{escape}} \geq p. \quad (4.30)$$

## 4.6 Summary

We summarize some properties of the ISPL algorithms in this section.

1. All the subproblems (minimization of infeasibility, some-to-some shortest path, and cost perturbation) take polynomial time.
2. The solution for the inner loop of algorithms is non-decreasing, and the solution for the outer loop is strictly decreasing (see Lemma 4.2).
3. The worst case absolute bound for 2-commodity ISPL is  $(z_2 - z_1)$ , and the bound for 3-commodity ISPL is  $Max \{z_3 + z_2 - z_1, 2z_3 - z_2 - z_1\}$ . The expected values for the worst case absolute bounds while algorithms applying cost perturbation are also calculated.
4. The worst case absolute bounds could be arbitrarily bad, but the worst case relative bound is  $|K| - 1$ .
5. All 6 ISPL algorithms are guaranteed to terminate in polynomial time.

## CHAPTER V

### COMPUTATIONAL RESULTS

In this chapter, we introduce three schemes to generate random ISPL instances to test the performance of our ISPL algorithms. We also provide some examples to illustrate the progress of our algorithms, in Section 5.2. Section 5.3 contains the computational results and analysis for random ISPL instances, including both intermediate-difficulty and hard ISPL instances. In Section 5.4, we test our algorithms on three real telecommunication instances, using networks from Tyco, Level 3, and Global Crossing. We summarize the results of the performance of the algorithms in Section 5.5.

#### *5.1 Random ISPL Instance Generation*

In order to test the performance of ISPL algorithms, we first need a set of ISPL instances. A feasible ISPL instance is hard to generate randomly. Given an instance, to check whether it is a feasible is as hard as solving ISPL. Thus, we choose to construct an instance by putting costs on edges and finding their shortest path lengths. We randomly determine the network topology, and then assign random costs to each edge. Using these costs, we find the shortest path length for each commodity  $k$ , and use these lengths as the desired  $z_k$ . Because our random costs are a feasible solution, we can guarantee this instance is feasible no matter how many origin/destination pairs we choose to be commodities.

In Chapter 4, we discussed ways of selecting an initial shortest path set for our algorithms. When testing the performance of our algorithms, we want to avoid too much similarity between the shortest paths used to determine  $z_k$  and the initial shortest paths for each commodity. In the worst case, if the initial shortest path set is totally the same as the  $z_k$  shortest paths, then the ISPL algorithms have a feasible solution immediately. Hence, the performance of algorithms will be biased.

In order to estimate the performance of our algorithms more accurately, we need more

work to make the initial shortest path set sufficiently different from the shortest paths used to calculate each  $z_k$ .

We now introduce three methods to generate random ISPL instances.

### 5.1.1 General Framework

Our random ISPL instances are all generated by the following steps: (1) generate a random network topology, (2) generate random costs, (3) generate the commodity origins and destinations, and (4) calculate the shortest path lengths for each commodity under the random costs. This method allow us to guarantee that we have a feasible instance of ISPL. The input parameters are number of nodes, edges, commodities, and maximum cost on edge. Let  $|N|$ ,  $|E|$ ,  $|K|$ , and  $C$  denote these parameters.

1. First construct a network with  $N$  nodes. Randomly generate each edge by choosing two endpoints from a uniform distribution. If the edge already exists, then choose endpoints again. Otherwise, add the edge to the network and repeat until there are  $E$  edges.
2. For each edge  $e$  in  $E$ , randomly generate the edge cost  $c_e$ .
3. Randomly generate the origin and destination of each commodity using the same procedure as for generating the endpoints of each edge. We can think of it as generating edges in the distance graph.
4. Calculate shortest path lengths for each commodity using a some-to-some (or all-to-all) shortest path algorithm.
5. Set the desired commodity shortest path lengths equal to the distances we found in step 4.

The generated network is not guaranteed to be connected. If there is more than one component in the network, then the effective size of the ISPL instance will be decreased. It may lead to bias in estimating the performance of algorithm. We avoid this problem by

first constructing a spanning tree when we generate the edges. After the spanning tree is constructed, we create the rest of the edges.

Each of our instance generation methods differs in step 2, the edge cost generation. In the following subsections, we suggest several ways of generating edge costs, and measure the ability of each to provide challenging test problems.

### 5.1.2 Random ISPL 1

The simplest method of choosing edge cost is to pick each independently using a uniform distribution in the interval  $[0, C]$ .

Since we use independent uniform distributions to generate the edge costs, the minimum cardinality path will have the smallest expected length for each commodity. Consider the expectation of the cost on edge  $e$  :

$$E(c_e) = \frac{0 + C}{2} = \frac{C}{2} \text{ for each edge } e.$$

Since the edge costs are independent, the expectation of length for path  $P_i$  is

$$E(l(P_i)) = E\left(\sum_{e \in P_i} c_e\right) = \sum_{e \in P_i} E(c_e) = \sum_{e \in P_i} \frac{C}{2} = |P_i| \frac{C}{2},$$

so the minimum cardinality path has the smallest expected length. This may lead to another bias in the algorithm performance since one of our initial shortest path selection methods is to start with the minimum cardinality path. In fact, as long as edge costs are independent of each other, the minimum cardinality path will always have the shortest expected length:

$$E(l(P_i)) = |P_i| E(c_e).$$

### 5.1.3 Random ISPL 2

The objective of this second random ISPL generating scheme is to reduce the probability that the minimum cardinality path has the smallest path length.

First, we note that it can be impossible to generate an instance where the shortest path of each commodity is different from its minimum cardinality path when the demand graph is a complete graph, i.e., there is commodity for every node pair.



**Lemma 5.1.** *If the demand graph is a complete graph, there is at least one commodity whose minimum cardinality path is also its shortest path.*

*Proof.* Consider an ISPL on a network  $G = (V, E)$  and a cost vector  $c > 0$ . Suppose the demand graph is a complete graph. Let  $c_{ij}$  be the smallest edge cost in  $c$  and let  $i$  and  $j$  be the two endpoints of this edge. Since the demand graph is a complete graph, there is a commodity between  $i$  and  $j$ . The minimum cardinality path for this commodity is just using this edge. Any other path for this commodity has to use more than one edge. Hence, the length of these paths other than  $(i, j)$  must be strictly greater than  $c_{ij}$ .  $\square$

Instead of using one uniform distribution  $[0, C]$  for each edge, we create three types of edges: short, median, and long. Let  $p_1$  and  $p_2$  represent the probabilities of an edge being a long edge or short edge ( $p_1 + p_2 < 1$ ). Let  $M$  be a large number. After generating a uniform random number  $c$  from an interval  $[0, C]$ , we generate another random number  $p$  in  $[0, 1]$  and set the edge costs to be the following:

$$\begin{aligned} c_{ij} &= Mc, & \text{if } 0 \leq p < p_1, \\ c_{ij} &= \frac{c}{M}, & \text{if } p_1 \leq p < p_1 + p_2, \\ c_{ij} &= c, & \text{otherwise.} \end{aligned} \tag{5.1}$$

Because the probability of each path being shortest is different under this scheme and using the minimum cardinality path, this method generates instances of ISPL that are more difficult for our algorithm to solve.

#### 5.1.4 Random ISPL 3

In this section, we propose a two-step generating scheme. Instead of using the minimum cardinality path to start our algorithm, we choose a different path based on the method we use to calculate the  $z_k$ .

In the stage of generating instance, we first generate the type of edges, short or long, and then generate the costs. We set the probability of long edge,  $p_1$ , (the probability of

short edge is  $1 - p_1$ ) for the stage of generating instance. Generate two independent random variable  $p$  in  $[0, 1]$  and  $c$  in  $[0, C]$ . Set the cost of this edge as following:

$$\begin{aligned} c_{ij} &= Mc, \quad \text{if } 0 \leq p < p_1, \\ c_{ij} &= \frac{c}{M}, \quad \text{otherwise.} \end{aligned} \tag{5.2}$$

In the stage of generating initial random paths, we only generate one random variable  $c$  in  $[0, C]$ . Whenever an edge is long when we generate the instance, we set it to be a short edge in the stage of choosing the initial shortest path set. If an edge is short, then we set it to be long in the later stage. Set the cost of edge according to (5.2). The type of edges between generating instance and initial paths is not independent anymore. Hence, we can guarantee the extreme edge types are different in the two stages. The results in Section 5.1.5 show that this generating scheme provides the most different shortest path sets.

#### 5.1.5 Random Instance Generation Results

In order to measure the performance of our algorithms, we have to generate random instances such that the initial shortest path sets we pick will be as different as possible from the shortest path set used for the  $z_k$ . We set the following parameters:  $C = 100$ ,  $M = 10$  and vary the random network structure by choosing different  $|N|$  and  $|E|$ .

We test 3 types of random networks which have 10 nodes and 10, 30, and 45 edges, respectively. There is a commodity on each node pair. For Random ISPL 1 and 2, we use parameters  $(C, M) = (100, 10)$  with different combinations of  $(p_1, p_2) \in \{0.05, 0.10, 0.15, \dots, 0.95\}^2$  and  $p_1 + p_2 \leq 1$ . We generate 100 random network topologies and assign the edge costs 50 times for each topology. Let I, II, III, and IV denote the following four types of generating methods:

- I: same cost on all edges,
- II: uniform random cost,
- III: random with probability( $p_1, p_2$ ),
- IV: random with probability( $p_2, p_1$ ).

**Table 2:** Percentage of similarity in shortest path set when  $(p_1, p_2) = (0.15, 0.7)$ 

	I	II	III	IV
I	100.00	36.04	23.86	43.97
II	36.24	14.87	10.04	17.88
III	23.44	10.02	6.75	11.76
IV	44.67	18.20	12.23	21.72

**Table 3:** Percentage of similarity in the shortest path set for switching edge type by  $p_1$ 

	$p_1$									
$( N ,  E ,  K )$	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
(30,50,435)	38.19	30.44	25.44	19.59	14.55	10.81	9.16	7.20	5.74	5.96
(30,100,435)	17.90	11.50	7.44	4.35	2.33	1.37	0.88	0.46	0.34	0.40
(30,200,435)	7.30	3.09	0.92	0.35	0.19	0.07	0.03	0.03	0.007	0.006
(30,300,435)	4.32	0.87	0.20	0.05	0.013	0.006	0.004	0.002	0.000	0.000
(30,435,435)	2.08	0.18	0.04	0.01	0.002	0.000	0.000	0.000	0.000	0.000

After testing the different values of  $p_1$  and  $p_2$ , we found that the lowest percentage of similarity in shortest path set occurs when  $(p_1, p_2) = (0.15, 0.7)$  (See Table 2). The similarity matrix is approximately symmetric and the lowest value usually falls into (III,III). Surprisingly, it did not fall into location (III,IV) or (IV,III) even though we switch the short and long edge probabilities.

We use a network with 30 nodes and different numbers of edges and commodities to test Random ISPL 3. The probability combinations  $(p_1, p_2)$  are set to be the following:

$$p_1 \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\},$$

$$p_2 = 1 - p_1.$$

We generate 100 network topologies and 50 instances for each topology, and compute the number of commodities which have same shortest path in the two cost cases. The average percent of commodities having the same shortest path in the two stages is shown in Table 3.

We find that switching types of edges provides us much more different shortest path sets. When the probability of  $p_1$  is 0.5, almost no commodities have the same shortest paths. Hence, we will choose Random ISPL 3 with probability (0.5, 0.5) as hard ISPL instances to test the performance of our algorithms (see Section 5.3).

**Table 4:** Similarity in SP path sets when  $(|N|,|E|,|K|,p1,p2)=(30,50,435,0.5,0.4)$ 

	I	II	III	IV
I	100.00	52.05	29.13	27.98
II	51.89	39.20	23.44	22.73
III	29.74	23.53	15.25	14.77
IV	28.37	22.82	14.85	14.38

**Table 5:** Similarity in SP path sets when  $(|N|,|E|,|K|,p1,p2)=(30,100,435,0.5,0.4)$ 

	I	II	III	IV
I	100.00	29.34	23.54	19.27
II	29.74	15.78	12.72	10.67
III	23.49	12.59	10.32	8.64
IV	19.17	10.56	8.72	7.32

Since we test hard instances using Random ISPL 3, it is not necessary to use Random ISPL 2 with probability  $(0.15, 0.7)$ . For comparison, we try another probability which does not provide such a different shortest path set. To test the relationship between the performance of our algorithms and the initial shortest path set, we choose Random ISPL 2 with probability  $(0.5, 0.4)$ .

The random networks we test have 30 nodes and 50, 100, 200, 300 and 435 edges. When testing similarity, we assume all node pairs are commodities. We generate 100 network topologies and assign edge costs 50 times as before. Table 4 through Table 8 show that the lowest similarity does not always fall into (III, III). When the graphs become dense, we have much lower similarity in the shortest paths. Notice that, when the underlying network is a complete graph, the uniform random instance also provides very different shortest path sets.

From Table 3, we know we can test our algorithms on problems where we purposely begin with an initial shortest path set that has 0% similarity compared to a feasible shortest

**Table 6:** Similarity in SP path sets when  $(|N|,|E|,|K|,p1,p2)=(30,200,435,0.5,0.4)$ 

	I	II	III	IV
I	100.00	18.11	21.47	18.97
II	18.08	6.39	7.46	6.67
III	21.51	7.59	8.78	7.91
IV	19.06	6.70	7.78	7.01

**Table 7:** Similarity in SP path sets when  $(|N|, |E|, |K|, p_1, p_2) = (30, 300, 435, 0.5, 0.4)$ 

	I	II	III	IV
I	100.00	16.27	21.58	19.92
II	16.32	4.14	5.46	5.01
III	21.62	5.42	7.12	6.63
IV	19.66	4.92	6.53	6.05

**Table 8:** Similarity in SP path sets when  $(|N|, |E|, |K|, p_1, p_2) = (30, 435, 435, 0.5, 0.4)$ 

	I	II	III	IV
I	100.00	16.01	23.07	21.37
II	15.83	2.85	4.06	3.77
III	23.02	4.10	5.92	5.49
IV	21.49	3.84	5.54	5.09

path set. However, we also would like to test the algorithms using an intermediate quality starting point. Hence, we will also use this random generating scheme  $((p_1, p_2) = (0.5, 0.4))$  to test the performance of our algorithms.

## 5.2 Sample Algorithm Performance

We show some examples to illustrate the progress of our ISPL algorithms. We generate 100 random ISPL instances and solve them by 6 algorithms. Each instance has 100 nodes, 3000 edges, and 4500 commodities and generated by using the Random ISPL 2 scheme with probability  $(0.5, 0.4)$ .

We plot three examples of the solutions of minimizing infeasibility at each iteration (see Figure 43, Figure 44, and Figure 45). We see that all 6 algorithms terminate within 30 iterations. Those algorithms (1, 3, and 5) that change the shortest path set at the end of each iteration terminate in 15 iterations. It seems that changing the shortest path set at each iteration can decrease the number of iterations required for algorithms to converge. We will discuss more detailed results in Section 5.3.

In these 100 instances, the average difference between ISPL solutions and optimal solutions is approximately 3%. More detailed computational results are discussed in the following section.

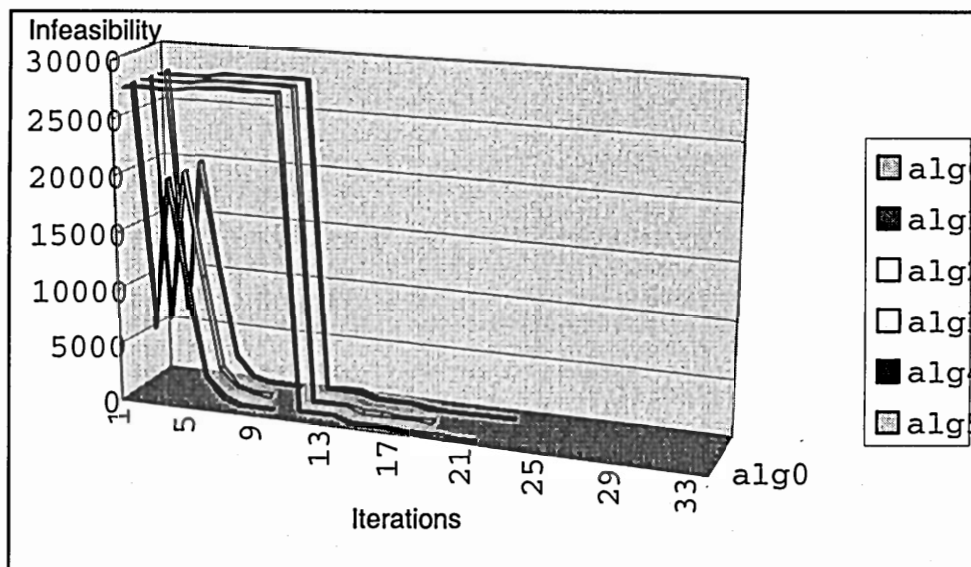


Figure 43: ISPL example 1

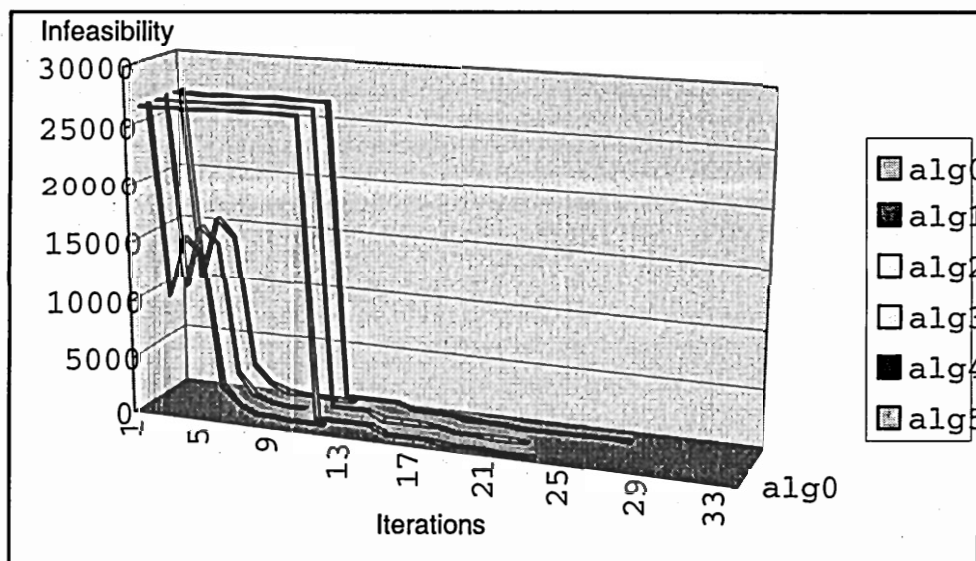


Figure 44: ISPL example 2

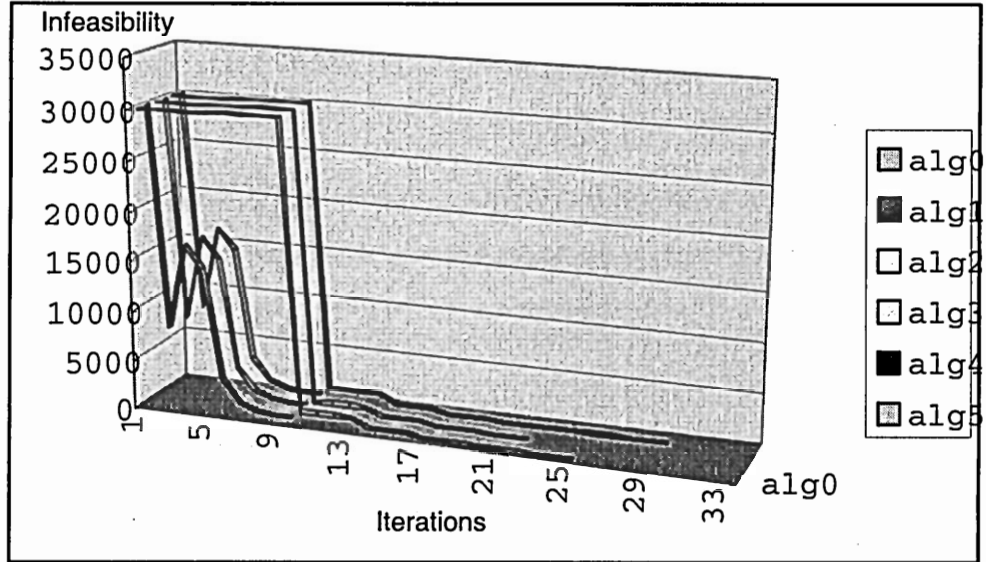


Figure 45: ISPL example 3

### 5.3 Performance of Algorithms on Random Instances

#### 5.3.1 Intermediate-difficulty Instances

For tests on intermediate-difficulty instances, we use a 100-node network. In order to test algorithms on both sparse and dense networks, we choose the number of edges to be equal to 150, 200, 300, 1650, and 3300. We do not choose the complete graph since we can get a feasible solution in polynomial time by Theorem 3.2. The number of commodities is equal to 1650, 3300, and 4950. Thus, there are 15 combinations of parameters. We generate 30 ISPL instances for each combination by using  $(p_1, p_2) = (0.5, 0.4)$  and solved them by each algorithm. There are two kinds of initial shortest path sets for starting our algorithms: the minimum cardinality paths and random paths using  $(p_1, p_2) = (0.5, 0.4)$ . All the tests are run on a UNIX platform and all subproblems which solve linear programs are solved by the LP subroutine of CPLEX 8.0[22].

##### 5.3.1.1 Performance of Algorithms

As we mentioned in Section 4.5, we measure the performance by calculating the percentage difference from the optimal solution (4.29). We evaluate the average performance of algorithms on the instances that are generated by Random ISPL 1 and Random ISPL 2 with

**Table 9:** Average performance for intermediate-difficulty ISPL instances (%)

Initial path type	Alg. 0	Alg. 1	Alg.2	Alg. 3	Alg. 4	Alg.5
minimum cardinality path	2.26	2.09	2.26	2.10	2.17	2.00
Random path	4.43	3.06	4.34	3.02	4.41	2.97

**Table 10:** Standard deviation of the performance for intermediate-difficulty ISPL (%)

Initial path type	Alg. 0	Alg. 1	Alg.2	Alg. 3	Alg. 4	Alg.5
minimum cardinality path	4.55	4.56	4.52	4.54	4.45	4.46
Random path	10.64	10.29	10.64	10.25	10.66	10.11

probability (0.5, 0.4), and solved by choosing the minimum cardinality paths and uniform random paths as initial shortest path set, respectively. Table 9 shows the mean of percentage of difference and Table 10 shows the corresponding standard deviation for 6 algorithms and two types of initial shortest path sets.

We can calculate the one-side  $t$ -interval [19] as following:

$$\mu \leq \bar{x} + \frac{t_{\alpha, n-1s}}{\sqrt{n}}.$$

Thus, we know that the performance of 6 algorithms by two types of initial shortest path set will be as shown in Table 11, Figure 46, and Figure 47.

From Table 11, we have the following observations:

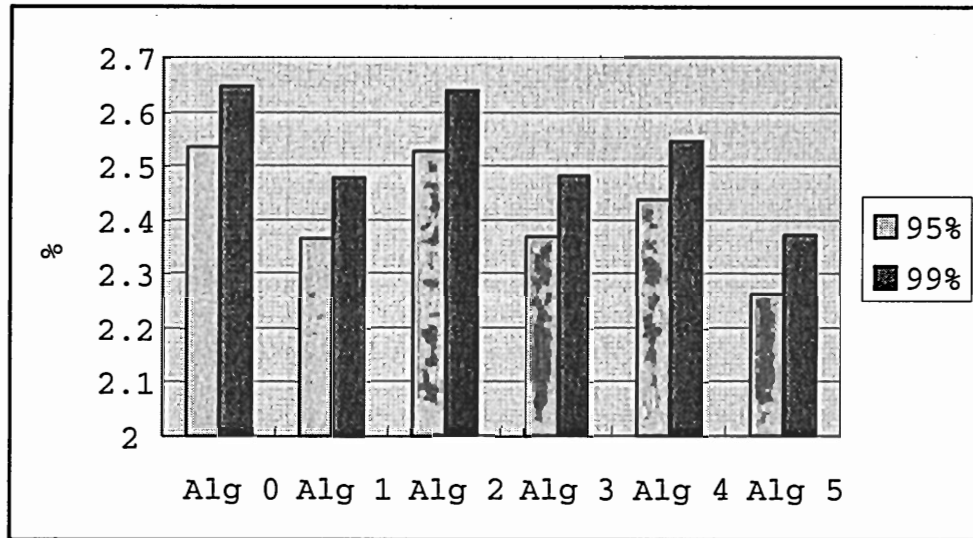
1. On average, 99% of instances can be solved to within 2.7% of optimality using intelligent starting paths, and to within 5.5% of optimality using random paths.
2. Changing the shortest path set at each iteration does help the performance of the algorithms because Algorithms 1, 3, and 5 are better than Algorithms 0, 2, and 4, especially when starting with random paths.

We compute the success probability  $p$  such that ISPL can be solved within 3% and 5%

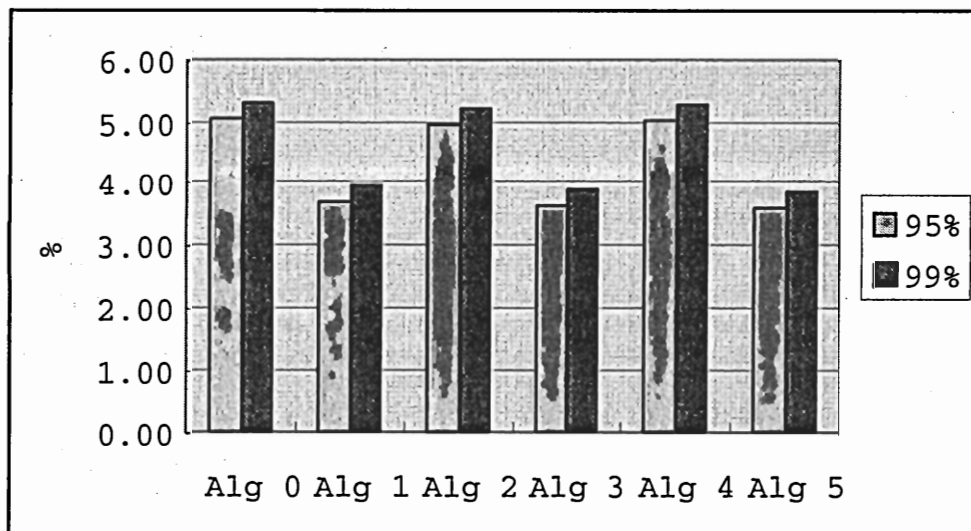
**Table 11:** Average performance for algorithms under 95 and 99 percent certainly (%)

	Initial path type	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
95%	minimum cardinality path	2.53	2.36	2.53	2.37	2.44	2.26
	Random path	5.06	3.68	4.98	3.63	5.04	3.58
99%	minimum cardinality path	2.65	2.48	2.64	2.48	2.55	2.37
	Random path	5.33	3.93	5.24	3.88	5.30	3.83





**Figure 46:** Performance of algorithms starting at the minimum cardinality paths



**Figure 47:** Performance of algorithms starting at uniform random paths

**Table 12:** Lower bound of probability of solving intermediate-difficult ISPL within 3 and 5 percent

	Initial path type	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
3%	minimum cardinality path	0.75	0.77	0.75	0.77	0.75	0.78
	Random path	0.63	0.79	0.64	0.80	0.63	0.79
5%	minimum cardinality path	0.82	0.82	0.82	0.82	0.82	0.83
	Random path	0.69	0.86	0.71	0.87	0.69	0.87

difference. According to the binomial distribution, we know that

$$p \in \left( \hat{p} - z_\alpha \sqrt{\frac{\hat{p}(1 - \hat{p})}{n}}, 1 \right).$$

Table 12 shows the lower bound of the 99% confidence interval of the success probability of all six algorithms. There is at least 0.75 probability that the intermediate-difficulty ISPL can be solved within 3% of optimality, and 0.82 probability to be solved within 5% of optimality when starting with the minimum cardinality paths. For Algorithms 1, 3, and 5, starting with random paths has higher probability to solve ISPL within 3% and 5% than starting with the minimum cardinality path. For Algorithms 0, 2, and 4, starting with the minimum cardinality path has higher probability than starting with random paths.

We perform another statistical test to answer the question: “Is there any algorithm dominated by any other?” Let  $\mu = x_i - x_j$  where  $x_i$  and  $x_j$  are means of algorithm  $i$  and  $j$ . The hypothesis is

$$H_0 : \mu \geq 0 \text{ versus } H_A : \mu < 0.$$

If we can reject the hypothesis  $H_0$ , we know that Algorithm  $i$  is better than Algorithm  $j$ . Table 13 is the corresponding  $t$ -value for testing the average performance of algorithm  $i$  and  $j$  using the minimum cardinality path as the initial shortest path. The  $p$ -value is calculated by

$$P(X \leq t).$$

If the  $p$ -value is less than 0.01, then we have enough evidence to reject the null hypothesis with 99% confidence. We find that Algorithms 1, 3, 4, 5 do get better solution than Algorithms 0, 2, and Algorithm 5 is better than Algorithms 1, 4 when the initial shortest path set chosen by minimum cardinality paths. Algorithms 3 and 5 have the best performance when starting with the minimum cardinality paths.

**Table 13:** t-value comparing the performance starting with the minimum cardinality path

	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
Alg. 0	–	2.98	-0.11	2.33	8.25	3.95
Alg. 1	–	–	-3.15	-0.22	-1.91	3.01
Alg. 2	–	–	–	2.54	2.46	4.17
Alg. 3	–	–	–	–	-1.44	1.46
Alg. 4	–	–	–	–	–	2.93
Alg. 5	–	–	–	–	–	–

**Table 14:** t-value comparing the performance starting with random paths

	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
Alg. 0	–	7.66	1.01	7.17	1.04	7.80
Alg. 1	–	–	-6.88	0.62	-7.59	1.63
Alg. 2	–	–	–	6.75	-0.83	7.01
Alg. 3	–	–	–	–	-7.11	-0.32
Alg. 4	–	–	–	–	–	7.73
Alg. 5	–	–	–	–	–	–

We do the same statistical hypothesis testing between algorithms when the initial shortest path set is chosen by uniform random shortest paths. Table 14 shows that corresponding  $t$ -values. We find that Algorithms 1, 3, 5 do get better solution than Algorithms 0, 2, 4 when we choose initial shortest path uniform randomly.

We also test which method of choosing the initial shortest path provides better performance. Table 15 shows that corresponding  $t$ -value. There is enough evidence to say that the minimum cardinality path will lead to a better solution than uniform random paths.

#### 5.3.1.2 Speed of Convergence of Algorithms

We now discuss the computation time of all algorithms. We used several UNIX machines simultaneously: two are Sun 280Rs, each with 2 900MHz UltraSparc-III CPUs and 2GB RAM, and the other is a Sun 220R, with 2 360MHz UltraSparc-II CPUs and 2GB RAM. Because these machines form a computational cluster and many people's jobs share the resources, it is difficult to compare the actual computation time of each algorithm on the

**Table 15:** t-value comparing two initial paths for all algorithms

	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
$t$ -value	-5.18	-2.39	-4.98	-2.26	-5.35	-2.44

**Table 16:** Average number of iterations for intermediate-difficulty ISPL

Initial path type	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
minimum cardinality path	20.53	9.26	21.09	9.20	29.46	12.49
Random path	27.92	10.96	30.78	10.57	30.40	12.67

**Table 17:** Standard deviation of the number of iterations for intermediate-difficulty ISPL

	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
minimum cardinality path	23.91	8.88	28.14	11.75	41.06	11.91
Random path	29.60	8.81	38.70	11.47	32.90	9.55

same machine. Hence, we compare the average iterations of each algorithm, instead of the computation time.

Table 16 and Table 17 are the mean and standard deviation of total iterations of the 6 algorithms. We find that 95% and 99% of instances will reach local optimal solutions in less than 35 iterations (see Table 18). Moreover, the speed of convergence of Algorithms 1, 3, and 5 are much better than Algorithms 0, 2, and 4 no matter what initializing method we use. We know that changing the shortest path set at each iteration does improve the speed of convergence of algorithms. For Algorithms 1, 2, 3, and 4, using minimum cardinality paths yields better speed than using uniform random paths (see Table 19).

### 5.3.1.3 Performance Summary of Combinations of Generating Schemes

In this section, we summarize the results to show how our algorithms work in different scenarios: generating scheme, initializing scheme, number of edges, and number of commodities. For each scenario, we generate 30 ISPL instances to test the performance of our algorithms.

First, consider ISPL instances that are generated by putting same cost on each edge and

**Table 18:** Average number of iterations to converge for intermediate-difficulty ISPL

		Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
95%	minimum cardinality path	21.95	9.79	22.76	9.90	31.90	13.20
	Random path	29.68	11.49	33.08	11.25	32.36	13.24
99%	minimum cardinality path	22.54	10.01	23.45	10.19	32.92	13.50
	Random path	30.41	11.71	34.04	11.54	33.18	13.48

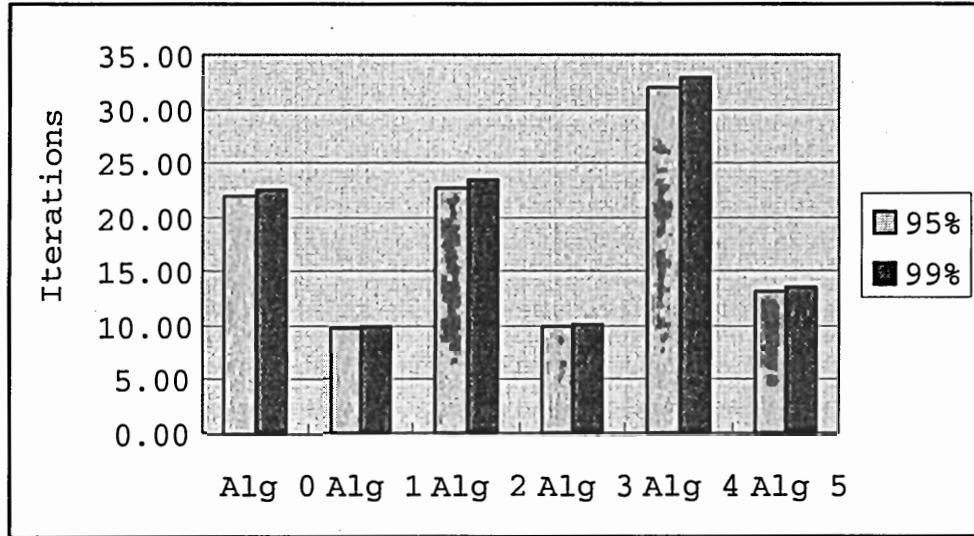


Figure 48: Iterations for convergence when starting with minimum cardinality paths

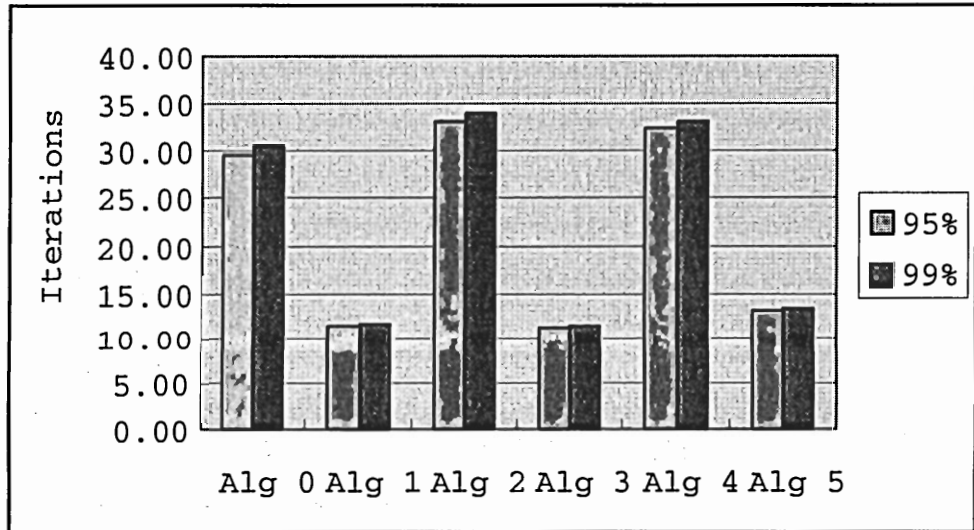


Figure 49: Iterations for convergence when starting with uniform random paths

Table 19: t-value comparing the speed of convergence between different initial paths

	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
t-value	-5.37	-3.77	-5.60	-2.30	-0.49	-0.33

**Table 20:** Algorithm performance: (ISPL instance I, the minimum cardinality paths)

Edge	Commodity	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
150	1650	0	0	0	0	0	0
	3300	0	0	0	0	0	0
	4950	0	0	0	0	0	0
200	1650	0	0	0	0	0	0
	3300	0	0	0	0	0	0
	4950	0	0	0	0	0	0
300	1650	0	0	0	0	0	0
	3300	0	0	0	0	0	0
	4950	0	0	0	0	0	0
1650	1650	0	0	0	0	0	0
	3300	0	0	0	0	0	0
	4950	0	0	0	0	0	0
3300	1650	0	0	0	0	0	0
	3300	0	0	0	0	0	0
	4950	0	0	0	0	0	0

solved by initializing with minimum cardinality paths and uniform random paths, respectively. Table 20 and Table 21 are the summaries of the results. Obviously, all algorithms will be terminated at an optimal solution in Table 20 since the minimum cardinality path is the real shortest path for each commodity. In Table 21, we find the average performance of algorithms is under 10% when starting with uniform random paths. Moreover, Algorithms 1, 3, and 5 perform better than Algorithms 0, 2, and 4 except for one instance.

Next we consider ISPL instances that are generated by Random ISPL 1 (uniform random cost distribution on each edge). We summarize the results of initializing by minimum cardinality paths and uniform random pats in Table 22 and Table 23. Comparing these two tables, we find that starting with minimum cardinality paths performs better than random paths when the network is sparse (number of edges equal to 150, 200, and 300). The difference between of our solution and the optimal solution is within 1.5% of optimality. We also see that ISPL is polynomially solvable when the distance graph is complete in Table 22.

Table 24 and Table 25 are the results when ISPL instances are generated by Random ISPL 2 with probability (0.5,0.4). Other than the case when the distance graph is complete, it seems that starting with minimum cardinality paths cannot guarantee to reach a better

**Table 21: Algorithm performance: (ISPL instance I, uniform random paths)**

Edge	Commodity	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
150	1650	0.25	0.02	0.34	0.21	0.25	0.02
	3300	0.26	0.19	0.44	0.32	0.26	0.16
	4950	0.45	0.45	0.45	0.45	0.45	0.45
200	1650	1.05	0.05	0.96	0.45	1.05	0.05
	3300	1.31	0.10	1.35	1.16	1.31	0.10
	4950	2.93	2.93	2.92	2.73	2.93	2.93
300	1650	8.46	3.35	8.18	3.98	8.46	2.37
	3300	7.46	6.30	7.16	6.68	7.46	6.30
	4950	8.59	8.59	8.59	8.46	8.59	8.59
1650	1650	3.69	1.45	3.82	1.41	3.69	1.40
	3300	2.29	0.79	2.38	0.64	2.29	0.73
	4950	6.75	7.76	6.35	7.33	6.35	7.33
3300	1650	8.82	0.03	11.33	0.59	9.32	0.03
	3300	2.01	0.51	4.70	0.65	2.00	0.47
	4950	0.19	0.18	0.18	0.18	0.18	0.18

**Table 22: Algorithm performance: (ISPL instance II, the minimum cardinality paths)**

Edge	Commodity	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
150	1650	0.06	0.08	0.10	0.10	0.06	0.08
	3300	0.00	0.00	0.00	0.00	0.00	0.00
	4950	0.00	0.00	0.00	0.00	0.00	0.00
200	1650	0.21	0.18	0.36	0.31	0.21	0.17
	3300	0.01	0.01	0.02	0.02	0.01	0.01
	4950	0.00	0.00	0.00	0.00	0.00	0.00
300	1650	0.81	1.36	1.14	1.23	0.80	1.05
	3300	0.23	0.23	0.29	0.27	0.23	0.23
	4950	0.00	0.00	0.00	0.00	0.00	0.00
1650	1650	6.43	5.75	5.89	5.25	6.26	5.78
	3300	7.12	8.09	7.33	8.76	6.48	7.48
	4950	0.00	0.00	0.00	0.00	0.00	0.00
3300	1650	2.30	1.22	2.34	0.55	2.00	1.27
	3300	3.89	4.19	3.63	4.28	3.84	4.14
	4950	0.00	0.00	0.00	0.00	0.00	0.00

**Table 23:** Algorithm performance: (ISPL instance II, uniform random paths)

Edge	Commodity	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
150	1650	0.65	0.59	1.06	0.60	0.65	0.28
	3300	0.99	0.88	1.13	1.04	0.99	0.85
	4950	1.45	1.45	1.45	1.43	1.45	1.45
200	1650	2.58	2.83	2.32	2.89	2.58	2.81
	3300	3.16	3.40	2.97	4.94	2.86	3.22
	4950	2.98	2.98	2.98	2.96	2.82	2.82
300	1650	8.19	5.80	9.32	3.09	8.19	5.78
	3300	7.94	5.07	7.99	5.95	7.94	5.07
	4950	8.79	8.79	8.79	8.79	8.79	8.79
1650	1650	7.68	2.99	5.20	1.90	7.50	3.01
	3300	2.57	2.13	2.60	2.18	2.45	2.12
	4950	0.00	0.00	0.00	0.00	0.00	0.00
3300	1650	5.56	0.44	5.24	0.14	5.66	0.46
	3300	0.50	0.19	0.43	0.18	0.64	0.17
	4950	0.00	0.00	0.00	0.00	0.00	0.00

solution than starting with random paths, and vice versa.

In general, the algorithms perform well when the physical network is sparse, as most real-life networks are. We also note that we get better solutions when there are many or few commodities than for intermediate sizes of  $|K|$ .

### 5.3.2 Hard Instances

#### 5.3.2.1 Solution Performance of Algorithms

We now discuss the performance of our algorithms when solving hard ISPL instances. For the hard ISPL instances, we choose the initial shortest path set by the method introduced in Section 5.1.4. Hence, we have the most different initial shortest path set from the shortest paths used to decide  $z_k$ . For each number of edges and commodities (15 combinations), we generate 60 ISPL instances. The mean and standard deviation of the hard ISPL performance are as shown in Table 26. We find that our algorithms can provide us a solution whose difference from the optimal solution is approximately 6% in average. Assume that the performance is normally distributed, then we know that 99% of ISPL instances can be solved under 7% difference (see Table 27 and Figure 50).

Table 28 is the lower bound on the success probability for algorithms to solve hard ISPL instances within 3% and 5%. Algorithms 1, 3, and 5 perform much better than Algorithms



**Table 24: Algorithm performance: (ISPL instance III, the minimum cardinality paths)**

Edge	Commodity	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
150	1650	1.28	1.25	1.29	1.31	1.27	1.19
	3300	0.19	0.17	0.23	0.23	0.19	0.16
	4950	0.00	0.00	0.00	0.00	0.00	0.00
200	1650	1.82	1.55	1.89	1.84	1.78	1.55
	3300	0.76	0.69	0.73	0.72	0.73	0.68
	4950	0.00	0.00	0.00	0.00	0.00	0.00
300	1650	3.12	2.49	3.39	2.84	3.10	1.91
	3300	0.26	0.19	0.26	0.28	0.26	0.19
	4950	0.00	0.00	0.00	0.00	0.00	0.00
1650	1650	9.16	7.04	8.76	7.03	8.82	6.61
	3300	10.71	11.27	10.96	11.46	10.66	11.26
	4950	0.00	0.00	0.00	0.00	0.00	0.00
3300	1650	3.93	2.26	3.81	1.82	3.54	2.20
	3300	8.04	8.45	7.90	8.52	7.92	8.35
	4950	0.00	0.00	0.00	0.00	0.00	0.00

**Table 25: Algorithm performance: (ISPL instance III, uniform random paths)**

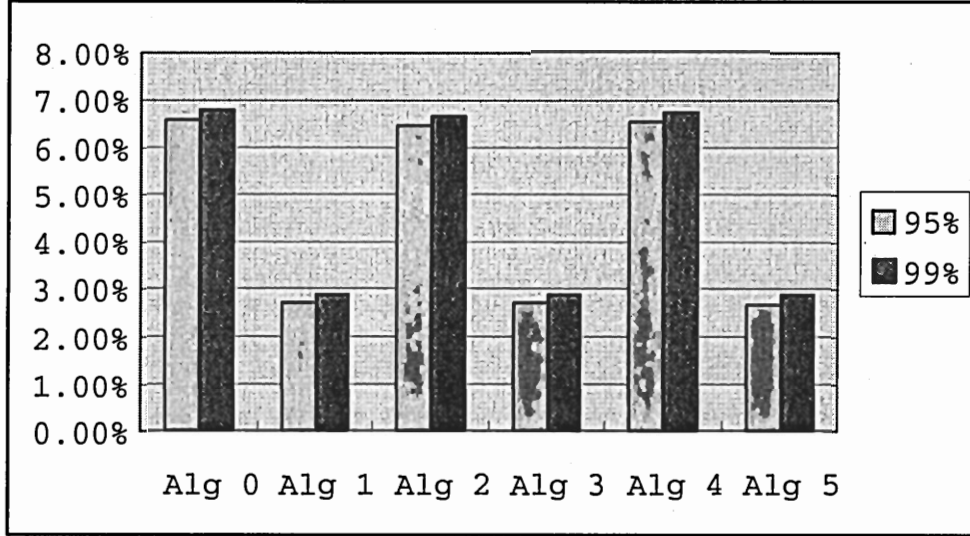
Edge	Commodity	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
150	1650	1.40	1.94	1.53	1.87	1.40	1.91
	3300	1.25	1.09	1.28	1.16	1.25	1.09
	4950	0.70	0.70	0.70	0.70	0.70	0.70
200	1650	3.91	3.11	4.02	3.07	3.91	3.10
	3300	2.96	3.09	3.08	3.22	2.96	3.07
	4950	0.40	0.40	0.40	0.40	0.43	0.40
300	1650	3.69	6.02	3.86	6.33	3.69	6.01
	3300	1.68	1.10	1.90	1.22	1.68	1.10
	4950	14.47	14.47	15.20	16.00	16.00	13.79
1650	1650	10.12	4.84	9.51	4.98	10.02	4.88
	3300	12.60	3.30	12.06	6.49	12.78	6.27
	4950	9.18	9.18	9.18	9.18	9.18	9.18
3300	1650	5.70	0.73	5.83	0.71	5.62	0.77
	3300	7.39	2.03	7.05	1.79	7.31	1.95
	4950	0.00	0.00	0.00	0.00	0.00	0.00

**Table 26: Mean and standard deviation of the performance for hard ISPL (%)**

	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
Mean	6.05	2.24	5.95	2.25	6.00	2.22
Std.	9.25	8.05	9.18	8.06	9.24	8.05

**Table 27: The performance of algorithms for hard ISPL (%)**

	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
95%	6.57	2.70	6.46	2.71	6.52	2.68
99%	6.79	2.88	6.68	2.90	6.74	2.87



**Figure 50:** Hard ISPL performance of algorithms

**Table 28:** Lower bound of probability of solving hard ISPL within 3 and 5 percent

	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
3%	0.46	0.82	0.46	0.82	0.46	0.82
5%	0.53	0.88	0.55	0.89	0.54	0.88

0, 2, and 4. There are at least 82% and 88% probability for Algorithms 1, 3, and 5 to solve hard ISPL instances within 3% and 5%.

We also perform the statistical test for the comparisons between algorithms. Table 29 shows the corresponding  $t$ -value. We can conclude that Algorithms 0 and 2 have the worst performance of the six algorithms. Algorithm 1 is better than Algorithm 4. Algorithms 3 and 5 have the best performance on the hard ISPL instances. This conclusion is similar to the intermediate-difficulty ISPL instance analysis. Changing the shortest path set in each iteration does help the performance of algorithms.

We also want to know the worst performance in each edge and commodity combination (see Table 30). We find that the algorithms sometimes can perform poorly when both the number of edges and the number of commodities are large (especially when distance graph is complete). We know that ISPL is polynomially solvable when the distance graph is complete (by Lemma 3.10). For these instances, we can solve them using the method in Lemma 3.10. Another interesting finding is that Algorithms 1, 3, and 5 work much worse

**Table 29:** t-value comparing the performance of algorithms for hard ISPL

	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
Alg. 0	–	20.58	1.61	20.44	5.93	20.63
Alg. 1	–	–	-20.37	-0.43	-20.38	2.70
Alg. 2	–	–	–	20.23	-0.91	20.43
Alg. 3	–	–	–	–	-20.24	1.02
Alg. 4	–	–	–	–	–	20.43
Alg. 5	–	–	–	–	–	–

**Table 30:** Algorithm performance for hard ISPL in different scenario

Edge	Commodity	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
150	1650	17.45	44.27	24.67	44.27	24.67	44.27
	3300	17.08	13.00	17.08	12.93	17.08	13.00
	4950	0.60	0.60	0.60	0.60	0.60	0.60
200	1650	37.24	6.48	25.97	6.52	37.24	6.48
	3300	21.06	7.75	21.12	7.75	21.06	7.75
	4950	8.73	7.64	8.73	8.73	8.73	7.64
300	1650	37.25	19.11	39.88	13.91	37.25	19.11
	3300	19.20	7.14	19.20	4.74	19.20	7.14
	4950	18.23	18.23	18.23	18.23	18.23	18.23
1650	1650	14.13	11.68	14.03	10.63	13.89	11.20
	3300	30.61	32.14	30.42	32.70	30.60	32.12
	4950	92.59	92.59	92.59	92.59	92.59	92.59
3300	1650	7.41	3.92	6.82	4.02	7.40	3.92
	3300	12.61	10.79	12.37	10.61	12.56	10.77
	4950	38.06	38.06	38.06	38.06	38.06	38.06

than Algorithms 0, 2, and 4 in the smallest instance.

#### 5.3.2.2 Speed of Convergence of Algorithms

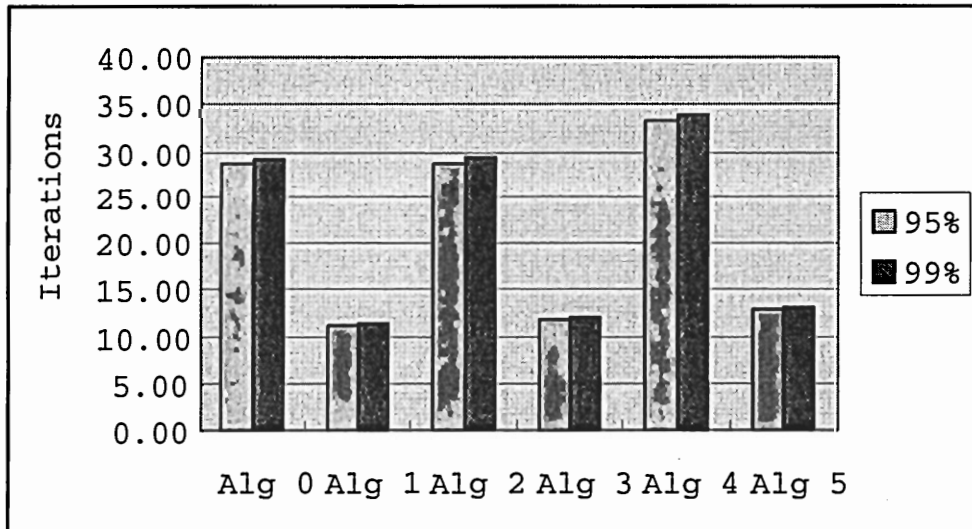
We discuss the speed of convergence of the algorithms. Table 31 shows the mean and standard deviation of iterations used to get a local optimal solution. Assuming a normal distribution, we know 95% and 99% of ISPL instances will reach a local optimal solution under 34 iterations (see Table 32 and Figure 51). We find that Algorithms 1, 3, and 5 have better speeds of convergence than Algorithms 0, 2, and 4. Comparing to the results in Table 18, we find that there is not much difference between starting with a uniform random path and starting with the most different shortest path for each commodity. The speed of convergence of algorithms that starting with these two different initial shortest path set are almost the same.

**Table 31:** Mean and standard deviation of iterations to converge for hard ISPL

	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
Mean	27.41	10.72	27.36	11.11	31.49	12.24
Std.	23.74	8.56	25.23	11.49	30.81	11.26

**Table 32:** Average iterations for convergence for hard ISPL under 95 and 99 percent

	Alg. 0	Alg. 1	Alg. 2	Alg. 3	Alg. 4	Alg. 5
95%	28.75	11.21	28.78	11.76	33.23	12.88
99%	29.31	11.41	29.37	12.03	33.95	13.14



**Figure 51:** Iterations for convergence in the worst case

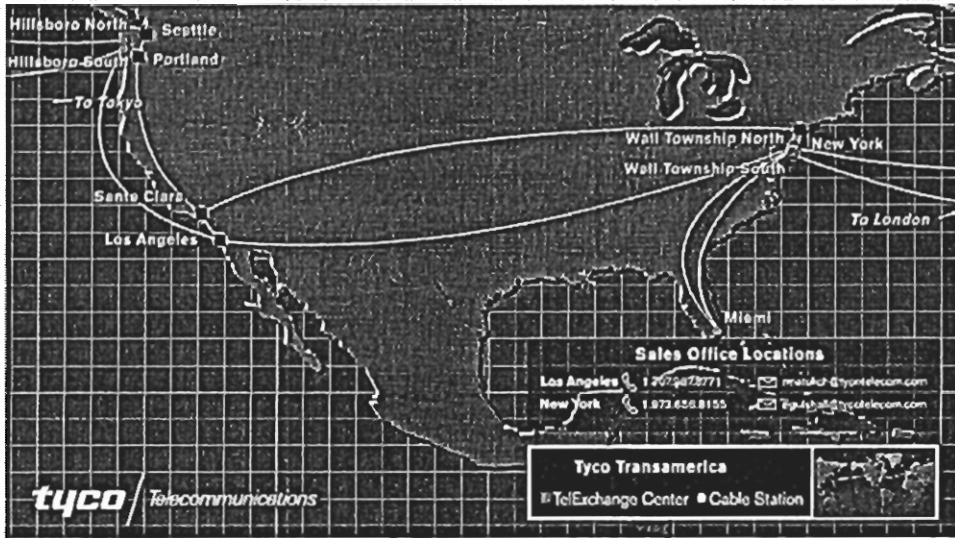


Figure 52: Tyco network

#### 5.4 Telecommunication Example

In this section, we test our algorithms on telecommunication ISPL instances from the real world. We use the networks of three major bandwidth providers, Tyco [34], Level 3 [33], and Global Crossing [32]. These companies' exact prices for bandwidth were unavailable, so we used the prices posted on bandwidthmarket.com [31], an on-line bandwidth exchange. For each network, we create three instances, corresponding to the first, second, and third lowest advertised prices on September 17, 2003. We solve these nine ISPL instances with all of our algorithms.

The underlying networks of Tyco, Level 3, and Global Crossing are shown in Figures 52, 53, and 54. The Tyco ISPL instances have parameters  $(|N|, |E|, |K|)$  equal to  $(8,9,12)$ ,  $(8,9,9)$  and  $(8,9,9)$ . The parameters of the Level 3 instances are  $(66,77,716)$ ,  $(66,77,577)$ , and  $(66,77,543)$ . Global Crossing has instances with parameters  $(103,117,1264)$ ,  $(103,117,820)$ , and  $(103,117,739)$ . Notice that Tyco has the smallest number of nodes and edges and commodities, and Global Crossing has the largest. Hence, we can test the performance of our algorithms on different sizes of telecom ISPL instances.

We also choose two starting points: minimum cardinality paths and random paths. Table 33, Table 34, and Table 35 are the summaries of the results. The performance of all

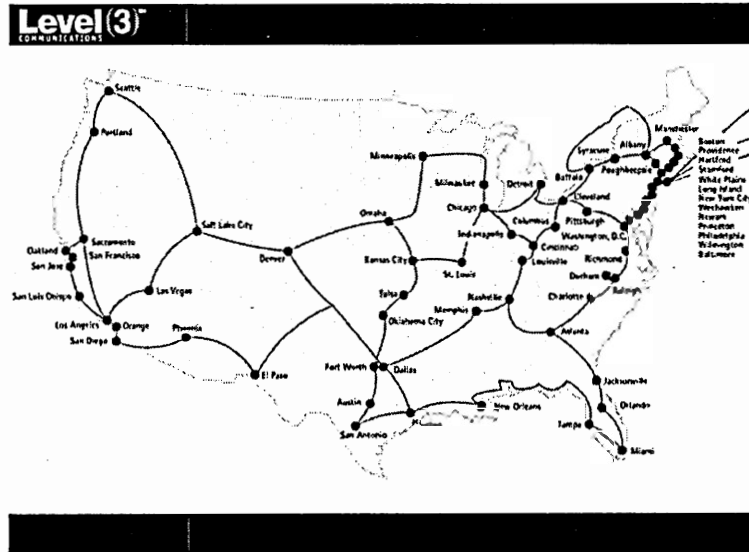


Figure 53: Level 3 network

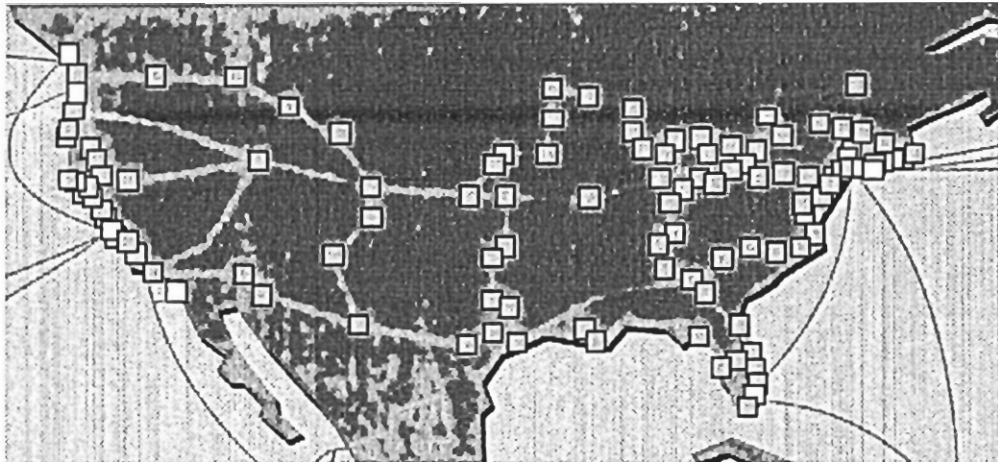


Figure 54: Global Crossing network

**Table 33: Algorithm performance for Tyco (%)**

Algorithm	Path type	Tyco		
		Instance 1	Instance 2	Instance 3
Alg. 0	0	48.27	62.39	66.85
	1	48.27	62.39	66.85
Alg. 1	0	48.27	62.39	66.85
	1	48.27	62.39	66.85
Alg. 2	0	48.27	62.39	66.85
	1	48.27	62.39	66.85
Alg. 3	0	48.27	62.39	66.85
	1	48.27	62.39	66.85
Alg. 4	0	48.27	62.39	66.85
	1	48.27	62.39	66.85
Alg. 5	0	48.27	62.39	66.85
	1	48.27	62.39	66.85

**Table 34: Algorithm performance for Level 3 (%)**

Algorithm	Path type	Level 3		
		Instance 1	Instance 2	Instance 3
Alg. 0	0	208.92	277.44	268.75
	1	207.14	277.42	268.75
Alg. 1	0	208.92	277.44	268.75
	1	207.14	277.42	268.75
Alg. 2	0	208.92	277.44	268.75
	1	207.14	277.42	268.75
Alg. 3	0	208.92	274.87	270.73
	1	207.14	277.13	272.01
Alg. 4	0	208.92	277.44	268.75
	1	207.14	277.42	268.75
Alg. 5	0	208.92	277.44	268.75
	1	207.14	277.42	268.75

algorithms are much worse than the results when we test on the random ISPL instances in Section 5.3. The reason is that these telecom ISPL instances are infeasible.

There are two types of infeasibility in these telecom ISPL instances. The first is that the length of the shortest path for some commodity might have to be greater than the desired length. The second is that distance graph might violate the triangular inequality, i.e., arbitrage exists.

We can compute the lower bound of the infeasibility of Type 1 in the following way: Set the costs of each edge  $(i, j)$  of the underlying network either equal to  $z_k$  if there is a

**Table 35:** Algorithm performance for Global Crossing (%)

Algorithm	Path type	Global Crossing		
		Instance 1	Instance 2	Instance 3
Alg. 0	0	259.30	266.67	254.76
	1	259.44	270.28	254.28
Alg. 1	0	259.30	266.67	254.72
	1	259.44	270.32	253.65
Alg. 2	0	259.30	266.67	253.11
	1	259.45	270.22	254.28
Alg. 3	0	259.30	268.47	253.66
	1	259.45	270.00	253.46
Alg. 4	0	259.30	266.67	253.11
	1	259.44	270.22	254.28
Alg. 5	0	259.30	266.67	253.07
	1	259.44	270.00	253.43

commodity  $k$  between node  $i$  and  $j$ , or equal to 0 if not, and then solve the shortest path problem for all commodities. If the length of the shortest path for commodity  $k$ , say  $z'_k$ , is greater than  $z_k$ , then we know the ISPL must be infeasible for this commodity and the infeasibility is at least as large as  $z'_k - z_k$ .

To check the infeasibility of Type 2, we set the cost of edges in the distance graph equal to  $z_k$  and solve the shortest path problem for all commodities on the distance graph. If the shortest path length for some commodity  $k$ , say  $\bar{z}_k$ , is less than  $z_k$ , then the instance must be infeasible and there is arbitrage in this instance. The infeasibility for this commodity must be at least  $z_k - \bar{z}_k$ .

Table 36 shows the lower bound of these two types of infeasibility in all telecom ISPL instances. We find that Type 1 infeasibility exists for all telecom ISPL instances. Infeasibility of Type 2 is much less common. This makes sense, since Type 2 infeasibility results in opportunities for arbitrage.

In order to evaluate the performance more accurately, we solve the Tyco ISPL instances to optimality by solving mixed-integer programming problems (5.3)-(5.7). The instances for Level 3 and Global Crossing are too large for this IP to be solved. The objective function in (5.3) is to minimize the total infeasibility of all commodities. Let  $\delta_k^*$  denote the optimal solution of (5.3)-(5.7). Each  $y_{kp}$  is an indicator variable to show whether path  $P$  is the



**Table 36:** Two infeasibilities of Telecom ISPL instances

Company	Instance	Infeasibility 1 (%)	Infeasibility 2 (%)
Tyco	1	8.43	0.00
	2	13.93	0.00
	3	15.67	0.00
Level 3	1	102.66	12.65
	2	89.86	2.44
	3	46.93	0.00
Global Crossing	1	201.28	23.30
	2	46.05	4.54
	3	21.97	2.36

shortest path for commodity  $k$ . Each  $s_{kp}$  represents the difference between the length of path  $P$  for commodity  $k$  and the length of the shortest path for commodity  $k$ . Constraint (5.4) guarantees that the length of any path  $P$  for commodity  $k$  must be greater than or equal to  $z_k + \delta_k^*$ . Constraints (5.5) and (5.6) guarantee that there is at least one path whose length is exactly equal to  $z_k + \delta_k^*$ . If the objective value is equal to 0, then we solve ISPL to optimality.

$$\text{Min} \sum_{k=1}^{|K|} \delta_k \quad (5.3)$$

$$\sum_{e \in P \in P_k} c_e - \delta_k - s_{kp} = z_k \quad \forall P \text{ for } k = 1, \dots, |K| \quad (5.4)$$

$$s_{kp} \leq M(1 - y_{kp}) \quad (5.5)$$

$$\sum_p y_{kp} = 1 \text{ for all } k \quad (5.6)$$

$$c, \delta, s \geq 0, y_{kp} \in \{0, 1\} \quad (5.7)$$

Since there is not much difference between the performance of our six algorithms, we only compare the solutions of Algorithm 0 (starting with the minimum cardinality paths) to test the performance. We summarize the result in Table 37.

Considering the effect of our bounds on infeasibilities of Type 1 and Type 2, the performance estimate for Global Crossing instance 1 changes from almost 260% to 11%. It tells us that the performance of our algorithms is much better than shown in Table 33, Table 34, and Table 35. Since the actual infeasibility must be at least as much as the summation of

**Table 37:** Adjusted performance by infeasibility bounds

Instance		$\sum z_k$	Infeasibility				Performance (%)	
			Type 1	Type 2	MIP	Alg. 0	vs. bounds	vs. optimal
		$a$	$b$	$c$	$d$	$e$	$\frac{e-b-c}{a+b+c}$	$\frac{e-d}{a+d}$
Tyco	1	7687	648	0	1970	3741	37.11	18.34
	2	7015	977	0	1827	4377	42.54	24.84
	3	8551	1340	0	2442	5716	44.24	29.78
Level 3	1	538854	553178	68171	–	1125794	43.48	–
	2	474123	426070	11553	–	1301166	94.71	–
	3	527705	405972	0	–	1418193	108.41	–
Global Crossing	1	1168093	2351156	272109	–	3028911	10.70	–
	2	690112	317784	31363	–	1840331	143.49	–
	3	741722	162979	17500	–	1889628	185.33	–

the infeasibilities of Type 1 and Type 2, we know that our algorithm should solve the Global Crossing ISPL instance 1 to within 11% of optimality. We also notice that our algorithms do not work as well for Tyco ISPL instances, even though Tyco has the simplest network structure. It coincides with the observation in Section 5.3.1.3: we get better solutions when there are many or few commodities than for intermediate sizes of  $|K|$ . Even though the Global Crossing network is more complicated, the number of commodities is large enough to get a better solution.

## 5.5 Summary

We propose 3 random schemes to generate feasible ISPL instances in this chapter. By using Random ISPL 3, we can generate a feasible ISPL instance and start our algorithms with the most different shortest path set.

We summarize some results of the performance of our algorithms as following:

1. For the intermediate-difficulty ISPL, 99% of instances can be solved to within 2.7% of optimality using intelligent starting paths, and to within 5.5% of optimality using random paths.
2. For hard ISPL, 95% of instances can be solved to within 6.6%, and 99% of instances can be solved to within 6.8%. These numbers can be decreased to 2.7% and 2.9% if we use Algorithms 1, 3, and 5.

3. Whether for intermediate-difficulty ISPL or hard ISPL, our algorithms will terminate in fewer than 35 iterations in 99% of instances. There is not much difference in the speed of convergence of algorithms when starting with minimum cardinality paths, random paths, or the more different paths (Random ISPL 3).
4. Algorithm 3 and Algorithm 5 are better than the other four algorithms.
5. Changing the shortest path set at each iteration does help the speed of convergence of our algorithms.
6. Algorithms did not work well for the telecommunication ISPL instances. The reason is that these instances are infeasible.

## CHAPTER VI

### CONCLUSIONS AND FUTURE RESEARCH

#### 6.1 *Conclusions*

This thesis deals with the Inverse Shortest Path Length Problem (ISPL). ISPL draws our attention because it has many applications in the real world, like transportation network improvement and bandwidth pricing.

Chapter 2 contains a literature review that includes previous research on inverse solution optimization problems, methods for solving inverse solution optimization problems, and the inverse objective value optimization problem. Previous work on the complexity of inverse objective value optimization problems is summarized in this chapter.

Chapter 3 discusses the complexity of ISPL. Previous work has already proved that ISPL is *NP*-complete in general. We show that ISPL can be solved in polynomial time for two extreme cases of physical networks, a tree and a complete graph. We find that ISPL is also polynomially solvable when the underlying network is a complete graph with a constant number of edges removed. We introduce some ideas such as a logical graph to discuss ISPL complexity when the network is a cycle. Some special classes of cycle ISPL, regular restricted cycle ISPL, and  $t - \max$  restricted cycle ISPL with  $|C| \geq 5t + 1$ , can also be solved in polynomial time. We also discuss the relationship between the number of commodities and the complexity of ISPL. When there are two commodities or the distance graph is complete, ISPL can be solved in polynomial time.

We propose six algorithms to solve ISPL in Chapter 4. The algorithms contain three subproblems: solving for shortest paths, minimizing infeasibility, and perturbing costs. Some properties of the algorithms, perturbation benefits and monotonicity, are discussed. We show that our six algorithms will be terminated in polynomial time and their tight worst case performance bound is  $|K| - 1$ .

In Chapter 5, we test the performance of our algorithms both on random ISPL instances

and ISPL instances from real world telecommunication problems. In order to generate feasible ISPL instances for testing, we propose three random generating schemes. For the intermediate-difficulty ISPL, 99% of instances can be solved to within 2.7% of optimality using intelligently-chosen starting paths. For hard ISPL, 99% of instances can be solved to within 2.9% of optimality if we use Algorithms 1, 3, and 5. Algorithms are terminated under 35 iterations for 99% instances of these two types of random ISPL. Algorithm 3 and Algorithm 5 perform better than the other four algorithms in general. The performance of the heuristics on telecommunication ISPL in the real world is worse than on random instances. Part of the infeasibility comes from the fact that the telecom instance of ISPL themselves are infeasible. We compute lower bounds on infeasibility by solving a mixed-integer program.

## 6.2 *Future Research*

Based on the conclusions we draw in Section 6.1, future research could cover both theoretical topics and practical applications.

1. The complexity of cycle ISPL.

We already identify some classes of cycle ISPL that can be solved in polynomial time. However, the complexity of general cycle ISPL is still unknown.

2. Improved perturbation algorithms.

We introduce an idea to improve our algorithms by adopting a different cost perturbation scheme. To see the benefit of combining the new cost perturbation scheme with our other algorithms could be interesting.

Cost perturbation cannot escape from the local optimum in Figure 31 because we maintain the monotonicity of infeasibility of each commodity. We do not allow the infeasibility of some commodity to increase when we perturb the cost vector, even if the total infeasibility remains the same.

To avoid the situation in Figure 31, we need some modification of the cost perturbation subproblem. Instead of putting an infeasibility restriction on every commodity, we

will restrict on the total infeasibility of commodities to remain constant. This can allow greater flexibility.

### 3. Telecommunication pricing.

According the data we got for telecommunication pricing, there is arbitrage and inconsistency. We could apply our ISPL algorithms to develop a more consistent pricing scheme for telecommunication companies. Another issue of interest in telecommunication pricing is finding a systematic way to fix the inconsistent pricing to remove arbitrary opportunities.

### 4. Transportation network improvement.

Inverse optimization can be applied in the improvement of transportation networks. Consider a transportation network with capacity on each edge, and the edge cost function depending on the flow through it. We are given a desired cost vector for some specific origins and destinations. We also have the construction cost function representing the cost to improve the edge. Under the budget, time and feasibility constraints, we must decide which edges should be improved to reach the desired service level.

According user equilibrium theory, drivers tend to choose their shortest path. Once we improve some edges, the flow on each edges will be changed again.

Inverse objective value optimization problem is a research frontier. Inverse shortest path length problem is just one topic in this field. We hope this thesis could be a starting point and numerous extensions or applications of ISPL will appear in future research.

## REFERENCES

- [1] S. Ahmed and Y. Guan. The inverse optimal value problem. Working paper, 2002.
- [2] R. Ahuja and J. Orlin. A faster algorithm for the inverse spanning tree problem. *Journal of Algorithms*, 34:177–193, 2000.
- [3] R. Ahuja and J. Orlin. Inverse optimization. *Operations Research*, April 2001.
- [4] D. Burton. *On the Inverse Shortest Path Problem*. PhD thesis, Facultes Universitaires Notre-Dame de la Paix de Namur, 1993.
- [5] D. Burton, B. Pulleyblank, and P. Toint. The inverse shortest paths problem with upper bounds on shortest paths costs. *Network Optimization*, pages 156–171, 1997.
- [6] D. Burton and P. Toint. On an instance of the inverse shortest paths problem. *Mathematical Programming*, 53:45–61, 1992.
- [7] D. Burton and P. Toint. On the use of an inverse shortest paths algorithm for recovering linearly correlated costs. Technical Report 91/09, June 1997.
- [8] M. Cai and Y. Li. Inverse matroid intersection problem. *Math. Methods Oper. Res.*, 45:235–243, 1997.
- [9] M. Cai and X. Yang. Inverse shortest path problems. In X. S. Zhang D. Z. Du and K. Cheng, editors, *Lecture Notes in Operations Research*, pages 1:242–248. Operations Research and its applications. First International Symposium, ISORA '95, Beijing World Publishing corporation, August 1995.
- [10] M. Cai, X. Yang, and J. Zhang. The complexity analysis of the inverse center location problem. *J. Global Optim.*, 15:213–218, 1999.
- [11] S. Chiu and J. Crametz. Taking advantage of the emerging bandwidth exchange market. Technical report, RateXLab, September 1999.
- [12] S. Chiu and J. Crametz. Surprising pricing. *Bandwidth Special Report, RISK, ENERGY and POWER RISK MANAGEMENT*, July 2000.
- [13] J. Day. *An Inverse Optimization Approach to Railroad Block*. PhD thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA., 2002.
- [14] J. Day, G. Nemhauser, and J. Sokol. Management of railroad impedances for shortest path-based routing. *Electronic Notes in Theoretical Computer Science*, (66), 2002.
- [15] M. Dell'Amico, F. Maffioli, and F. Malucelli. The base-matroid and inverse combinatorial optimization problem. *Discrete Applied Mathematics*, 128(2-3):337–353, June 2003.

- [16] A. Faragó, Á. Szentesi, and B. Szviatovszki. Inverse optimization in high-speed networks. *Discrete Applied Mathematics*, 129(1):89–98, June 2003.
- [17] S. Fekete, W. Hochstättler, S. Kromberg, and C. Moll. The complexity of an inverse shortest path problem. In Graham et. al., editor, *In Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the future*, pages 113–128, 1999.
- [18] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27:1–33, 1983.
- [19] A. Hayter. *Probabilities and Statistics*. Duxbury, 2002.
- [20] C. Heuberger. Inverse optimization:a survey on problems, methods, and results. *To appear in Journal of Combinatorial Optimization*, 2002.
- [21] D. Hochbaum. Efficient algorithms for the inverse spanning-tree problem. *Operations Research*, 51(5):0785–0797, September-October 2003.
- [22] ILOG. *ILOG CPLEX 7.0 Reference Manual*. 2000.
- [23] Z. Liu and J. Zhang. On inverse problem of maximum perfect matching. preprint, 2003.
- [24] G. Paleologo and S. Takriti. Bandwidth trading: A new market looking for help from the or community. *AIRO*, (3):1–4, Autumn 2001.
- [25] RateXLabs. Arbitrage opportunities-an update. Technical report, December 2000.
- [26] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1999.
- [27] P. Sokkalingam. The minimum cost flow problem:primal algorithms and cost perturbations. Master’s thesis, Department of Mathematics, Indian Institute of Technology, 1995.
- [28] P. Sokkalingam, R. Ahuja, and J. Orlin. Solving inverse spanning tree problem through network flow techniques. *Operations Research*, 47:291–298, 1999.
- [29] C. Tong and K. Lam. An embedded connectionist approach for the inverse shortest path problem. In *Neural Networks, 1994. IEEE world Congress on Computational Intelligence. 1994 IEEE International Conference*, pages 4607–4612, vol. 7, June 1994.
- [30] I. Wang. *Shortest Paths and Multicommodity Network Flows*. PhD thesis, Georgia Institute of Technology, May 2003.
- [31] Bandwidthmarket.com website. [www.bandwidthmarket.com/perl/lookuputil?db1=band](http://www.bandwidthmarket.com/perl/lookuputil?db1=band).
- [32] Global Crossing website. [www.globalcrossing.com/xml/network/net\\_map.xml](http://www.globalcrossing.com/xml/network/net_map.xml).
- [33] Level 3 website. [www.level3.com/577.html](http://www.level3.com/577.html).
- [34] Tyco website. [www.tycotelecom.com/networkservices/content.asp?page=networkmaps.asp](http://www.tycotelecom.com/networkservices/content.asp?page=networkmaps.asp).
- [35] C. Yang and J. Zhang. Two general methods for inverse optimization problems. *Applied mathematical letter*, 12(2):69–72, 1999.



- [36] X. Yang and J. Zhang. An inverse optimal value problem for minimum spanning tree. In *Internattional Conference of Inverse Problems*. Hong Kong City University, January 2002.
- [37] H. Zhang and M. Ishukawa. A neural network approach to inverse optimization. In *The 2nd R.I.E.C. International Symposium on Design and Architecture of Information Processing Systems Based on The Brain Information Principles (DAIPS)*, pages 197–200, Mar. 16-18, Sendai, 1998.
- [38] J. Zhang and Y. Lin. Computation of the reverse shortest-path problem. preprint, 2003.
- [39] J. Zhang and Z. Liu. Calculating some inverse linear programming problems. *Journal of computational applied mathematics*, 72:261–273, 1996.
- [40] J. Zhang and Z. Liu. A general model of some inverse combinatorial optimization problems and its solution method under l-infinity norm. *Journal of Combinatorial Optimization*, 6(2):207–227, 2002.
- [41] J. Zhang, Z. Ma, and C. Yang. A column generation method for inverse shortest path problems. *ZOR-Math. Methods Oper. Res.*, 41:347–358, 1995.

## VITA

Cheng-Huang Hung was born in Taipei, Taiwan on February 6, 1968. Cheng-Huang lived in Taipei until he graduated from Taipei Municipal Chien-Kuo High School in 1986.

Cheng-Huang did his undergraduate studies in National Chiao-Tung University, majoring in Transportation Engineering and Management. In 1990, he graduated and kept pursuing graduate studies in National Chiao-Tung University, majoring in Traffic and Transportation. During two years of graduate studies, he worked mainly in network and transportation. His master thesis topic is "A Study of Route Selection and Type of Control for Transportation of Low Level Radioactive Wastes". In 1992, he got a Master's degree in Engineering.

Cheng-Huang decided to stay in the same department to pursue a Ph.D. degree. In 1993, he met Susan and fell in love with her quickly. They married on March 27, 1994. At that moment, Cheng-Huang decided to leave school and work for his family. At the end of that year, his first child, Jim, was born.

Cheng-Huang first worked in Provincial Taiwan Housing & Urban Development Bureau as a contracted engineer in 1994. He was in charge of the planning and organization of two provincial public projects: ILan Living Circle and Peng Hu Living Circle. In 1995, he chose to join Taiwan Area National Expressway Engineering Bureau to contribute what he learned before. He worked there for four years. During these years, he was involved in several projects, including freeway construction, traffic improvement, draft of traffic control strategy. He found a shortage of theoretic foundation to enhance his work as time went by.

Cheng-Huang made a decision to continue graduate studies in Georgia Institute of Technology, majoring in Industrial and System Engineering in 1999. His research interest is in Operations Research, mainly focused on Optimization. He is not only interested in exploring theoretical research, but also in application. During the process of pursuing the Ph.D.,

his little daughter, Sophie, was born in 2002. He completed his Ph.D. in 2003 with a dissertation, "On the Inverse Shortest Path Length Problem". The problem can be applied in telecommunication pricing.

Cheng-Huang's primary hobbies are reading and sport. He is a baseball fan of the Braves. Also he likes to watch a lot of other sports, including football, soccer, and racing. He is also a baseball and soft ball player even though he is not as good as professional. His other hobbies include computers, movies, and music. Actually, the most fun Cheng-Huang has comes from playing with his two lovely children, Jim and Sophie, and beautiful wife, Susan.