

INTEGER PROGRAMMING APPROACHES FOR EQUAL-SPLIT NETWORK FLOW PROBLEMS

A Thesis
Presented to
The Academic Faculty

by

Amandeep Parmar

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology
August 2007

INTEGER PROGRAMMING APPROACHES FOR EQUAL-SPLIT NETWORK FLOW PROBLEMS

Approved by:

Shabbir Ahmed, Advisor
H. Milton Stewart School of Industrial
and Systems Engineering
Georgia Institute of Technology

Joel Sokol, Co-Advisor
H. Milton Stewart School of Industrial
and Systems Engineering
Georgia Institute of Technology

George Nemhauser
H. Milton Stewart School of Industrial
and Systems Engineering
Georgia Institute of Technology

Özlem Ergun
H. Milton Stewart School of Industrial
and Systems Engineering
Georgia Institute of Technology

Zonghao Gu
ILOG Inc.

Date Approved: May 2007

*To my family, who have always offered me unconditional love and support throughout the
course of my life.*

ACKNOWLEDGEMENTS

This work would not have been possible without the support and encouragement of my advisor, Dr. Shabbir Ahmed, under whose supervision I chose this topic and began the thesis. I want to thank him for the support, inspiration and helpfulness he showed me over the course of my graduate study at Georgia Tech.

I am also very grateful to Dr. Joel Sokol, my co-advisor, for guiding me throughout my thesis, providing insightful comments and assisting me in numerous ways. Much appreciation goes to Dr. George Nemhauser, Dr. Özlem Ergun, and Dr. Zonghao Gu, for making time out of their busy schedules to be serve on my thesis committee.

I would not have been able to finish this thesis, and for that matter graduate school without the unconditional love and constant support and encouragement from my parents and my future wife, Ardaman. They have always supported all my dreams and aspirations and stood by me in the good as well as the bad times.

I would also like to acknowledge my fellow PhD students, Adora, Alejandro, Clarence, Claudio, Daniel, Dan, Deniz, Doug, Faram, Fatma, Gizem, Helder, Jim, Juan Pablo, Kapil, Marcos, Ricardo, Richa, Sid, ,Ulas, Vijay, Yao-Hsuan and Yi-He, in my department at Georgia Tech for all their help throughout my graduate study.

Lastly I would like to thank all the faculty and staff members in my department for helping in my graduate study at Georgia Tech.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
LIST OF TABLES	viii
LIST OF FIGURES	ix
SUMMARY	x
I INTRODUCTION	1
1.1 Introduction	1
1.2 Integer Programming	2
1.2.1 Polyhedral Theory	3
1.2.2 Branch-and-Bound for solving MIPs	5
1.3 Overview of contributions	6
1.3.1 Equal-Split Multi-commodity flow problem	6
1.3.2 OSPF weight setting problem	7
1.3.3 Equal-Split problem on bipartite graphs	8
II EQUAL-SPLIT MULTI-COMMODITY FLOW PROBLEMS	10
2.1 Introduction	10
2.2 Problem formulation	13
2.3 Valid inequalities	15
2.3.1 A simple single node system	15
2.3.1.1 Family 1	17
2.3.1.2 Family 2	21
2.3.1.3 Family 3	25
2.3.2 A two node system	27
2.3.3 An alternate two node system	35
2.3.4 A multiple node system	41
2.3.5 Valid inequalities based on capacity	43
2.4 A branch-and-cut implementation	46
2.4.1 Test Problems	46

2.4.2	Cuts	46
2.4.3	Computational Results	47
2.4.4	Relative Strength of Cuts	48
2.5	Final Thoughts	55
III	OSPF WEIGHT SETTING PROBLEM	57
3.1	Introduction	57
3.2	Problem Formulation	60
3.3	A Decomposition based approach	63
3.3.1	Results	65
3.3.1.1	Implementation within branch-and-bound	66
3.4	Valid Inequalities	67
3.4.1	Specific Cuts	68
3.5	A branch-and-cut implementation	70
3.5.1	Test Problems	70
3.5.2	Cuts	70
3.5.3	Heuristics	71
3.5.3.1	Local search	72
3.5.3.2	Integer Programming (IP) heuristic	73
3.5.4	Symmetry	74
3.6	Computational Results	74
3.7	Comparison of Heuristic Quality	76
3.7.1	Genetic Algorithm Based Heuristic (GA)	77
3.7.2	Simulated Annealing Based Heuristic (SA)	78
3.7.3	Traditional Recommendations	79
3.7.4	Proposed Heuristics	79
3.7.5	Discussion of Results	80
3.8	Final Thoughts	83
IV	EQUAL-SPLIT FLOW PROBLEMS ON BIPARTITE GRAPHS	84
4.1	Problem Definition	84
4.2	Problem Formulation	87

4.3	Valid Inequalities	89
4.3.1	A simple system	89
4.3.2	A simple single node system	94
4.4	Structure based cuts	101
4.5	Branch-and-Cut Results	103
4.5.1	Heuristics	104
4.5.2	Computational Experiments	105
4.6	An Alternate Formulation	107
4.6.1	Valid Inequalities for $(P2)$	110
4.6.2	Comparison of two formulations	113
4.7	Final Thoughts	119
V	CONCLUSIONS	120
5.1	Summary of work done	120
5.2	Remaining Technical Issues	121
5.2.1	Equal-split multi-commodity network flow capacity cuts	121
5.2.2	OSPF weight setting symmetry	121
5.2.3	Equal-split flow on bipartite graphs: extended polyhedral study	122
	REFERENCES	124
	VITA	129

LIST OF TABLES

1	Node counts and time for instances of multi-commodity network flow problems	48
2	Time (in seconds) for instances of multi-commodity network flow problem(time limit: 4 hours)	52
3	Node Counts for instances of multi-commodity network flow problem(time limit: 4 hours)	53
4	Lower bound comparison of the decomposition method with Linear Programming lower bound (bold-face represents optimal soln found). Time limit: 600s	66
5	Size of the problems	71
6	Percentage optimality gap for instances of OSPF problem (time limit: 800 secs).	75
7	Comparison of different heuristic strategies	81
8	Lower and upper bounds for instances of equal-split assignment on bipartite graphs (time limit: 300s)	105
9	Time and node counts comparison	106
10	Comparison of two formulations	114
11	Comparison of two formulations: lower and upper bounds were returned at the end of 300s of computation time	115
12	Comparison of two formulations: Node counts and solve times	116
13	Comparison of two formulations with cutting planes and heuristics: lower and upper Bounds after 300s of computation time	117
14	Comparison of two formulations with cutting Planes and heuristics: node counts and computation times	118

LIST OF FIGURES

1	An equal-split system	1
2	A single commodity equal-split network flow instance	12
3	A single node equal-split system	16
4	Convex hull of the set Y	18
5	Convex quadratic function $h(k)$	21
6	A two node equal-split system	27
7	An alternate two node equal-split system	36
8	A multiple node equal-split system	41
9	Performance Profile: Solution times	54
10	Performance Profile: Node Counts	54
11	Performance Profile: Solution Times (performance ratio limited to 5)	55
12	Performance Profile: Node Counts (performance ratio limited to 5)	55
13	Equal-Cost Multipath principle	61
14	A Decomposition Algorithm	65
15	A general node	69
16	A Local Search Heuristic	72
17	Genetic Algorithm	77
18	Simulated Annealing	79
19	An equal-split problem on bipartite graphs	85
20	An (x, y, f) single node system.	95
21	A Single sink node with multiple sources	102
22	A simple subsystem	111

SUMMARY

In this thesis we develop integer programming approaches for solving network flow problems with equal-split restrictions. Such problems arise in traffic engineering of internet protocol networks that utilize shortest path based routing methods and use multiple paths for routing. Equal-split structure is used in protocols like OSPF and IS-IS that allow flow to be split among the multiple shortest paths. Equal-split assumptions also arise in peer-to-peer networks and road optimization problems.

First we consider a general multi-commodity network flow problem with equal-split restrictions. This problem is NP-hard in general. We perform a polyhedral study on mixed integer linear programming formulation for this problem. Valid inequalities are obtained, and are incorporated within a branch-and-cut framework to solve the problem. We provide fast separation schemes for several of the valid inequalities. Computational results show that our approach outperforms default CPLEX. On average, the number of nodes and computational time decreased by a factor of 30 and 90 respectively, with our approach.

Next, we consider the OSPF weight setting problem. Under the OSPF protocol every link in the network is assigned a weight, and flow is routed on shortest paths that are calculated with respect to these weights. If there are multiple shortest paths for some source destination pair, then flow is split equally among the outgoing arcs on the shortest path. OSPF weight setting problem is to determine the set of weights such that if flow is routed based on OSPF protocol, some measure of congestion in the network is minimized. We propose an integer programming formulation for this problem. A decomposition based approach to solve the problem is presented next. Valid inequalities, exploiting the structure, are obtained for this problem. We also propose heuristic methods to get good starting solutions for the problem. The proposed cutting planes and heuristic methods are integrated within a branch-and-cut framework to solve the problem. We present computational experiments that demonstrate the effectiveness of our approach to obtain solutions with

tight optimality gaps as compared with default CPLEX. On average, we are able to provide solutions with worst case optimality gap of 4.6% as compared with default CPLEX's gap of 35.1%.

Finally, we consider an equal-split flow problem on bipartite graphs. This problem is shown to be NP-complete in general. We present an mixed integer programming linear programming formulation for this problem that models the equal-split in a different way than the multi-commodity network flow problem discussed before. Valid inequalities and heuristic methods for this problem are proposed, and are integrated within the branch-and-cut framework. We present computational experiments demonstrating the effectiveness of our solution strategy. We present an alternate formulation for the problem with some favorable polyhedral properties. Lastly, a computational comparison between the two formulations is presented.

CHAPTER I

INTRODUCTION

1.1 Introduction

In this thesis we study network flow problems with equal-split restrictions. An equal-split restriction at a given node in the network requires that the flow out of the node has to be split equally among the set of chosen outgoing arcs. Consider a single node with positive flow coming in b and three outgoing arcs as shown in Figure 1. If arcs 1 and 2 are chosen to route flow then the equal-split restriction requires that the incoming flow b has to be equally split among arcs 1 and 2 and the outgoing flow on these arcs should be $\frac{b}{2}$ and the flow on arc 3 should be 0.

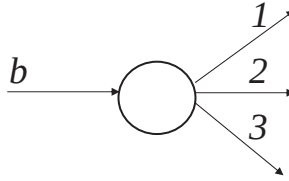


Figure 1: An equal-split system

Equal-split network flow problems arise in traffic engineering of internet flow networks. The optimization goal of traffic engineering is to make efficient use of network resources for routing flow in Internet Protocol (IP) networks. Open Shortest Path First (OSPF) [25][51] is the most commonly used intra-domain routing protocol in IP networks. Under OSPF protocol, traffic flow in an Internet Protocol (IP) network is routed on the shortest paths, based on pre-assigned weights on the network links, between each source and destination. If there are multiple shortest paths from some source to destination then the flow is split equally among all the outgoing arcs that are on the shortest paths. Intermediate System to Intermediate System (IS-IS) [56] is another example of the shortest path routing protocol that splits the flow equally on multiple shortest paths.

Equal-split restrictions on the flow add discrete structure to the network flow problems. This is different than just discrete version of the network flow problem where the flow on each arc has to be integral. In our problem the flow can be fractional, but the fractionality of the flow on any given arc depends on the amount of incoming flow as well as the number of outgoing arcs chosen.

While there exists efficient polynomial time algorithms for solving min cost network flow problems [3], with the equal-split restriction, the network flow problem becomes NP-hard, in general. Also for certain classes of equal-split network flow problems it is NP-hard even to get a constant factor approximation algorithm. So there is need for integer programming (IP) based approaches, to provide good solutions with quality guarantees.

In this thesis we develop IP based approaches to equal-split network flow problems. Our general approach is to consider simple subsystems of the problem, study the related polyhedra of the subsystems and derive valid inequalities. Since these inequalities are valid for subsystems, which are relaxations of original system, these inequalities are also valid for our original problem. For some families of valid inequalities we show that they are facet defining. For most of the families we also provide fast separation routines, to find violated inequalities at a given fractional solution. We provide computational experiments to demonstrate the effectiveness of these valid inequalities within branch-and-cut schemes. We specialize some of our results to the OSPF weight setting problem and an equal-split problem on bipartite graphs.

1.2 Integer Programming

In this section we provide a brief introduction to integer programming. We discuss the concepts of polyhedral theory used in this thesis. Nemhauser and Wolsey [53] and Schrijver [65] are comprehensive references for integer programming and polyhedral theory. The background presented in this section can be found, in much more detail, in both of these references.

Integer programming deals with maximizing or minimizing a linear function over many variables subject to (a) linear constraints and (b) integrality constraints on some or all the

variables. A general mixed integer linear program (MILP) is given by:

$$\min\{cx + hy : Ax + Gy \leq b, x \in \mathbb{Z}_+^n, y \in \mathbb{R}_+^p\} \quad (1)$$

where $\mathbb{Z}_+^n$ is the set of nonnegative integral n -dimensional vectors, $\mathbb{R}_+^p$ is the set of nonnegative real p -dimensional vectors, and $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_p)$ are the variables. An instance of the problem is specified by the data, (c, h, A, G, b) , with c an n -vector, h a p -vector, A an $m \times n$ matrix, G a $m \times p$ matrix and b a m -vector. The linear programming relaxation of an MILP is obtained by relaxing the constraints $x \in \mathbb{Z}_+^n$ to $x \in \mathbb{R}_+^n$. Non negativity restrictions on the decision variables can be removed, but it is common in most of the integer programming problems.

1.2.1 Polyhedral Theory

Let X be the set of feasible solutions of (1). The *convex hull* of X , denoted by $\text{co}(X)$, is given by:

$$\begin{aligned} \text{co}(X) = & \left\{ (x, y) \in \mathbb{R}_+^{n+p} : \exists (x^1, y^1), \dots, (x^k, y^k) \in X \text{ and } \lambda_1, \dots, \lambda_k \in \mathbb{R}_+ \right. \\ & \left. \text{s.t. } (x, y) = \sum_{i=1}^k \lambda_i (x^i, y^i), \sum_{i=1}^k \lambda_i = 1. \right\} \end{aligned}$$

The convex hull of a set of discrete points can be defined by a finite set of linear inequalities and therefore is a polyhedron. The inequalities that are necessary and sufficient to define the convex hull are the *facets* of the polyhedron. We now present some definitions and basic results from polyhedral theory. Let $P \subseteq \mathbb{R}^n$ be a polyhedron. Denote by $\dim(P)$, the dimension of polyhedron P .

Definition 1. P has dimension m if and only if the maximum number of affinely independent points within P is $m + 1$.

Definition 2. An inequality $\pi x \leq \pi_0$ is called a *valid inequality* for P if it is satisfied by all the points in P .

Definition 3. If $\pi x \leq \pi_0$ is a valid inequality for P , and $F = \{x \in P : \pi x = \pi_0\}$, F is called a *face* of P .

Definition 4. A face F of P is a facet of P if $\dim(F) = \dim(P) - 1$.

An inequality is called *facet-defining* if the face induced by the inequality is a facet. All the facet-defining inequalities of the polyhedron $P = \text{co}(X)$ are valid inequalities of $\text{co}(X)$, hence are valid for X as well.

We now give a theorem that characterizes facets and is used in this thesis to establish when a valid inequality is facet defining. For a polyhedron $P = \{x \in \mathbb{R}^n : a^i x \leq b_i, i = 1 \dots m\}$, equality set is defined as: $M^= = \{i \in \{1, \dots, m\} : a^i x = b_i \forall x \in P\}$. Denote the matrix defined by the indices of the equality set of P as $(E^=, f^=)$ of size $|M^=|$.

Theorem 1. Let $(E^=, f^=)$ be the equality set of the polyhedron $P \subseteq \mathbb{R}^n$, of size m_1 , and let $F = \{x \in P : \pi x \leq \pi_0\}$ be a non empty face of P , $F \subsetneq P$. Then the following two statements are equivalent:

- i. F is a facet of P .
- ii. If $\lambda x = \lambda_0$ for all $x \in F$ then

$$(\lambda, \lambda_0) = (\alpha \pi + u E^=, \alpha \pi_0 + u f^=) \text{ for some } \alpha \in \mathbb{R}^1 \text{ and some } u \in \mathbb{R}^{m_1}$$

Definition 5. A point $x \in P$ is an **extreme point** of P if there do not exists $x^1, x^2 \in P$, $x^1 \neq x^2$, such that $x = \frac{1}{2}x^1 + \frac{1}{2}x^2$.

Theorem 2. If $P \neq \emptyset$, and there exists at least one an extreme point of P , and $\max\{cx : x \in P\}$ is finite, then there is an optimal solution that is an extreme point.

This fact leads to the following theorem:

Theorem 3. The problem (1) is equivalent to:

$$\min\{cx + hy : (x, y) \in \text{co}(X)\} \tag{2}$$

Thus if we know all the facets of X , then the integer program (1) can be solved as a linear programming problem (2).

The aim of polyhedral study is to utilize the structure of the feasible region of the integer program, to develop valid inequalities or cutting planes to strengthen the linear

programming relaxation. We hope that the valid inequalities developed are facet defining for the convex hull of the feasible set. The aim is to characterize all the facet defining inequalities for the convex hull, but it is not possible in general because of the inherent complexity of the problem. So at best we can hope to provide partial description of the convex hull. However, experimental results have indicated that even identifying simple small classes of valid inequalities, are very useful in solving integer programming problems using branch-and-cut. In this thesis we perform polyhedral studies for developing useful cutting planes for the equal-split polyhedron.

1.2.2 Branch-and-Bound for solving MIPs

Solving MIPs is NP-hard in general. So enumeration based methods are required for getting optimal solutions to MIPs. The branch-and-bound algorithm is the most practical and most used method for MIPs. It was first proposed by Land and Doig [44]. The branch-and-bound algorithm explores the feasible region by partitioning it into smaller regions and smartly discarding some, based on bounds. The basic algorithm is presented below:

Branch-and-Bound Algorithm:

Step 0 Initialize first node, $k = 1$, set $z_{LB} = -\infty$ and $z_{UB} = +\infty$ and $X_k = X_{LP}$ (the linear programming relaxation of X).

Step 1 Solve the LP

$$z_k = \min\{cx + hy : (x, y) \in X_k\}.$$

Let $(\hat{x}, \hat{y})$ be an optimal solution.

If LP is infeasible, remove node k , Goto Step 3.

If $\hat{x} \in \mathbb{Z}_+^n$, and $z_k < z_{UB}$ set $z_{UB} = z_k$, remove node k and Goto Step 3.

If $z_k > z_{UB}$, remove node k , Goto Step 3.

Step 2 (Branching) Since x is not integer there exists j s.t. $\hat{x}_j \notin \mathbb{Z}_+$. Create two node k_1 and k_2 and partition X_k into two regions X_{k_1} and X_{k_2} where:

$$X_{k_1} = \{(x, y) \in X_k : x_j \leq \lfloor \hat{x}_j \rfloor\}$$

and

$$X_{k_2} = \{(x, y) \in X_k : x_j \geq \lfloor \hat{x}_j \rfloor + 1\}$$

Mark k as processed.

Step 3 If all the nodes are processed, set $z_{LB} = z_{UB}$. Otherwise, let $z^* = \min\{z_k : \forall k \text{ not processed}\}$. If $z^* > z_{LB}$, set $z_{LB} = z^*$.

Step 4 If $z_{LB} = z_{UB}$, terminate. Else select an unprocessed node and goto step 1.

The above description is the most simple branch-and-bound algorithm. More sophisticated node selection and branching techniques have been proposed and shown to work very well in practice. See Nemhauser and Wolsey [53] for more details.

At every node of the branch-and-bound algorithm, we need to solve a linear programming relaxation to get lower bounds. So a tighter LP relaxation will provide better lower bounds and possibly lead to exploration of lesser number of nodes. In order to make the LP relaxation tighter, violated valid inequalities or cutting planes, can be added at any of the fractional nodes within the branch-and-bound. This version of branch-and-bound method is called *branch-and-cut*. Refer to Crowder and Johnson [27] for more details. Such an approach has been demonstrated to be significantly superior to the straightforward branch-and-bound method. This emphasizes the need and importance of polyhedral studies for solving integer programming problems.

1.3 Overview of contributions

1.3.1 Equal-Split Multi-commodity flow problem

In Chapter 2, we consider a general multi-commodity flow problem with equal-split restrictions. This problem is similar to the general multi-commodity network flow problem but with the additional restriction that the flow outgoing from each node has to be equally split among the chosen arcs. This problem is NP-hard. We provide a mixed integer linear programming formulation (MILP) for the problem. Because of the weak linear programming relaxation of the mixed integer program we focus on developing cutting planes for this problem. We consider several simple subsystems of the problem and exploit the equal-split structure to generate cutting planes. We show that some of these valid inequalities are

facet defining for the subsystems considered and we develop fast separation procedures for most of them. The effectiveness of these cutting planes within branch-and-cut schemes is demonstrated by performing computational experiments and comparing it with the default branch-and-cut algorithm of CPLEX [26].

1.3.2 OSPF weight setting problem

In Chapter 3, we consider the OSPF weight setting problem. The OSPF weight setting problem is a traffic engineering problem for Internet Protocol networks that use OSPF [25][51] routing protocol. Under the OSPF protocol, flow is routed on the shortest paths, which are calculated based on weights on the arcs. If there are multiple paths of equal length, then the OSPF protocol stipulates the flow to be equally split among the outgoing arcs on the multiple shortest paths. These weights are pre-assigned and can be changed by network administrator. Given an IP network and traffic flow that need to be routed, the OSPF weight setting problem is to determine these set of weights such that, if the flow is routed based on the OSPF protocol, some measure of congestion in the network is minimized.

For this problem we propose a mixed integer linear programming formulation (MILP). We propose a decomposition based approach for the problem. Computational results indicate that this approach does not provide good results and also does not provide better lower bounds than the linear programming relaxation. In order to strengthen the linear programming relaxation, and to obtain better lower bounds, we use valid inequalities from Chapter 2 and develop specific families of valid inequalities for the problem. We propose heuristic methods to get good starting solutions to warm start the branch-and-bound procedure. We incorporate our valid inequalities and heuristics are within a branch-and-cut algorithm to solve the OSPF weight setting problem. Computational results, experimenting with some problems from literature and some randomly generated problems, are provided to demonstrate the effectiveness of the solution strategy. We implement two of the many heuristic methods from the literature, to get a comparison with our heuristic methods. Also, to see the additional benefits of these heuristic methods when incorporated with our cutting planes. The results from these heuristic methods are compared with our heuristics and

also the traditional recommendations for setting the OSPF weights as proposed by CISCO [24][25].

1.3.3 Equal-Split problem on bipartite graphs

In Chapter 4, we study an equal-split flow problem on bipartite graphs. This problem is a variant of the assignment problem. We are given two sets of nodes, sources and sinks. Each source node has a positive incoming flow that has to be routed out but can be sent to any of the given sinks. Each sink has a capacity on flow that can be routed to it. The problem is to send the flow from sources to sinks such that all the flow is routed and the capacity of the sinks is not exceeded. An additional requirement is that if the flow from a particular source is routed to multiple sinks then it has to be equally split among them. It is easy to see that for a given set of data this problem might not be feasible, so we allow the flow assignment on the sinks to go over its capacity. The objective is to minimize the sum of deviation over the capacity of the sinks.

To our knowledge no previous work has been done on the equal-split problem on bipartite graphs. We show that this problem is *NP*-hard in general. We provide a mixed integer linear programming formulation for this problem that models the equal-split restriction in a different way as compared to the equal-split multi-commodity flow problem of Chapter 2. We develop valid inequalities that exploit the special bipartite structure, by studying simple subsystems of the problem. Heuristic methods based on the integer programming formulation are also provided for this problem. We provide computational tests on randomly generated instances are provided to test the effectiveness of the cutting planes and the heuristic methods. Next an alternate formulation to model equal-split restriction is provided. The formulation proposed is shown to have favorable polyhedral properties. A comparison between the two formulation proposed for the equal-split problem on bipartite graphs is presented. This comparison is based on the optimal linear programming solutions and the effectiveness of each formulation to solve the problem. The comparison showed that the second formulation performed better in solving the problem with default branch-and-bound. But adding cutting planes to the first formulation led to better results.

Finally, in Chapter 5, we provide a summary of our contributions, and propose directions for further research.

CHAPTER II

EQUAL-SPLIT MULTI-COMMODITY FLOW PROBLEMS

2.1 Introduction

The network flow problem with equal-split restrictions is a generalization of the traditional network flow problem. We are given a directed graph $G = (V, E)$, with V the set of vertices and E the set of arcs with capacities u_e for e in E , and cost c_e for sending unit flow on arc $e \in E$. We are also given a demand matrix D , where D_{st} represents the flow to be sent from source s to destination t . In addition we have the restriction that, for each node in the network if more than one outgoing arc is chosen to route flow, then the flow has to be split equally among that node's chosen arcs. We wish to route the flow from each source to destination subject to the capacity requirements and equal-split restrictions at minimum cost.

Equal-split problems arise in traffic engineering of internet protocol networks that utilize shortest path based routing methods and use multiple paths for routing. Protocols like the Open Shortest Path First (OSPF) [25][51] and the Intermediate System to Intermediate System (IS-IS) [56] that allow Equal-Cost Multipath (ECMP) routing use the equal-split structure. ECMP routing means that flow is split equally among multiple paths used to send flow. Similar equal-split assumptions are also used in design of buses in embedded systems [38]. The equal-split restriction for data communication is made for reducing power dissipation. Equal-split allocation also arises in optimization of peer-to-peer networks [29], in road traffic optimization problems [30], and also in throughput maximization for a multi-hop packet radio network using slotted ALOHA [55].

A related problem, called the Equal Flow problem, has been studied in the literature in detail. In this problem we are given sets of arcs $R_1, \dots, R_k$ and the restriction is that flow has to be routed such that for each set R_i , each arc in the set carries equal flow. The integer version of the equal flow problem, where all the flows are restricted to be integer,

was first introduced by Sahni [63]. Ali et al. [5] showed that the integer equal flow problem, where the flow has to be integer, is NP-hard in general. They also showed that the special case when each set of arcs R_i is of cardinality 2 is also NP-complete. Sahni [63] showed that the maximum flow version of the equal flow problem is NP-complete. Also, Even et al. [32] showed that the problem remains NP-complete even if each arc has unit capacity. Srinathan et al. [67] showed that the problem is NP-complete even if we restrict that all arcs from each set R_i originate from a single node. Meyers [48] provided hardness results for getting an approximate solutions for this problem.

Integer version of the equal flow problem finds applications in crew scheduling [23], estimating driver costs for transit operations [72], and the two-duty period scheduling problem [66].

Calvete [22] presented a version of the network simplex algorithm for solving the general (non-integer) minimum cost equal flow problem. Ahuja et al. [2] studied the general (non-integer) version of the equal flow problem, where there is only one set R , that they called the simple equal flow problem, as a means of modelling a water resource management system. They presented polynomial time algorithms for solving this problem.

The equal flow problem is quite different from the equal-split problem we study in this thesis. For the equal-split problem, similar to the equal flow problem, we can say that we are given sets R_i for each node i ; R_i consists of all the outgoing arcs from node i . But unlike the equal flow problem, where all the arcs in each of these sets have to carry equal amount of flow, in the equal-split problem we can chose which arcs to use from R_i , to send the flow and the flow has to be split equally among only the chosen arcs. The equal-split multi-commodity flow problem is NP-complete.

Theorem 4. *The equal-split multi-commodity flow problem is NP-complete even for a single source destination pair.*

Proof. We will prove this result by reducing an instance of NP-complete problem X3C (exact cover by 3-sets) [37] to the equal-split network flow problem. The proof is taken from Pioro et al. [57], where the authors show that the OSPF weight setting problem is NP-complete.

An instance of X3C consists of a set $X = \{x_1, x_2, \dots, x_p\}$ of size $p = 3q$ elements and a family of subsets of X , $C = \{C_1, C_2, \dots, C_n\}$, $C_i \subset X$, $|C_i| = 3 \forall i = 1, \dots, n$. The question is whether there exists a cover $C' \subset C$ of q pairwise disjoint subsets of X .

Given an instance of X3C, we construct an instance of an equal-split multi-commodity network flow problem with a single commodity as follows (refer to Figure 2): For every set in the family C , construct a node C_i , $i = 1, \dots, n$. For every element in X construct a node x_j , $j = 1, \dots, p$. Construct three more nodes, a source node s , a sink node t and an intermediate node t' . Source node s is connected to every node C_i , and the sink node t with arcs each with capacity of 1 unit. Nodes C_i are connected with their elements (3) x with arcs each with capacity of $\frac{1}{3}$. For example in Figure 2, $C_1 = \{x_1, x_2, x_3\}$ and node C_1 is connected to nodes x_1, x_2 and x_3 with arcs of capacities $\frac{1}{3}$ each. All nodes x_i 's are connected to intermediate node t' with arcs each with capacity of $\frac{1}{3}$ units. Finally node t' is connected to sink node t with arc of capacity q . We need to send $q + 1$ units of flow from s to t . The question is whether there exists a flow assignment, satisfying the equal-split restriction, that sends $q + 1$ units of flow from s to t .

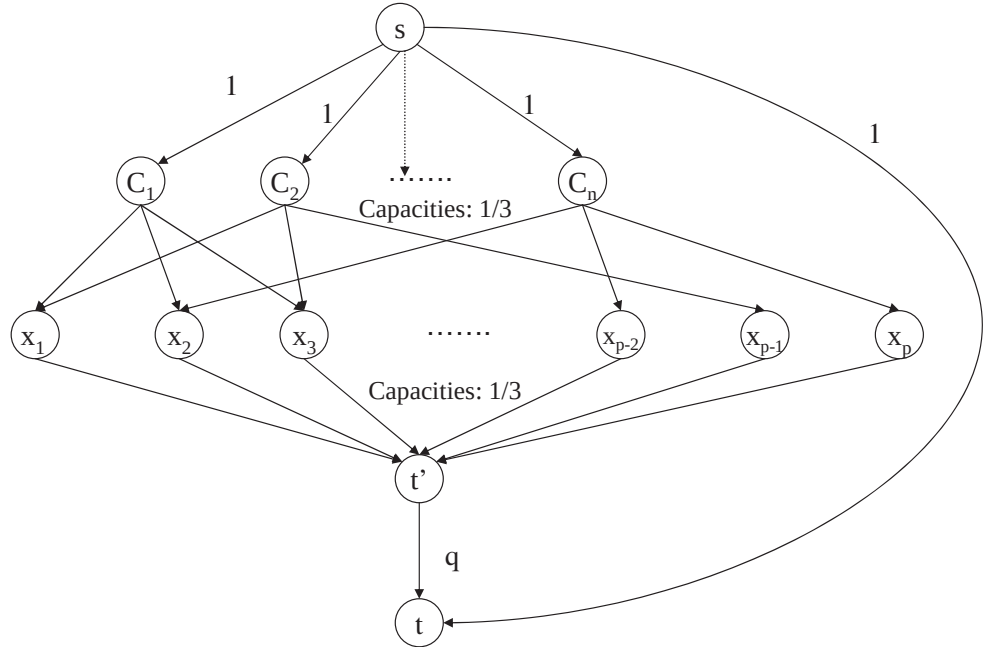


Figure 2: A single commodity equal-split network flow instance

Suppose there exists a cover $C' \subset C$ consisting of q pairwise disjoint subsets of X . Let $C' = \{C_{i(1)}, C_{i(2)}, \dots, C_{i(q)}\}$. Then the flow assignment of sending one unit of flow on arcs $(s, C_{i(j)})$, and sending $\frac{1}{3}$ on arcs from $C_{i(j)}$ to the three corresponding x 's. Sending $\frac{1}{3}$ from x_j to t' (since C' is a cover this flow will be q). Finally q units on (t', t) and one unit from (s, t) . All the other arcs carry zero flow. It is easy to see that this flow satisfies the equal-split restriction.

Note that the maximum flow from s to t is at most $q + 1$ (because of the capacities of arcs (t', t) and (s, t)). So if we have a flow of size $q + 1$, then the arcs (t', t) and (s, t) have to be saturated. Also all the arcs (x_j, t') $j = 1, \dots, p$ will have to be saturated as well. Now suppose we have a flow of size $q + 1$ from source s to destination t satisfying the equal-split restriction. Because of the equal-split restriction on source node s , there has to be q outgoing arcs from s other than (s, t) carrying one unit of flow. Let these arcs be $(s, C_{i(j)})$ $j = 1, \dots, q$. Also, since all the arcs (x_j, t') are saturated there will have to be $\frac{1}{3}$ flow coming into each node x_j . This implies that the family $C' = \{C_{i(1)}, C_{i(2)}, \dots, C_{i(q)}\}$ exactly covers X .

We have constructed a polynomial reduction from X3C to our problem where the answer to the X3C question is "yes" if and only if the answer to the equal-split problem is "yes". This shows that the equal-split multi-commodity network flow problem is NP-complete. $\square$

In the next section, we present an integer programming based formulation for the equal-split multi-commodity flow problem. Next, we study simple subsystems of the problem and develop valid inequalities for these subsystems. A branch-and-cut implementation using the cutting planes derived is presented next. Finally computational results are provided to demonstrate the effectiveness of the cutting planes.

2.2 Problem formulation

The equal-split multi-commodity flow problem can be mathematically formulated as a mixed integer linear programming (MILP) problem using the variables defined below:

Decision variables:

x_e^{st} binary variable denoting whether arc e is chosen to carry flow for the source destination pair (s, t) .

f_e^{st} flow on arc e for (s, t) pair.

d_v^{st} dummy flow variables for splitting the (s, t) flow for node v .

Let the set $(v, -)$ denote the set of outgoing arcs from node v and $(-, v)$ denote the set of incoming arcs into node v . The constraints for the problem are defined as follows:

Flow conservation constraints:

$$\sum_{e \in (-, v)} f_e^{st} - \sum_{e \in (v, -)} f_e^{st} = 0 \quad \forall v \in V \setminus \{s, t\} \quad \forall \text{ pairs } (s, t) \quad (3)$$

$$\sum_{e \in (-, s)} f_e^{st} - \sum_{e \in (s, -)} f_e^{st} = -D_{st} \quad \forall \text{ sources } s \quad (4)$$

$$\sum_{e \in (-, t)} f_e^{st} - \sum_{e \in (t, -)} f_e^{st} = D_{st} \quad \forall \text{ destinations } t \quad (5)$$

$$f_e^{st} \leq M x_e^{st} \quad \forall e \quad \forall \text{ pairs } (s, t) \quad (6)$$

Constraints (3), (4) and (5) ensure that total inflow is equal to total outflow. Constraints (6) force the flow to be sent on only arcs that are chosen. Here M is a large number that can be set to $\max\{D_{st}, u_e\}$.

Flow splitting constraints:

$$f_e^{st} \leq d_v^{st} \quad \forall e \in (v, -) \quad \forall v, \quad \forall \text{ pairs } (s, t) \quad (7)$$

$$d_v^{st} - f_e^{st} \leq M(1 - x_e^{st}) \quad \forall e \in (v, -) \quad \forall v, \quad \forall \text{ pairs } (s, t) \quad (8)$$

Constraints (7) and (8) ensure that each (s, t) flow is equally split amongst all the chosen outgoing arcs. Here M is a large number that can be set to D_{st} .

Capacity constraints:

$$\sum_{(s, t)} f_e^{st} \leq u_e \quad \forall e \quad (9)$$

Constraint (9) restricts the total flow on an arc to satisfy the capacity.

Using the variables and constraints defined above, a MILP model for multi-commodity flow problem with equal-split restrictions is as follows:

$$\begin{aligned}
\min \quad & \sum_e c_e \sum_{(s,t)} f_e^{st} \\
\text{s.t.} \quad & (3), (4), (5) \\
& (6), (7), (8), (9) \\
& x_e^{st} \in \{0, 1\}, f_e^{st} \geq 0, d_v^{st} \geq 0.
\end{aligned} \tag{P}$$

For networks containing more than 25 nodes and 25 demand pairs, the above MILP is very difficult to solve using a state-of-art MILP solver such as CPLEX. The main reason is the weak linear programming relaxation and therefore slow convergence of the branch-and-bound method. In order to strengthen the MILP formulation, we perform a polyhedral study and develop valid inequalities. We present the valid inequalities in the next section.

2.3 Valid inequalities

In this section we develop valid inequalities for simple subsystems of the equal-split multi-commodity network flow problem (P). Simple subsystems are studied because they are easy to characterize and usually provide good insights for the whole problem. Since the subsystem is a relaxation of (P) the proposed inequalities are also valid for (P). We consider four different subsystems.

2.3.1 A simple single node system

Consider a single node system which has n outgoing arcs and a flow of $b > 0$ units coming in, with the restriction that flow has to be equally split amongst the outgoing arcs that carry positive flow. See Figure 3.

Denote f_i to be flow on arc i , x_i to be a binary variable equal to 1 if and only if arc i is used to send flow, and d to be a dummy variable used to ensure that we split the flow equally. The system is given by:

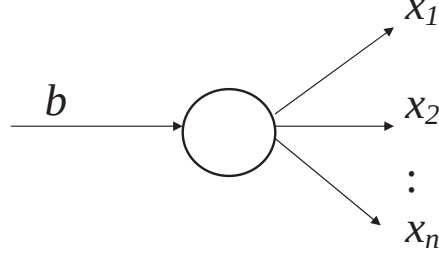


Figure 3: A single node equal-split system

$$\begin{aligned}
X^1 = \Big\{ \quad & (x, f, d) \in \{0, 1\}^n \times \mathbb{R}_+^n \times \mathbb{R}_+ : \\
& \sum_{i=1}^n f_i = b, \\
& f_i \leq bx_i \quad i = 1, \dots, n, \\
& f_i \leq d \quad i = 1, \dots, n, \\
& d \leq f_i + b(1 - x_i) \quad i = 1, \dots, n \quad \Big\}.
\end{aligned}$$

The first constraint balances flow in to flow out; the second constraint enforces that the flow can only be sent out on the chosen arcs; and the last two sets of constraints enforce equal-split of flow among the chosen arcs. We can now state the following result for $\text{co}(X^1)$.

Proposition 1. *The dimension of $\text{co}(X^1)$ is:*

$$\dim(\text{co}(X^1)) = \begin{cases} 0 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \\ 2n & \text{if } n \geq 3 \end{cases}$$

Proof. For $n = 1$ there is only one feasible point and for $n = 2$ there are only three feasible points in X^1 (and it can be easily verified that these points are affinely independent), so the result easily follows for $n = 1, 2$. We know that the dimension of convex hull of X^1 can be at most $2n$, because of the equality constraint. And to show that the dimension is exactly

$2n$, consider the following set of $2n + 1$ points in X^1 , for $n \geq 3$;

$$\begin{pmatrix} x_1 & x_2 & \dots & x_n & f_1 & f_2 & \dots & f_n & d \\ 1 & 0 & \dots & 0 & b & 0 & \dots & 0 & b \\ 0 & 1 & \dots & 0 & 0 & b & \dots & 0 & b \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 & 0 & \dots & b & b \\ 0 & 1 & \dots & 1 & 0 & \frac{b}{n-1} & \dots & \frac{b}{n-1} & \frac{b}{n-1} \\ 1 & 0 & \dots & 1 & \frac{b}{n-1} & 0 & \dots & \frac{b}{n-1} & \frac{b}{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \dots & 0 & \frac{b}{n-1} & \frac{b}{n-1} & \dots & 0 & \frac{b}{n-1} \\ 1 & 1 & \dots & 1 & \frac{b}{n} & \frac{b}{n} & \dots & \frac{b}{n} & \frac{b}{n} \end{pmatrix}$$

To see that these points are affinely independent, let $\lambda_1, \dots, \lambda_{2n+1}$ be the multipliers. Then we have the following set of equalities,

$$\lambda_i + \sum_{j \neq i} \lambda_{n+j} + \lambda_{2n+1} = 0 \quad \forall i = 1, \dots, n \quad (10)$$

$$b\lambda_i + \left(\frac{b}{n-1}\right) \sum_{j \neq i} \lambda_{n+j} + \left(\frac{b}{n}\right) \lambda_{2n+1} = 0 \quad \forall i = 1, \dots, n \quad (11)$$

$$b \sum_i \lambda_i + \left(\frac{b}{n-1}\right) \sum_j \lambda_{n+j} + \left(\frac{b}{n}\right) \lambda_{2n+1} = 0 \quad (12)$$

$$\sum_{i=1}^{2n+1} \lambda_i = 0. \quad (13)$$

Subtracting (10) from (13) and (11) from (12) yields $\sum_{j \neq i} \lambda_j + \lambda_{n+i} = 0$ and $b \sum_{j \neq i} \lambda_j + \left(\frac{b}{n-1}\right) \lambda_{n+i} = 0$ respectively. Subtracting the above two we get $\lambda_{n+i} = 0 \quad \forall i$. Plugging back values of λ_{n+i} , we get $\lambda_i = 0 \quad \forall i$, which yields $\lambda_{2n+1} = 0$. The only solution to (10)-(13) is $\lambda_i = 0$ for all i . Hence the points are affinely independent and the result follows. $\square$

We will now present some valid inequalities for the single node system defined above.

2.3.1.1 Family 1

Let us project out the variables f_i , and then aggregate the x_i variables to a general integer variable z , i.e., $z = \sum_{i=1}^n x_i$. Note that $z \in \{1, \dots, n\}$. Under this projection, the set X^1

becomes:

$$Y = \{(d, z) \in \mathbb{R}_+ \times \mathbb{Z}_+ : dz = b, 1 \leq z \leq n\}.$$

Y is a two dimensional set with one general integer variable between 1 and n and one continuous variable with one degree of freedom. Once z is chosen d is set to $\frac{b}{z}$. So the set Y consists of following set of n points in $\mathbb{R}^2$:

$$\left\{ (b, 1), \left(\frac{b}{2}, 2\right), \dots, \left(\frac{b}{n}, n\right) \right\}$$

The convex hull of these points can be constructed by joining consecutive points $(\frac{b}{k}, k)$ and $(\frac{b}{k+1}, k+1)$ for $k = 1, \dots, n-1$, and the first $(b, 1)$ and the last $(\frac{b}{n}, n)$ point. See Figure 4 for details. Thus the convex hull of Y is given as follows.

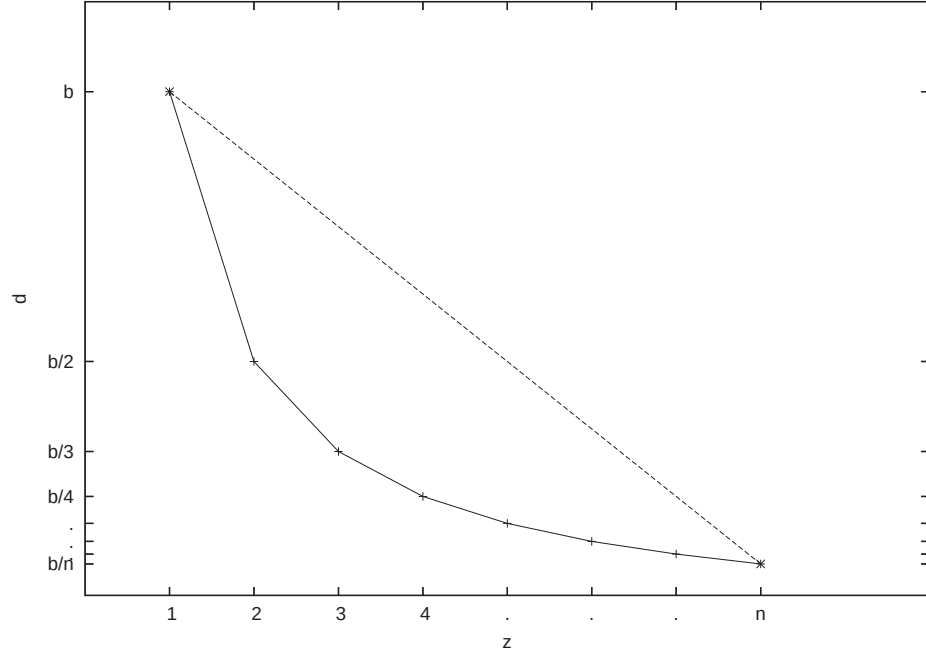


Figure 4: Convex hull of the set Y

Theorem 5. *The following inequalities define $\text{co}(Y)$:*

$$k(k+1)d + bz \geq (2k+1)b \quad k = 1, \dots, n-1, \quad (14)$$

$$nd + bz \leq b(n+1). \quad (15)$$

The proof of the theorem (omitted here) is constructed by obtaining the convex hull of n points in a two dimensional plane as discussed above (Fig 4. Valid inequalities (14) with z replaced by $\sum_{i=1}^n x_i$ are also valid for X^1 and indeed are facet defining.

Proposition 2. *The following inequalities are valid for X^1*

$$k(k+1)d + b \sum_{i=1}^n x_i \geq (2k+1)b \quad k = 1, \dots, n-1. \quad (16)$$

Proof. Let $\sum_i x_i = l \geq 1$, then $d = \frac{b}{l}$. We have $k(k+1)\frac{b}{l} + bl - (2k+1)b = \frac{b}{l} \left((l - \frac{2k+1}{2})^2 - \frac{1}{4} \right) \geq 0 \quad \forall l \text{ integer.}$ $\square$

Proposition 3. *The family of valid inequalities (16) for X^1 are facet defining for $1 \leq k \leq n-2$.*

Proof. Consider all the points in X^1 that satisfy (16) at equality. It is easy to see that these points are such that $x_i = 1 \quad \forall i \in I$, $x_i = 0$ otherwise, where $I \subseteq \{1, \dots, n\}$ and $|I| = k$ or $k+1$. Now suppose such points satisfy the identity $\sum_{i=1}^n \lambda_i x_i + \sum_{i=1}^n \lambda_{n+i} f_i + \lambda_{2n+1} d = \lambda_0$ for some $(\lambda, \lambda_0) \in \mathbb{R}^{2n+2}$. Then (λ, λ_0) should satisfy the following set of equalities;

$$\sum_{i \in I} \lambda_i + \frac{b}{k} \sum_{i \in I} \lambda_{n+i} + \frac{b}{k} \lambda_{2n+1} = \lambda_0 \quad \forall I \subseteq \{1, \dots, n\}, |I| = k \quad (17)$$

$$\sum_{i \in I} \lambda_i + \frac{b}{k+1} \sum_{i \in I} \lambda_{n+i} + \frac{b}{k+1} \lambda_{2n+1} = \lambda_0 \quad \forall I \subseteq \{1, \dots, n\}, |I| = k+1 \quad (18)$$

From (17), taking appropriate sets I and subtracting we get $\lambda_i + \frac{b}{k} \lambda_{n+i} = \lambda_j + \frac{b}{k} \lambda_{n+j}$. Similarly from (18) we get $\lambda_i + \frac{b}{k+1} \lambda_{n+i} = \lambda_j + \frac{b}{k+1} \lambda_{n+j}$. From these two equations we get $\lambda_i = \lambda_j$ and $\lambda_{n+i} = \lambda_{n+j} \quad \forall i, j$. Plugging back into (17) and (18) we get two equations with four unknowns (we denote $\lambda_i = \lambda_1 \quad \forall i$ and $\lambda_{n+i} = \lambda_{n+1} \quad \forall i$)

$$k\lambda_1 + b\lambda_{n+1} + \frac{b}{k} \lambda_{2n+1} - \lambda_0 = 0 \quad (19)$$

$$(k+1)\lambda_1 + b\lambda_{n+1} + \frac{b}{k+1} \lambda_{2n+1} - \lambda_0 = 0 \quad (20)$$

Both the equations (19) and (20) are linearly independent and hence the solution set is a two dimensional linear subspace. A general solution to above equalities (19) and (20) is

given by:

$$\begin{pmatrix} \lambda_1 \\ \lambda_{n+1} \\ \lambda_{2n+1} \\ \lambda_0 \end{pmatrix} = \begin{pmatrix} b \\ 0 \\ k(k+1) \\ (2k+1)b \end{pmatrix} \mu_1 + \begin{pmatrix} 0 \\ 1 \\ 0 \\ b \end{pmatrix} \mu_2$$

To check the correctness of the solution, plugging the values into (19) we get:

$$kb\mu_1 + b\mu_2 + \frac{b}{k}k(k+1)\mu_1 - ((2k+1)\mu_1 + \mu_2)b = 0$$

and plugging the values into (19) we get:

$$(k+1)b\mu_1 + b\mu_2 + \frac{b}{k+1}k(k+1)\mu_1 - ((2k+1)\mu_1 + \mu_2)b = 0.$$

Hence both the vectors of the solution belong to the two dimensional solution space. Also since the vectors are linearly independent, they span the solution space.

So the general solution to (17) and (18) can be written as:

$$\begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \lambda_{n+1} \\ \vdots \\ \lambda_{2n} \\ \lambda_{2n+1} \\ \lambda_0 \end{pmatrix} = \begin{pmatrix} b \\ \vdots \\ b \\ 0 \\ \vdots \\ 0 \\ k(k+1) \\ (2k+1)b \end{pmatrix} \mu_1 + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 1 \\ 0 \\ b \end{pmatrix} \mu_2$$

The first vector is the vector defining the valid inequality (16) and the second vector is the one defining the equality in the description of X^1 . Hence the result follows from Theorem 1. We also require $1 \leq k \leq n-2$ to get adequate number of affinely independent points for the facet defining property. $\square$

Separation

In order to use the valid inequalities (16) in a branch-and-cut algorithm, we need to separate

them, i.e., find out which inequalities are violated at a given fractional solution. Since the family of inequalities (16) is polynomial in size ($(n - 1)$ for each node i), a straightforward separation scheme could just go through each of them, but we can do better as shown next.

Given a fractional solution $(\widehat{\mathbf{x}}, \widehat{d})$, let us define $h(k)$ as

$$h(k) = k(k + 1)\widehat{d} + b \sum_{i=1}^n \widehat{x}_i - (2k + 1)b \quad (21)$$

as the difference between the left hand and the right hand side of (16). In order to find a violated inequality of the form (16) we need to find a $0 \leq k \leq n - 1$ such that $h(k) < 0$. From the definition of $h(k)$, it is clear that it is a convex quadratic function in k (since $\widehat{d} > 0$ when $b > 0$). Let k^* be the point where $h(k)$ is minimized. We know that $k^* = \frac{b}{\widehat{d}} - \frac{1}{2}$. Now if $h(k^*) \geq 0$ then we know that $h(k) \geq 0 \forall k$, otherwise we can find the two roots k^1 and k^2 of $h(k) = 0$ and $h(k) < 0$ for all the integers between (k^1, k^2) , see Figure 5. This is true only if both the roots of $h(k)$ are real. If both the roots are not real then $h(k) > 0 \forall k$. This gives a constant time separation routine for the family of valid inequalities (16).

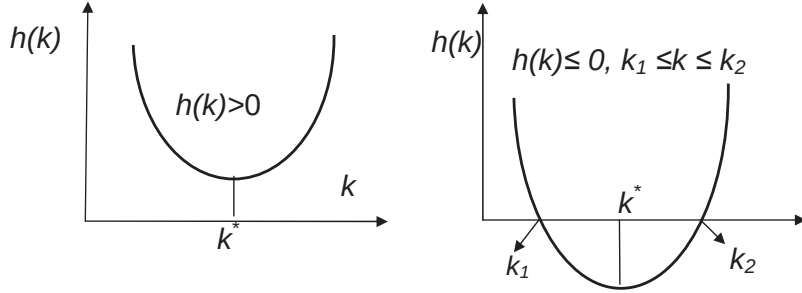


Figure 5: Convex quadratic function $h(k)$

2.3.1.2 Family 2

We present another family of valid inequalities for X^1 .

Proposition 4. Let $S \subseteq \{1, \dots, n\}$ and $\{f^{(1)}, \dots, f^{(|S|)}\}$ denote a permutation of $\{f_i$:

$i \in S\}$. The following inequalities are valid for X^1 :

$$k(k+1)d + b \sum_{i \notin S} x_i - b \sum_{i \in S} x_i + 2 \sum_{i=1}^{|S|} (k+i) f^{(i)} \geq (2k+1)b \quad (22)$$

$$\forall S : |S| \leq n-2, 1 \leq k \leq n-|S|-1$$

Proof. To prove the validity of (22) consider a point in X^1 and let $\sum_i x_i = l$, $|S| = o$ and $\sum_{i \in S} x_i = m$, $o \geq m$. We know that $d = \frac{b}{l}$ and the flow will be equal to $\frac{b}{l}$ for all the arcs with $x_i = 1$. Now, among all the permutations of set S the smallest value of left hand side of (22) is given by the permutation $\{f^{(1)}, \dots, f^{(m)}, \dots, f^{(o)}\}$, where $f^{(1)} = \dots = f^{(m)} = \frac{b}{l}$ and $f^{(m+1)} = \dots = f^{(o)} = 0$. The value of left hand side of the valid inequality in question for this permutation becomes:

$$k(k+1)\frac{b}{l} + b(l-m) - bm + \frac{2kmb}{l} + \frac{m(m+1)b}{l}$$

hence we need to show that $k(k+1)\frac{b}{l} + b(l-m) - bm + \frac{2kmb}{l} + \frac{m(m+1)b}{l} - (2k+1)b \geq 0$.

Note that the above condition is equivalent to

$$\frac{b}{l} \left[l^2 - \frac{(2m+2k+1)l}{2} + 2km + m^2 + m + k^2 + k \right] \geq 0 \Leftrightarrow$$

$$\frac{b}{l} \left[\left(l - \frac{(2m+2k+1)}{2} \right)^2 + \left(2km + m^2 + m + k^2 + k - \left(\frac{(2m+2k+1)}{2} \right)^2 \right) \right] \geq 0 \Leftrightarrow$$

$$\frac{b}{l} \left[\left(l - \frac{(2m+2k+1)}{2} \right)^2 - \frac{1}{4} \right] \geq 0$$

which is true since both l and m are integers. This completes the validity proof. $\square$

Proposition 5. The family of valid inequalities (22) for X^1 are facet defining for all sets S with $|S| = 1$.

Proof. We prove the facet defining property when $|S| = 1$. Let $S = \{j\}$, then the inequality (22) becomes,

$$k(k+1)d + b \sum_{i \neq j} x_i - bx_j + 2(k+1)f_j \geq (2k+1)b \quad \forall j, 1 \leq k \leq n-2 \quad (23)$$

To prove the facet defining property, consider all the points in X^1 that satisfy (23) at equality. The points are;

$$x_j = 0, x_i = 1 \forall i \in I, 0 \text{ o.w.}, \forall I \subseteq \{1, \dots, j-1, j+1, \dots, n\} \text{ s.t. } |I| = k \text{ or } k+1$$

$$x_j = 1, x_i = 1 \forall i \in I, 0 \text{ o.w.}, \forall I \subseteq \{1, \dots, j-1, j+1, \dots, n\} \text{ s.t. } |I| = k \text{ or } k+1$$

Now suppose such points satisfy the identity $\sum_{i=1}^n \lambda_i x_i + \sum_{i=1}^n \lambda_{n+i} f_i + \lambda_{2n+1} d = \lambda_0$ for some $(\lambda, \lambda_0) \in \mathbb{R}^{2n+2}$. Then (λ, λ_0) should satisfy the following set of equalities;

$$\sum_{i \in I} \lambda_i + \frac{b}{k} \sum_{i \in I} \lambda_{n+i} + \frac{b}{k} \lambda_{2n+1} = \lambda_0 \quad \forall I \subseteq \{1, \dots, n\}, |I| = k \quad (24)$$

$$\sum_{i \in I} \lambda_i + \frac{b}{k+1} \sum_{i \in I} \lambda_{n+i} + \frac{b}{k+1} \lambda_{2n+1} = \lambda_0 \quad \forall I \subseteq \{1, \dots, n\}, |I| = k+1 \quad (25)$$

$$\sum_{i \in I} \lambda_i + \lambda_j + \frac{b}{k+1} \sum_{i \in I} \lambda_{n+i} + \frac{b}{k+1} \lambda_{n+j} + \frac{b}{k+1} \lambda_{2n+1} = \lambda_0 \quad \forall I \subseteq \{1, \dots, n\}, |I| = k \quad (26)$$

$$\sum_{i \in I} \lambda_i + \lambda_j + \frac{b}{k+2} \sum_{i \in I} \lambda_{n+i} + \frac{b}{k+2} \lambda_{n+j} + \frac{b}{k+2} \lambda_{2n+1} = \lambda_0 \quad \forall I \subseteq \{1, \dots, n\}, |I| = k+1 \quad (27)$$

From (24), taking appropriate sets I we get $\lambda_i + \frac{b}{k} \lambda_{n+i} = \lambda_l + \frac{b}{k} \lambda_{n+l}$ for all $i \neq l \neq j$. Also from (25), taking appropriate sets I we get $\lambda_i + \frac{b}{k+1} \lambda_{n+i} = \lambda_l + \frac{b}{k+1} \lambda_{n+l}$ for all $i \neq l \neq j$. From these two equalities we get that $\lambda_i = \lambda_l$ & $\lambda_{n+i} = \lambda_{n+l} \forall i \neq l \neq j$. Plugging it back into (24)-(27) we get the following four equations with six unknowns (denote $\lambda_i = \lambda_1, \lambda_{n+i} = \lambda_{n+1} \ i \neq j$);

$$k\lambda_1 + b\lambda_{n+1} + \frac{b}{k} \lambda_{2n+1} = \lambda_0 \quad (28)$$

$$(k+1)\lambda_1 + b\lambda_{n+1} + \frac{b}{k+1} \lambda_{2n+1} = \lambda_0 \quad (29)$$

$$k\lambda_1 + \lambda_j + \frac{bk}{k+1} \lambda_{n+1} + \frac{b}{k+1} \lambda_{n+j} + \frac{b}{k+1} \lambda_{2n+1} = \lambda_0 \quad (30)$$

$$(k+1)\lambda_1 + \lambda_j + \frac{b(k+1)}{k+2} \lambda_{n+1} + \frac{b}{k+2} \lambda_{n+j} + \frac{b}{k+2} \lambda_{2n+1} = \lambda_0 \quad (31)$$

A general solution to above equalities (28)-(31) is given by:

$$\begin{pmatrix} \lambda_1 \\ \lambda_j \\ \lambda_{n+1} \\ \lambda_{n+j} \\ \lambda_{2n+1} \\ \lambda_0 \end{pmatrix} = \begin{pmatrix} b \\ -b \\ 0 \\ 2 \\ k(k+1) \\ (2k+1)b \end{pmatrix} \mu_1 + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ b \end{pmatrix} \mu_2$$

It is easy to check that the given solution is indeed a general solution to (28)-(31). Also the solution space is a two dimensional linear subspace and both the vectors are linearly independent of each other, and the subspace spanned by these two vectors indeed is the solution set to (28)-(31).

The first vector is the vector defining the valid inequality (23) and the second vector is the vector defining the equality in the description of X^1 . Hence the result follows from Theorem 1. $\square$

Separation

We now present an efficient way to separate the family of valid inequalities (22).

Let k be fixed and $(\widehat{x}, \widehat{f}, \widehat{d})$ be a fractional point. In order to separate (22), we need to find a set S such that the inequality is violated. Rewriting the inequality by collecting the terms that do not depend on S and aggregating them as C , we have the difference between left hand side and right hand side of (22) as

$$C + \sum_{i \in S} (2k\widehat{f}_i - 2b\widehat{x}_i) + 2 \sum_{i=1}^{|S|} i f^{(i)} \quad (32)$$

We need to find a set S and the permutation such that the term (32) is < 0 . We know that if an arc i is in the set S then the minimum contribution it would have towards (32) is: $(2k\widehat{f}_i - 2b\widehat{x}_i) + 2\widehat{f}_i = \beta_i$. If $\beta_i \geq -C$ for all i then none of the inequalities are violated. Otherwise if $\beta_i < -C$ for some i , then the set $S = \{i\}$ is one of the violated set. This is true for a fixed value of k , so we still need to check for all values of k . This gives a linear

time (checking for each k) separation scheme for the family of valid inequalities given by (22).

2.3.1.3 Family 3

Proposition 6. *Let $S \subseteq \{1, \dots, n\}$. Then the following inequalities are valid for X^1 :*

$$d + b \sum_{j \in S} x_j + \sum_{j \notin S} f_j \leq (|S| + 1)b \quad \forall |S| \geq 1 \quad (33)$$

Proof. Let $\sum_j x_j = l$, $|S| = o$ and $\sum_{i \in S} x_i = m$. We need to show that $\frac{b}{l} + bm + (l - m)\frac{b}{l} \leq (o+1)b$. But since $(m+1)b \leq (o+1)b$, it is sufficient to show that $\frac{b}{l} + bm + (l - m)\frac{b}{l} \leq (m+1)b$. Note that this condition is equivalent to $\frac{b}{l}(1 - m) \leq 0$ which is true for all $m \geq 1$. This completes the proof. $\square$

Proposition 7. *The family of valid inequalities (33) for X^1 are facet defining for $|S| = 1$ and $n \geq 3$.*

Proof. We prove the facet defining property for the case when $|S| = 1$. Let $S = \{j\}$, then the inequality becomes,

$$d + bx_j + \sum_{i \neq j} f_i \leq 2b \quad (34)$$

Now consider all the points that satisfy (34) at equality. These points are precisely,

- For each k , $x_k = 1, x_i = 0 \quad \forall i \neq k$
- For each $k \neq j$, $x_j = 1, x_k = 1, x_i = 0 \quad \forall i \neq j, k$
- For each $k, l \neq j$, $x_j = 1, x_k = 1, x_l = 1, x_i = 0 \quad \forall i \neq j, k, l$

Now suppose such points satisfy the identity $\sum_{i=1}^n \lambda_i x_i + \sum_{i=1}^n \lambda_{n+i} f_i + \lambda_{2n+1} d = \lambda_0$ for some $(\lambda, \lambda_0) \in \mathbb{R}^{2n+2}$. Then (λ, λ_0) should satisfy the following set of equalities;

$$\lambda_k + b\lambda_{n+k} + b\lambda_{2n+1} = \lambda_0 \quad \forall k \quad (35)$$

$$\lambda_k + \lambda_j + \frac{b}{2}\lambda_{n+k} + \frac{b}{2}\lambda_{n+j} + \frac{b}{2}\lambda_{2n+1} = \lambda_0 \quad \forall k \neq j \quad (36)$$

$$\lambda_k + \lambda_j + \lambda_l + \frac{b}{3}\lambda_{n+k} + \frac{b}{3}\lambda_{n+j} + \frac{b}{3}\lambda_{n+l} + \frac{b}{3}\lambda_{2n+1} = \lambda_0 \quad \forall k, l, k \neq l \neq j \quad (37)$$

We can easily derive from (35)-(37) that $\lambda_k = \lambda_l$ & $\lambda_{n+k} = \lambda_{n+l} \quad \forall k, l \quad k \neq l \neq j$. Plugging the values back into (35)-(37) we get four equations with six unknowns (denote $\lambda_k = \lambda_1$ & $\lambda_{n+k} = \lambda_{n+1} \quad \forall k \neq j$):

$$\lambda_1 + b\lambda_{n+1} + b\lambda_{2n+1} = \lambda_0 \quad (38)$$

$$\lambda_j + b\lambda_{n+j} + b\lambda_{2n+1} = \lambda_0 \quad (39)$$

$$\lambda_1 + \lambda_j + \frac{b}{2}\lambda_{n+1} + \frac{b}{2}\lambda_{n+j} + \frac{b}{2}\lambda_{2n+1} = \lambda_0 \quad (40)$$

$$2\lambda_1 + \lambda_j + \frac{2b}{3}\lambda_{n+1} + \frac{b}{3}\lambda_{n+j} + \frac{b}{3}\lambda_{2n+1} = \lambda_0 \quad (41)$$

A general solution to (38)-(41) is given by :

$$\begin{pmatrix} \lambda_1 \\ \lambda_j \\ \lambda_{n+1} \\ \lambda_{n+j} \\ \lambda_{2n+1} \\ \lambda_0 \end{pmatrix} = \begin{pmatrix} 0 \\ b \\ 1 \\ 0 \\ 1 \\ 2b \end{pmatrix} \mu_1 + \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ b \end{pmatrix} \mu_2$$

It is easy to check that the given solution is indeed a general solution to (38)-(41). Also the solution space is a two dimensional linear subspace and both the vectors are linearly independent of each other, and the subspace spanned by these two vectors indeed is the solution set to (38)-(41).

The first vector is the vector defining the valid inequality in question and the second vector is the vector defining the equality in the description of X^1 . Hence the result follows from Theorem 1. $\square$

Separation

We now present efficient way to separate the family (33): Let $(\hat{x}, \hat{f}, \hat{d})$ be a fractional point. In order to separate the inequality (33) we need to find a set S s.t.

$$\hat{d} + b \sum_{j \in S} \hat{x}_j + \sum_{j \notin S} \hat{f}_j > (|S| + 1)b$$

Let z_j be a binary variable which is 1 if $j \in S$, 0 otherwise. Then we need to maximize the following function to find a violated inequality:

$$\begin{aligned} & \sum_{j=1}^n b\hat{x}_j z_j + \sum_{j=1}^n \hat{f}_j(1 - z_j) - \left(\sum_{j=1}^n z_j + 1\right)b \\ &= \sum_{j=1}^n (b\hat{x}_j - \hat{f}_j - b)z_j \quad (\text{since } \sum f_j = b.) \end{aligned}$$

Now if $\sum_{j=1}^n (b\hat{x}_j - \hat{f}_j - b)z_j > -\hat{d}$, then we have a violated inequality. In order to test it, compute $(b\hat{x}_j - \hat{f}_j - b)$ for every arc j , denote it by β_j . It is easy to see that $\beta_j \leq 0 \forall j$. So if $\beta_j \leq -\hat{d}$ for all j , then none of the inequalities are violated, since $\beta_j \leq 0$ and $d \geq 0$. If $\exists j$ s.t. $\beta_j > -\hat{d}$, then the set $S = \{j\}$ represents a violated inequality. This provides a linear time separating scheme for above-defined inequalities.

In this section we considered a single node system and developed three different families of valid inequalities for this system. We also proved that these valid inequalities are facet defining under some conditions and provided fast schemes to separate these valid inequalities. Now we will generalize the single node system into a two node system and perform similar analysis.

2.3.2 A two node system

We now consider a two node system of a network flow problem with equal-split shown in Figure 6. Node 1 has n_1 outgoing arcs with arc 1 connecting to node 2 which has n_2 outgoing arcs. There is a positive flow b entering node 1 and there is no exogenous flow entering node 2.

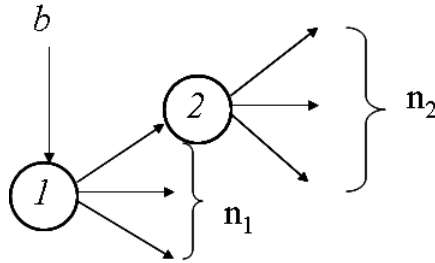


Figure 6: A two node equal-split system

We shall use the following notation:

Node 1 variables

f_i Flow on arc i

x_i Binary variable indicating if arc i is used or not

d^f Dummy variable to equate flow on chosen arcs

b Flow entering arc i (we assume $b > 0$)

Node 2 variables

g_i Flow on arc i

y_i Binary variable indicating if arc i is used or not

d^g Dummy variable to equate flow on chosen arcs

The two node system is defined as X^2 :

$$\begin{aligned}
X^2 = \Big\{ & (d^f, d^g, f, g, x, y) \in \mathbb{R}_+^2 \times \mathbb{R}_+^{n_1+n_2} \times \{0, 1\}^{n_1+n_2} : \\
& \sum_{i=1}^{n_1} f_i = b, \\
& \sum_{i=1}^{n_2} g_i = f_1, \\
& f_i \leq b x_i \quad i = 1, \dots, n_1, \\
& g_i \leq f_1 y_i \quad i = 1, \dots, n_2, \\
& f_i \leq d^f \quad i = 1, \dots, n_1, \\
& g_i \leq d^g \quad i = 1, \dots, n_2, \\
& d^f \leq f_i + b(1 - x_i) \quad i = 1, \dots, n_1, \\
& d^g \leq g_i + f_1(1 - y_i) \quad i = 1, \dots, n_2 \Big\}
\end{aligned}$$

As before the flow has to be equally split among the chosen arcs for both the nodes 1 and 2. In this system even though the arc from node 1 to 2 is not chosen, some of the outgoing arcs from node 2 can still have y equal to 1 but the flow corresponding to those arcs will be zero in that case. Let $X^1(n)$ be defined as a single node system with n outgoing arcs, so if $n_1 = 0$ or $n_2 = 0$ our system X^2 can be written as $X^1(n_2)$ or $X^1(n_1)$ respectively. We have the following result for the dimension of $\text{co}(X^2)$, the convex hull of X^2 .

Proposition 8. *The dimension of $\text{co}(X^2)$ is given by:*

$$\dim(\text{co}(X^2)) = \begin{cases} \dim(\text{co}(X^1(n_2))) & \text{if } n_1 = 1, n_2 \geq 1 \\ \dim(\text{co}(X^1(n_1))) + 1 & \text{if } n_2 = 1, n_1 \geq 2 \\ 2 + 2n_2 & \text{if } n_1 = 2, n_2 \geq 2 \\ 2n_1 + 2n_2 & \text{o.w.} \end{cases}$$

Proof. To prove the result we consider the following four cases:

- (i.) $n_1 = 1$. The system reduces to $X^1(n_2)$ since the arc outgoing from node 1 to node 2 has to be chosen and it serves as an exogenous flow entering node 2. Hence the system just reduces to a single node system with positive incoming flow b and n_2 leaving arcs.
- (ii.) $n_2 = 1$. If $n_1 = 2$ we can easily check that there are only four points in X^2 all of which are affinely independent, hence $\dim(\text{co}(X^2)) = 3 = \dim(\text{co}(X^1(2))) + 1$. For $n_1 \geq 3$, first note that since $n_2 = 1$ all the feasible solutions satisfy $g_1 = d^g$, and this constraint is linearly independent of the other equality constraints, so the rank of the equality set of convex hull of X^2 is at least 3, hence the dimension of convex hull is at most $2n_1 + 1 + 3 - 3 = 2n_1 + 1$. But we can find $2n_1 + 2$ affinely independent points in X^2 , $2n_1 + 1$ are the same as defined in Proposition 1 (along with y, g and d^g). The last point comes from the fact that when $x_1 = 0$ we can have both $y_1 = 0$ and $y_1 = 1$ and these points are affinely independent. The points are given below:

$$\begin{pmatrix}
x_1 & x_2 & \dots & x_n & f_1 & f_2 & \dots & f_n & d^f & y_1 & g_1 & d^g \\
1 & 0 & \dots & 0 & b & 0 & \dots & 0 & b & 1 & b & b \\
0 & 1 & \dots & 0 & 0 & b & \dots & 0 & b & 1 & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \dots & 1 & 0 & 0 & \dots & b & b & 1 & 0 & 0 \\
0 & 1 & \dots & 1 & 0 & \frac{b}{n-1} & \dots & \frac{b}{n-1} & \frac{b}{n-1} & 1 & 0 & 0 \\
1 & 0 & \dots & 1 & \frac{b}{n-1} & 0 & \dots & \frac{b}{n-1} & \frac{b}{n-1} & 1 & \frac{b}{n-1} & \frac{b}{n-1} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
1 & 1 & \dots & 0 & \frac{b}{n-1} & \frac{b}{n-1} & \dots & 0 & \frac{b}{n-1} & 1 & \frac{b}{n-1} & \frac{b}{n-1} \\
1 & 1 & \dots & 1 & \frac{b}{n} & \frac{b}{n} & \dots & \frac{b}{n} & \frac{b}{n} & 1 & \frac{b}{n} & \frac{b}{n} \\
\hline
0 & 1 & \dots & 0 & 0 & b & \dots & 0 & b & 0 & 0 & 0
\end{pmatrix}$$

(iii.) $n_1 = 2, n_2 \geq 2$. We know from Proposition 1 that for the case of $n_1 = 2$, the equality set for convex hull of $X^1(n_1)$ is of rank 3 ($5 - 2$). The same equality set must be present in convex hull of X^2 , since X^2 contains all the constraints of $X^1(n_1)$. So the rank of the equality set of convex hull of X^2 must be at least $3 + 1$ (1 comes from the equality for second node). Hence the dimension of convex hull of X^2 is at most $5 + 2n_2 + 1 - 4 = 2 + 2n_2$. We

now show $3 + 2n_2$ affinely independent points in X^2 .

$$\left(\begin{array}{cccccccccccccccc} x_1 & x_2 & f_1 & f_2 & d^f & y_1 & y_2 & \dots & y_{n_2} & g_1 & g_2 & \dots & g_{n_2} & d^g \\ \hline 1 & 0 & b & 0 & b & 1 & 0 & \dots & 0 & b & 0 & \dots & 0 & b \\ 1 & 0 & b & 0 & b & 0 & 1 & \dots & 0 & 0 & b & \dots & 0 & b \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & b & 0 & b & 0 & 0 & \dots & 1 & 0 & 0 & \dots & b & b \\ 1 & 0 & b & 0 & b & 1 & 1 & \dots & 0 & \frac{b}{2} & \frac{b}{2} & \dots & 0 & \frac{b}{2} \\ \hline 1 & 1 & \frac{b}{2} & \frac{b}{2} & \frac{b}{2} & 1 & 0 & \dots & 0 & \frac{b}{2} & 0 & \dots & 0 & \frac{b}{2} \\ 1 & 1 & \frac{b}{2} & \frac{b}{2} & \frac{b}{2} & 0 & 1 & \dots & 0 & 0 & \frac{b}{2} & \dots & 0 & \frac{b}{2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \frac{b}{2} & \frac{b}{2} & \frac{b}{2} & 0 & 0 & \dots & 1 & 0 & 0 & \dots & \frac{b}{2} & \frac{b}{2} \\ 1 & 1 & \frac{b}{2} & \frac{b}{2} & \frac{b}{2} & 1 & 1 & \dots & 0 & \frac{b}{4} & \frac{b}{4} & \dots & 0 & \frac{b}{4} \\ \hline 0 & 1 & 0 & b & b & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \end{array} \right)$$

The above defined points gives us $2n_2 + 3$ points, which can be easily verified to be affinely independent.

(iv.) $n_1 \geq 3, n_2 \geq 2$. In this case we have the rank of equality set of convex hull of X^2 is at least 2 (one from each node), hence dimension of convex hull of X^2 is at most $2n_1 + 1 + 2n_2 + 1 - 2 = 2n_1 + 2n_2$. We now show $2n_1 + 2n_2 + 1$ affinely independent points in X^2 .

x_1	x_2	$\dots$	x_{n_1}	f_1	f_2	$\dots$	f_{n_1}	d^f	y_1	y_2	$\dots$	y_{n_2}	g_1	g_2	$\dots$	g_{n_2}	d^g
1	1	$\dots$	0	$\frac{b}{2}$	$\frac{b}{2}$	$\dots$	0	$\frac{b}{2}$	1	0	$\dots$	0	$\frac{b}{2}$	0	$\dots$	0	$\frac{b}{2}$
1	1	$\dots$	0	$\frac{b}{2}$	$\frac{b}{2}$	$\dots$	0	$\frac{b}{2}$	0	1	$\dots$	0	0	$\frac{b}{2}$	$\dots$	0	$\frac{b}{2}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
1	1	$\dots$	0	$\frac{b}{2}$	$\frac{b}{2}$	$\dots$	0	$\frac{b}{2}$	0	0	$\dots$	1	0	0	$\dots$	$\frac{b}{2}$	$\frac{b}{2}$
1	0	$\dots$	1	$\frac{b}{2}$	0	$\dots$	$\frac{b}{2}$	$\frac{b}{2}$	1	0	$\dots$	0	$\frac{b}{2}$	0	$\dots$	0	$\frac{b}{2}$
1	0	$\dots$	1	$\frac{b}{2}$	0	$\dots$	$\frac{b}{2}$	$\frac{b}{2}$	0	1	$\dots$	0	0	$\frac{b}{2}$	$\dots$	0	$\frac{b}{2}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
1	0	$\dots$	1	$\frac{b}{2}$	0	$\dots$	$\frac{b}{2}$	$\frac{b}{2}$	0	0	$\dots$	1	0	0	$\dots$	$\frac{b}{2}$	$\frac{b}{2}$
1	0	$\dots$	0	b	0	$\dots$	0	b	1	1	$\dots$	0	$\frac{b}{2}$	$\frac{b}{2}$	$\dots$	0	$\frac{b}{2}$
0	1	$\dots$	0	0	b	$\dots$	0	b	1	1	$\dots$	0	0	0	$\dots$	0	0
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
0	0	$\dots$	1	0	0	$\dots$	b	b	1	1	$\dots$	0	0	0	$\dots$	0	0
1	0	$\dots$	0	b	0	$\dots$	0	b	1	0	$\dots$	1	$\frac{b}{2}$	0	$\dots$	$\frac{b}{2}$	$\frac{b}{2}$
0	1	$\dots$	0	0	b	$\dots$	0	b	1	0	$\dots$	1	0	0	$\dots$	0	0
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
0	0	$\dots$	1	0	0	$\dots$	b	b	1	0	$\dots$	0	0	0	$\dots$	0	0
1	1	$\dots$	1	$\frac{b}{n_1}$	$\frac{b}{n_1}$	$\dots$	$\frac{b}{n_1}$	$\frac{b}{n_1}$	1	1	$\dots$	1	$\frac{b}{n_1 n_2}$	$\frac{b}{n_1 n_2}$	$\dots$	$\frac{b}{n_1 n_2}$	$\frac{b}{n_1 n_2}$

We have $2n_1 + 2n_2 + 1$ points above, which can be easily seen to be affinely independent.

This completes the proof. $\square$

We now present some valid inequalities for X^2 .

Proposition 9. *The following inequalities are valid for X^2 :*

$$\begin{aligned} -bx_1 + b \sum_{j \neq 1} x_j + k(k+1)d^f + 2by_i - 2(k+1)(n_2-1)g_i \\ + 2(k+1)(n_2-1)d^g \geq (2k+1)b \quad \forall i \quad \forall k \end{aligned} \quad (42)$$

$$\begin{aligned} -bx_1 + b \sum_{j \neq 1} x_j + k(k+1)d^f - 2by_i + 4(k+1)g_i \\ + 2(k+1) \sum_{j \neq i} g_j - 2(k+1)d^g \geq (2k-1)b \quad \forall i \quad \forall k \end{aligned} \quad (43)$$

$$bx_1 + by_i - 2n_1g_i - n_1 \sum_{j \neq i} g_j + n_1d^g \leq b \quad \forall i \quad (44)$$

$$bx_1 - by_i + (n_2-1)n_1g_i - (n_2-1)n_1d^g \leq 0 \quad \forall i \quad (45)$$

Proof. To prove the validity of the inequalities (42) we will consider the following four cases. Take any point in X^2 and let $\sum_j x_j = l$ and $\sum_j y_j = m$. Denote the difference between the left hand side and right hand side of an inequality by (LHS - RHS).

(i). $x_1 = 0$ and $y_i = 0$. We have (LHS - RHS) equal to

$$\frac{b}{l} (l^2 - (2k+1)l + k(k+1)) = \frac{b}{l} \left(\left(l - \frac{(2k+1)}{2} \right)^2 - \frac{1}{4} \right) \geq 0$$

for all l integer.

(ii). $x_1 = 0$ and $y_i = 1$. Since $x_1 = 0$ even though $y_i = 1$ we still have $g_i = 0$. We have

$$\text{(LHS - RHS) equal to } \frac{b}{l} \left(\left(l - \frac{(2k+1)}{2} \right)^2 - \frac{1}{4} \right) + 2b \geq 0$$

(iii). $x_1 = 1$ and $y_i = 0$. We have (LHS - RHS) equal to

$$\begin{aligned} & \left(-b + b(l-1) + k(k+1)\frac{b}{l} + 2(k+1)(n_2-1)\frac{b}{lm} - (2k+1)b \right) \\ & \geq \frac{b}{l} \left(l^2 - \frac{(2k+3)l}{2} + k^2 + 3k + 2 \right) = \frac{b}{l} \left(\left(l - \frac{(2k+3)}{2} \right)^2 - \frac{1}{4} \right) \geq 0 \end{aligned}$$

for all l integer.

(iv). $x_1 = 1$ and $y_i = 1$. We have (LHS - RHS) equal to

$$\left(-b + b(l-1) + k(k+1)\frac{b}{l} + 2b - (2k+1)b \right) = \frac{b}{l} \left(\left(l - \frac{(2k+1)}{2} \right)^2 - \frac{1}{4} \right) \geq 0$$

for all l integer.

Similarly to prove the validity of (43) we also consider four cases. Take any point in X^2 and let $\sum_j x_j = l$ and $\sum_j y_j = m$.

(i). $x_1 = 0$ and $y_i = 0$. We have (LHS - RHS) equal to

$$\frac{b}{l} (l^2 - (2k-1)l + k(k+1)) = \frac{b}{l} \left(\left(l - \frac{(2k-1)}{2} \right)^2 - \frac{1}{4} + 2k \right) \geq 0$$

for all l integer.

(ii). $x_1 = 0$ and $y_i = 1$. Since $x_1 = 0$ even though $y_i = 1$ we still have $g_i = 0$. We have

$$(\text{LHS} - \text{RHS}) \text{ equal to } (bl + k(k+1)\frac{b}{l} - 2b - (2k-1)b) = \frac{b}{l} \left(\left(l - \frac{(2k+1)}{2} \right)^2 - \frac{1}{4} \right) \geq 0.$$

(iii). $x_1 = 1$ and $y_i = 0$. We have (LHS - RHS) equal to

$$\begin{aligned} & \left(-b + b(l-1) + k(k+1)\frac{b}{l} + 2(k+1)\frac{b}{l} - 2(k+1)\frac{b}{lm} - (2k-1)b \right) \\ & \geq \frac{b}{l} \left(l^2 - \frac{(2k+1)l}{2} + k(k+1) \right) = \frac{b}{l} \left(\left(l - \frac{(2k+1)}{2} \right)^2 - \frac{1}{4} \right) \geq 0 \end{aligned}$$

for all l integer.

(iv). $x_1 = 1$ and $y_i = 1$. We have (LHS - RHS) equal to

$$\begin{aligned} & (-b + b(l-1) + k(k+1)\frac{b}{l} - 2b + 4(k+1)\frac{b}{lm} + \\ & + 2(k+1)\frac{b(m-1)}{lm} - 2(k+1)\frac{b}{lm} - (2k-1)b) \\ & = \frac{b}{l} \left(l^2 - \frac{(2k+3)l}{2} + k(k+1) + 2(k+1) \right) = \frac{b}{l} \left(\left(l - \frac{(2k+3)}{2} \right)^2 - \frac{1}{4} \right) \geq 0 \end{aligned}$$

for all l integer.

We will prove the validity of the set of inequalities (44) by considering the following four cases. Take any point in X^2 and let $\sum_j x_j = l$ and $\sum_j y_j = m$.

(i). $x_1 = 0$ and $y_i = 0$. We have the (LHS - RHS) equal to $0 - b \leq 0$.

(ii). $x_1 = 0$ and $y_i = 1$. We have (LHS - RHS) equal to $b - b \leq 0$.

(iii). $x_1 = 1$ and $y_i = 0$. We have (LHS - RHS) equal to $b - n_1 \frac{b}{l} + n_1 \frac{b}{lm} - b \leq 0$, since $m \geq 1$.

(iv). $x_1 = 1$ and $y_i = 1$. We have (LHS - RHS) equal to $b + b - 2n_1 \frac{b}{lm} - n_1 \frac{b(m-1)}{lm} + n_1 \frac{b}{lm} - b = b - (\frac{2n_1 + n_1(m-1) - n_1}{lm})b = b - \frac{n_1}{l}b \leq 0$ (since $l \leq n_1$).

Similarly, to prove the validity of (45) we consider the following four cases. Take any point in X^2 and let $\sum_j x_j = l$ and $\sum_j y_j = m$.

(i). $x_1 = 0$ and $y_i = 0$. We have left hand side equal to $0 \leq 0$.

(ii). $x_1 = 0$ and $y_i = 1$. We have left hand side equal to $-b \leq 0$.

(iii). $x_1 = 1$ and $y_i = 0$. We have left hand side equal to $b - (n_2 - 1)n_1 \frac{b}{lm} \leq 0$ ($m \leq (n_2 - 1)$ since $y_i = 0$).

(iv). $x_1 = 1$ and $y_i = 1$. We have left hand side equal to $b - b - (n_2 - 1)n_1 \frac{b}{lm} - (n_2 - 1)n_1 \frac{b}{lm} = 0 \leq 0$.

□

Separation

The set of inequalities defined in Proposition 9 are polynomial in size and hence a straightforward separation scheme could go through the list of inequalities one at a time. However we can do better by using a separation scheme similar to the one used to separate the single node inequalities (16).

2.3.3 An alternate two node system

We now consider a slight variation of the above-mentioned two node system, where we assume that there is positive incoming flow into both nodes as shown in Figure 7. Let the incoming flow into node 1 and 2 be b_1 and b_2 respectively. The rest of the variables are defined similarly as in the two node system considered in the Section 2.3.2. For this two

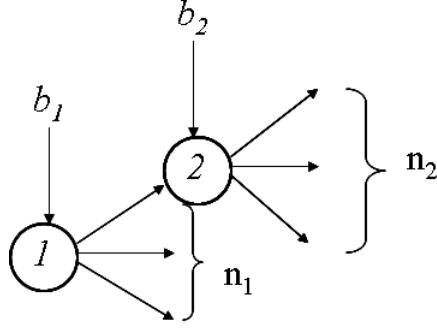


Figure 7: An alternate two node equal-split system

node system the set of feasible solutions is:

$$\begin{aligned}
X^{2+} = \Big\{ & (d^f, d^g, f, g, x, y) \in \mathbb{R}_+^2 \times \mathbb{R}_+^{n_1+n_2} \times \{0, 1\}^{n_1+n_2} : \\
& \sum_{i=1}^{n_1} f_i = b_1, \\
& \sum_{i=1}^{n_2} g_i = f_1 + b_2, \\
& f_i \leq b_1 x_i \quad i = 1, \dots, n_1, \\
& g_i \leq (b_1 + b_2) y_i \quad i = 1, \dots, n_2, \\
& f_i \leq d^f \quad i = 1, \dots, n_1, \\
& g_i \leq d^g \quad i = 1, \dots, n_2, \\
& d^f \leq f_i + b_1(1 - x_i) \quad i = 1, \dots, n_1, \\
& d^g \leq g_i + (b_2 + b_1)(1 - y_i) \quad i = 1, \dots, n_2 \Big\}
\end{aligned}$$

We have the following result for the dimension of $\text{co}(X^{2+})$, the convex hull of X^{2+} .

Proposition 10. *The dimension of $\text{co}(X^{2+})$ is given by:*

$$\dim(\text{co}(X^{2+})) = \begin{cases} \dim(\text{co}(X^1(n_2))) & \text{if } n_1 = 1, n_2 \geq 1 \\ \dim(\text{co}(X^1(n_1))) & \text{if } n_2 = 1, n_1 \geq 2 \\ 2 + 2n_2 & \text{if } n_1 = 2, n_2 \geq 2 \\ 2n_1 + 2n_2 & \text{o.w.} \end{cases}$$

Proof. We will consider the following four cases:

- (i.) $n_1 = 1$. The system reduces to $X^1(n_2)$ since the arc outgoing from node 1 to node 2 has to be chosen and it serves as an exogenous flow entering node 2. Hence the system just reduces to a single node system with positive incoming flow $b_1 + b_2$ and n_2 leaving arcs.
- (ii.) $n_2 = 1$. Since $n_2 = 1$, the outgoing flow from node 2 will always be equal to b_2 plus the flow on arc connecting node 1 to node 2. Hence the system reduces to a single node system of node 1.
- (iii.) $n_1 = 2, n_2 \geq 2$. We know from Proposition 1 that for the case of $n_1 = 2$, the rank of the equality set for convex hull of $X^1(n_1)$ is 3. So the rank of the equality set of convex hull of X^{2+} must be at least $3 + 1$ (1 comes from the equality for second node). Hence the dimension of convex hull of X^2 is at most $5 + 2n_2 + 1 - 3 - 1 = 2 + 2n_2$. We now show $3 + 2n_2$ affinely independent points in X^{2+} .

$$\left(\begin{array}{cccccccccccccccc} x_1 & x_2 & f_1 & f_2 & d^f & y_1 & y_2 & \dots & y_{n_2} & g_1 & g_2 & \dots & g_{n_2} & d^g \\ \hline 0 & 1 & 0 & b_1 & b_1 & 1 & 0 & \dots & 0 & b_2 & 0 & \dots & 0 & b_2 \\ 0 & 1 & 0 & b_1 & b_1 & 0 & 1 & \dots & 0 & 0 & b_2 & \dots & 0 & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & 0 & b_1 & b_1 & 0 & 0 & \dots & 1 & 0 & 0 & \dots & b_2 & b_2 \\ 0 & 1 & 0 & b_1 & b_1 & 1 & 1 & \dots & 0 & \frac{b_2}{2} & \frac{b_2}{2} & \dots & 0 & \frac{b_2}{2} \\ \hline 1 & 1 & \frac{b_1}{2} & \frac{b_1}{2} & \frac{b_1}{2} & 1 & 0 & \dots & 0 & \frac{b_1}{2} + b_2 & 0 & \dots & 0 & \frac{b_1}{2} + b_2 \\ 1 & 1 & \frac{b_1}{2} & \frac{b_1}{2} & \frac{b_1}{2} & 0 & 1 & \dots & 0 & 0 & \frac{b_1}{2} + b_2 & \dots & 0 & \frac{b_1}{2} + b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \frac{b_1}{2} & \frac{b_1}{2} & \frac{b_1}{2} & 0 & 0 & \dots & 1 & 0 & 0 & \dots & \frac{b_1}{2} + b_2 & \frac{b_1}{2} + b_2 \\ 1 & 1 & \frac{b_1}{2} & \frac{b_1}{2} & \frac{b_1}{2} & 1 & 1 & \dots & 0 & \frac{b_1}{4} + b_2 & \frac{b_1}{4} + b_2 & \dots & 0 & \frac{b_1}{4} + b_2 \\ \hline 0 & 1 & 0 & b & b & 1 & 1 & \dots & 1 & \frac{b_2}{n_2} & \frac{b_2}{n_2} & \dots & \frac{b_2}{n_2} & \frac{b_2}{n_2} \end{array} \right)$$

The matrix above gives us $2n_2 + 3$ points, it can be easily verified that the points are affinely independent.

- (iv.) $n_1 \geq 3, n_2 \geq 2$. In this case we have the rank of equality set of convex hull of X^2 is at least 2, hence dimension of convex hull of X^2 is at most $2n_1 + 1 + 2n_2 + 1 - 2 = 2n_1 + 2n_2$. We now show $2n_1 + 2n_2 + 1$ affinely independent points in X^{2+} .

x_1	x_2	$\dots$	x_{n_1}	f_1	f_2	$\dots$	f_{n_1}	d^f	y_1	y_2	$\dots$	y_{n_2}	g_1	g_2	$\dots$	g_{n_2}	d^g
1	1	$\dots$	0	$\frac{b_1}{2}$	$\frac{b_1}{2}$	$\dots$	0	$\frac{b_1}{2}$	1	0	$\dots$	0	$\frac{b_1}{2} + b_2$	0	$\dots$	0	$\frac{b_1}{2} + b_2$
1	1	$\dots$	0	$\frac{b_1}{2}$	$\frac{b_1}{2}$	$\dots$	0	$\frac{b_1}{2}$	0	1	$\dots$	0	0	$\frac{b_1}{2} + b_2$	$\dots$	0	$\frac{b_1}{2} + b_2$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
1	1	$\dots$	0	$\frac{b_1}{2}$	$\frac{b_1}{2}$	$\dots$	0	$\frac{b_1}{2}$	0	0	$\dots$	1	0	0	$\dots$	$\frac{b_1}{2} + b_2$	$\frac{b_1}{2} + b_2$
1	0	$\dots$	1	$\frac{b_1}{2}$	0	$\dots$	$\frac{b_1}{2}$	$\frac{b_1}{2}$	1	0	$\dots$	0	$\frac{b_1}{2} + b_2$	0	$\dots$	0	$\frac{b_1}{2} + b_2$
1	0	$\dots$	1	$\frac{b_1}{2}$	0	$\dots$	$\frac{b_1}{2}$	$\frac{b_1}{2}$	0	1	$\dots$	0	0	$\frac{b_1}{2} + b_2$	$\dots$	0	$\frac{b_1}{2} + b_2$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
1	0	$\dots$	1	$\frac{b_1}{2}$	0	$\dots$	$\frac{b_1}{2}$	$\frac{b_1}{2}$	0	0	$\dots$	1	0	0	$\dots$	$\frac{b_1}{2} + b_2$	$\frac{b_1}{2} + b_2$
1	0	$\dots$	0	b_1	0	$\dots$	0	b_1	1	1	$\dots$	0	$\frac{b_1+b_2}{2}$	$\frac{b_1+b_2}{2}$	$\dots$	0	$\frac{b_1+b_2}{2}$
0	1	$\dots$	0	0	b_1	$\dots$	0	b_1	1	1	$\dots$	0	$\frac{b_2}{2}$	$\frac{b_2}{2}$	$\dots$	0	$\frac{b_2}{2}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
0	0	$\dots$	1	0	0	$\dots$	b_1	b_1	1	1	$\dots$	0	$\frac{b_2}{2}$	$\frac{b_2}{2}$	$\dots$	0	$\frac{b_2}{2}$
1	0	$\dots$	0	b_1	0	$\dots$	0	b_1	1	0	$\dots$	1	$\frac{b_1+b_2}{2}$	0	$\dots$	$\frac{b_1+b_2}{2}$	$\frac{b_1+b_2}{2}$
0	1	$\dots$	0	0	b_1	$\dots$	0	b_1	1	0	$\dots$	1	$\frac{b_2}{2}$	0	$\dots$	$\frac{b_2}{2}$	$\frac{b_2}{2}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
0	0	$\dots$	1	0	0	$\dots$	b_1	b_1	1	0	$\dots$	0	0	$\frac{b_2}{2}$	$\dots$	$\frac{b_2}{2}$	$\frac{b_2}{2}$
1	1	$\dots$	1	$\frac{b_1}{n_1}$	$\frac{b_1}{n_1}$	$\dots$	$\frac{b_1}{n_1}$	$\frac{b_1}{n_1}$	1	1	$\dots$	1	$\frac{b_1+n_1b_2}{n_1n_2}$	$\frac{b_1+n_1b_2}{n_1n_2}$	$\dots$	$\frac{b_1+n_1b_2}{n_1n_2}$	$\frac{b_1+n_1b_2}{n_1n_2}$

We have $2n_1 + 2n_2 + 1$ points above, which can be easily seen to be affinely independent.

This completes the proof. $\square$

In the following proposition we present some valid inequalities for X^{2+}

Proposition 11. *The following inequalities are valid for X^{2+} :*

$$\begin{aligned}
& -b_1x_1 + b \sum_{j \neq 1} x_j + k(k+1)d^f + (2b_1 + 2(k+1)b_2)y_i - 2(k+1)(n_2-1)g_i \\
& + 2(k+1)(n_2-1)d^g \geq (2k+1)b_1 + 2(k+1)b_2 \quad \forall i \quad \forall k \quad (46)
\end{aligned}$$

$$-b_1x_1 + (b_1 + n_1b_2)y_i - n_1(n_2-1)g_i + n_1(n_2-1)d^g \geq n_1b_2 \quad \forall i \quad (47)$$

$$-b_1x_1 + b \sum_{j \neq 1} x_j + k(k+1)d^f + 2(k+1) \sum_j g_j \geq (2k+1)b_1 + 2(k+1)b_2 \quad \forall k \quad (48)$$

$$b_1x_1 + (n_1b_2 + b_1)y_i - 2n_1g_i - n_1 \sum_{j \neq i} g_j + n_1d^g \leq b_1 \quad \forall i \quad (49)$$

$$b_2 \sum_i y_i - n_2 \sum_i g_i + n_2d^g \leq b_2 \quad (50)$$

Proof. We will prove the validity of the set of inequalities (46) by considering the following four cases. Take any point in X^{2+} and let $\sum_j x_j = l$ and $\sum_j y_j = m$.

(i). $x_1 = 0$ and $y_i = 0$. We have (LHS - RHS) equal to

$$\begin{aligned} & b_1 l + k(k+1) \frac{b_1}{l} + 2(n_2 - 1)(k+1) \frac{b_2}{m} - (2k+1)b_1 - 2(k+1)b_2 \\ & \geq \frac{b_1}{l} \left(\left(l - \frac{2k+1}{2} \right)^2 - \frac{1}{4} \right) \geq 0 \quad (\text{since } m \leq n_2 - 1, \text{ as } y_i = 0) \end{aligned}$$

for all l integer.

(ii). $x_1 = 0$ and $y_i = 1$. We have (LHS - RHS) equal to

$$\begin{aligned} & b_1 l + k(k+1) \frac{b_1}{l} + 2b_1 + 2(k+1)b_2 - (2k+1)b_1 - 2(k+1)b_2 \\ & \geq \frac{b_1}{l} \left(\left(l - \frac{2k+1}{2} \right)^2 - \frac{1}{4} \right) + 2b_1 \geq 0 \end{aligned}$$

for all l integer.

(iii). $x_1 = 1$ and $y_i = 0$. We have (LHS - RHS) equal to

$$\begin{aligned} & -b_1 + b_1(l-1) + k(k+1) \frac{b_1}{l} + 2(k+1)(n_2-1) \left(\frac{b_1}{lm} + \frac{b_2}{m} \right) - (2k+1)b_1 - 2(k+1)b_2 \\ & \geq b_1(l-2) + k(k+1) \frac{b_1}{l} + 2(k+1) \frac{b_1}{l} - (2k+1)b_1 \quad (\text{since } m \leq n_2 - 1, \text{ as } y_i = 0) \\ & = \frac{b_1}{l} \left(\left(l - \frac{2k+3}{2} \right)^2 - \frac{1}{4} \right) \geq 0 \end{aligned}$$

for all l integer.

(iv). $x_1 = 1$ and $y_i = 1$. We have (LHS - RHS) equal to:

$$\begin{aligned} & -b_1 + b_1(l-1) + k(k+1) \frac{b_1}{l} + 2b_1 + 2(k+1)b_2 - (2k+1)b_1 - 2(k+1)b_2 \\ & = \frac{b_1}{l} \left(\left(l - \frac{2k+1}{2} \right)^2 - \frac{1}{4} \right) \geq 0 \end{aligned}$$

for all l integer.

Similarly to prove the validity of (47) we consider the following four cases. Take any point in X^{2+} and let $\sum_j x_j = l$ and $\sum_j y_j = m$.

(i). $x_1 = 0$ and $y_i = 0$. We have (LHS - RHS) equal to $n_1(n_2 - 1)\frac{b_2}{m} - n_1b_2 \geq 0$ (since $m \leq n_2 - 1$).

(ii). $x_1 = 0$ and $y_i = 1$. We have (LHS - RHS) equal to $n_1b_2 - n_1(n_2 - 1)\frac{b_2}{m} + n_1(n_2 - 1)\frac{b_2}{m} - n_1b_2 = 0$.

(iii). $x_1 = 1$ and $y_i = 0$. We have (LHS - RHS) equal to $-b_1 + n_1(n_2 - 1)\left(\frac{b_1}{lm} + \frac{b_2}{m}\right) - n_1b_2 \geq -b_1 + \frac{n_1b_1}{l}$ (since $m \leq n_2$) and is ≥ 0 (since $l \leq n_1$).

(iv). $x_1 = 1$ and $y_i = 1$. We have (LHS - RHS) equal to $-b_1 + b_1 + n_1b_2 - n_1b_2 = 0 \geq 0$.

To prove the validity of (48) we consider the following two cases. Take any point in X^{2+} and let $\sum_j x_j = l$ and $\sum_j y_j = m$.

(i). $x_1 = 0$. We have the (LHS - RHS) equal to $b_1l + k(k+1)\frac{b_1}{l} + 2(k+1)b_2 - 2(k+1)b_2 - (2k+1)b_1 = \frac{b_1}{l}\left(\left(l - \frac{2k+1}{2}\right)^2 - \frac{1}{4}\right) \geq 0$ for all l integer.

(ii). $x_1 = 1$. We have (LHS - RHS) equal to

$$-b + b(l-1) + k(k+1)\frac{b}{l} + 2(k+1)\left(\frac{b_1}{l} + b_2\right) - (2k+1)b_1 - 2(k+1)b_2$$

$$\frac{b_1}{l}\left(\left(l - \frac{2k+3}{2}\right)^2 - \frac{1}{4}\right) \geq 0 \text{ for all } l \text{ integer.}$$

To prove the validity of (49) we consider the following four cases. Take any point in X^{2+} and let $\sum_j x_j = l$ and $\sum_j y_j = m$.

(i). $x_1 = 0$ and $y_i = 0$. We have (LHS - RHS) equal to $-n_1\frac{b_2}{m} + n_1\frac{b_2}{m} - b_1 \leq 0$.

(ii). $x_1 = 0$ and $y_i = 1$. We have (LHS - RHS) equal to $n_1b_2 + b_1 - 2n_1\frac{b_2}{m} - n_1\frac{(m-1)b_2}{m} + n_1\frac{b_2}{m} - b_1 = 0$.

(iii). $x_1 = 1$ and $y_i = 0$. We have (LHS - RHS) equal to $b_1 - n_1\left(\frac{b_1}{l} + b_2\right) + n_1\left(\frac{b_1}{lm} + \frac{b_2}{m}\right) - b_1 \leq 0$ (since $m \geq 1$).

(iv). $x_1 = 1$ and $y_i = 1$. We have (LHS - RHS) equal to

$$b_1 + n_1b_2 + b_1 - 2n_1\left(\frac{b_1}{lm} + \frac{b_2}{m}\right) - n_1(m-1)\left(\frac{b_1}{lm} + \frac{b_2}{m}\right) + n_1\left(\frac{b_1}{lm} + \frac{b_2}{m}\right) - b_1$$

$$= b_1 + n_1 b_2 - n_1 \left(\frac{b_1}{l} + b_2 \right) = b_1 - n_1 \frac{b_1}{l} \leq 0 \quad (\text{since } l \leq n_1.)$$

For the proof of validity of (50) note that it is sufficient to show that the inequality is valid for the case when $x_1 = 0$, since the left hand side $(b_2 \sum_i y_i - n_2 \sum_i g_i + n_2 d^g)_{x_1=1} \leq (b_2 \sum_i y_i - n_2 \sum_i g_i + n_2 d^g)_{x_1=0}$, for same y .

Let $\sum_i y_i = m$, then (LHS - RHS) is equal to $b_2 m - n_2 b_2 + n_2 \frac{b_2}{m} - b_2 = (m-1)(b_2 - n_2 \frac{b_2}{m}) \leq 0$ (since $m \leq n_2$). $\square$

Separation

We can separate the families of valid inequalities (46)-(50) using a scheme similar to that outlined in Section 2.3.2.

2.3.4 A multiple node system

Let us now consider a set of nodes $\mathcal{P}_i = \{1, 2, \dots, P\}$ on a path to a node i as depicted in Figure 8 (assume that $b_i > 0$). For each $p = 1, \dots, P$, let n_p denote the number of outgoing arcs from node p and $b_p (\geq 0)$ denote the exogenous flow into node p .

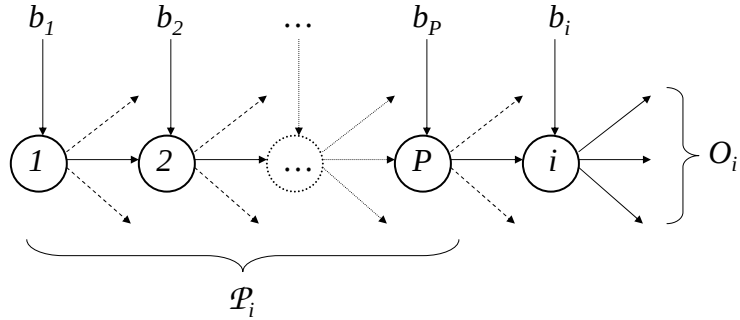


Figure 8: A multiple node equal-split system

The binary variable x_p for $p = 1, \dots, P-1$ corresponds to arcs going from p to $p+1$, and for $p = P$, corresponds to the arc from node P to i . If $x_p = 1$ for all $p = 1, \dots, P$, then a lower bound on the flow into node i is $b_i + l(\mathcal{P}_i)$ where

$$l(\mathcal{P}_i) = \frac{b_1 + n_1 b_2 + n_1 n_2 b_3 + \dots + n_1 n_2 \dots n_{P-1} b_{P-1}}{n_1 n_2 \dots n_P}.$$

Thus, the following inequality is valid

$$d_i \geq \frac{b_i + l(\mathcal{P}_i)}{z_i} + l(\mathcal{P}_i) \left(\sum_{p \in \mathcal{P}_i} x_p - |\mathcal{P}_i| \right) \quad (51)$$

where d_i is the flow on every used arc from i and z_i is the number of out-going arcs used. Now considering the inequality (51) only in (d_i, z_i) space (by considering other variables fixed), we can linearize the following set of $|O_i|$ points,

$$\left\{ b_i + l(\mathcal{P}_i) + l(\mathcal{P}_i) \left(\sum_{p \in \mathcal{P}_i} x_p - |\mathcal{P}_i| \right), 1 \right\}, \dots, \left\{ \frac{b_i + l(\mathcal{P}_i)}{|O_i|} + l(\mathcal{P}_i) \left(\sum_{p \in \mathcal{P}_i} x_p - |\mathcal{P}_i| \right), |O_i| \right\}$$

by applying the same idea as presented for the set of inequalities in (14). This leads to the following result.

Proposition 12. *The following inequalities are valid for the multiple node system defined above:*

$$\begin{aligned} k(k+1)d_i + (b_i + l(\mathcal{P}_i)) \sum_{j \in O_i} x_j - k(k+1)l(\mathcal{P}_i) \sum_{p \in \mathcal{P}_i} x_p \geq (2k+1)(b_i + l(\mathcal{P}_i)) \\ - k(k+1)l(\mathcal{P}_i)|\mathcal{P}_i| \quad k = 1, \dots, n_i - 1 \end{aligned} \quad (52)$$

where O_i is the set of outgoing arcs from i .

Separation

It is not clear how to separate the family (52) efficiently. However, we will now present a heuristic idea for separation. We will present this idea for a fixed source node i . Let $(\hat{x}, \hat{f}, \hat{d})$ be a fractional point. Then we want to check whether there exists a path $\mathcal{P}_i$ and k between 1 and n_i , such that

$$\begin{aligned} k(k+1)\hat{d}^i + (b_i + l(\mathcal{P}_i)) \sum_{j \in O_i} \hat{x}_j - k(k+1)l(\mathcal{P}_i) \sum_{p \in \mathcal{P}_i} \hat{x}_p < (2k+1)(b_i + l(\mathcal{P}_i)) \\ - k(k+1)l(\mathcal{P}_i)|\mathcal{P}_i| \end{aligned}$$

Now for the fixed source node i , we can aggregate the variables for the node that do not depend on the path $\mathcal{P}_i$, as C :

$$C = k(k+1)\hat{d}^i + b_i \sum_{j \in O_i} \hat{x}_j - (2k+1)b_i$$

If $C < 0$, we already have a violated inequality with $\mathcal{P}_i = \{\emptyset\}$. Otherwise, if $C \geq 0$, then we need to check whether there exists a path $\mathcal{P}_i$ and k , such that

$$C + l(\mathcal{P}_i) \sum_{j \in O_i} \widehat{x}_j - k(k+1)l(\mathcal{P}_i) \sum_{p \in \mathcal{P}_i} \widehat{x}_p < (2k+1)l(\mathcal{P}_i) - k(k+1)l(\mathcal{P}_i)|\mathcal{P}_i|$$

(assuming that $l(\mathcal{P}_i) > 0$, since if $l(\mathcal{P}_i) = 0$, then the inequality reduces to the single node inequality (16)),

$$\sum_{p \in \mathcal{P}_i} \widehat{x}_p > \frac{C}{k(k+1)l(\mathcal{P}_i)} + \frac{\sum_{j \in O_i} \widehat{x}_j}{k(k+1)} - \frac{(2k+1)}{k(k+1)} + |\mathcal{P}_i|$$

or once again aggregating the terms that do not depend on the path $\mathcal{P}_i$ to D ,

$$\sum_{p \in \mathcal{P}_i} \widehat{x}_p > \frac{C}{k(k+1)l(\mathcal{P}_i)} + |\mathcal{P}_i| + D. \quad (53)$$

If $D \geq 0$, then there does not exist a path that satisfies the above strict inequality. Since $\sum_{p \in \mathcal{P}_i} \widehat{x}_p \leq |\mathcal{P}_i|$, $D \geq 0$ and $C \geq 0$, hence all paths $\mathcal{P}_i$ satisfy $\sum_{p \in \mathcal{P}_i} \widehat{x}_p \leq \frac{C}{k(k+1)l(\mathcal{P}_i)} + |\mathcal{P}_i| + D$.

If $D < 0$, we need to find a path $\mathcal{P}_i$ such that (53) is satisfied. It is not clear how to find one efficiently. But to get a heuristic separation, we can try to find a path $\mathcal{P}_i$ that maximizes $\sum_{p \in \mathcal{P}_i} \widehat{x}_p$. Once we have a path, we can check whether the path satisfies (53) or not. If it does satisfy (53), we have a violated inequality; otherwise we can not say that there does not exist one. We are trying to maximize the right hand side, in hope that the path we get would satisfy (53).

Finding a path that maximizes $\sum_{p \in \mathcal{P}_i} \widehat{x}_p$ corresponds to finding a longest path to node i with weights $\widehat{x}$ on the arcs, which is a NP-hard problem. But we can use heuristic methods such as those available in the literature [6][50], to get approximate solutions.

2.3.5 Valid inequalities based on capacity

The subsystems considered so far and the corresponding valid inequalities developed ignored the capacities on the arcs and basically assumed that all binary solutions were feasible. It might be possible that, based on the capacities, certain solutions might become infeasible, e.g. an arc with capacity less than the flow coming in can take at most half of the flow

(because of the equal-split restriction). In this section we provide valid inequalities for the multi-commodity flow problem with equal-split restriction, which use capacities of the arcs to cut off infeasible solutions.

Consider the single node system studied in Section 2.3.1 and let u_i be the capacity on each arc. Without loss of generality we may assume that $u_1 \geq u_2 \geq \dots \geq u_n$. We have the following set of valid inequalities for X^1 :

Proposition 13. *Let M_i be the largest $k \in \{1, \dots, n\}$ such that $u_i < \frac{b}{k}$, or 0 if $u_i \geq b$ for all i , i.e. $M_i = \min \left\{ \lfloor \frac{b}{u_i} \rfloor, \lceil \frac{b}{u_i} \rceil - 1 \right\}$, then the following inequalities are valid for X^1 :*

$$f_i \leq \frac{b}{(M_i + 1)} x_i \quad \forall i \in \{1, \dots, n\} \quad (54)$$

Proof. Since $f_i \in \{0, b, \frac{b}{2}, \dots, \frac{b}{n}\}$ and $\frac{b}{M_i+1} \leq u_i < \frac{b}{M_i}$, we know that $f_i \leq \frac{b}{(M_i+1)}$ if arc i is chosen. $\square$

Proposition 14. *If $u_1 \geq b$ and $u_2 < b$ then following inequalities are valid for X^1 :*

$$bx_1 + 2 \sum_{j=2}^n f_j + 2d \leq 3b \quad (55)$$

Proof. To prove the validity of (55), consider a point in X^1 and let $\sum_j x_j = k, k \geq 1$. We consider the following two cases:

- (i). $x_1 = 1$. We have the (LHS-RHS) equal to $b + 2(k-1)\frac{b}{k} + 2\frac{b}{k} - 3b = 0 \leq 0$.
- (ii). $x_1 = 0$. We have the (LHS-RHS) equal to $2(k)\frac{b}{k} + 2\frac{b}{k} - 3b = -b + 2\frac{b}{k} \leq 0$ since $k \geq 2$ (only arc 1 can take the whole flow b).

$\square$

Proposition 15. *Let $M_i = \min \left\{ \lfloor \frac{b}{u_i} \rfloor, \lceil \frac{b}{u_i} \rceil - 1 \right\}$, and $\frac{b}{n} \leq u_n < \frac{b}{n-1}$ then following inequalities are valid for X^1 :*

$$bx_i + (n-2) \sum_{j \neq i}^n f_j + (n-2)d \geq (n-1)b \quad \forall i \neq n \quad (56)$$

$$-bx_j + (M_j + 1)f_j + (n - M_j - 1)f_n \leq 0 \quad \forall j \neq n \quad (57)$$

Proof. To prove the validity of (56), consider a point in X^1 and let $\sum_j x_j = k, k \geq 1$. We consider the following two cases:

- (i). $x_i = 1$. We have the (LHS-RHS) equal to $b + (n-2)(k-1)\frac{b}{k} + (n-2)\frac{b}{k} - (n-1)b = 0 \geq 0$.
- (ii). $x_i = 0$. Because of the capacity restriction on arc n , it can only be chosen if all n arcs are chosen, so when $x_i = 0 \implies x_n = 0$. We have the (LHS-RHS) equal to $(n-2)(k)\frac{b}{k} + (n-2)\frac{b}{k} - (n-1)b = -b + (n-2)\frac{b}{k} \geq 0$ since $k \leq (n-2)$.

To prove the validity of (57), consider a point in X^1 and let $\sum_i x_i = k, k \geq 1$. We consider the following cases:

- (i). $x_j = 0$. We have that $f_j = 0$ and $x_n = 0$ which implies $f_n = 0$. The inequality is valid.
- (ii). $x_j = 1$ and $x_n = 0$. The inequality reduces to (54) that we already shown to be valid.
- (iii). $x_j = 1$ and $x_n = 1$. Since x_n can only be one if all the x are 1, it implies $f_i = \frac{b}{n} \forall i$. Hence (LHS-RHS) is equal to $-b + (M_j + 1)\frac{b}{n} + (n - M_j - 1)\frac{b}{n} = 0 \leq 0$.

□

Proposition 16. *If $\frac{b}{n-1} \leq u_n < \frac{b}{n-2}$ then following inequalities are valid for X^1 :*

$$f_j - f_n \geq \frac{b}{n-1}(x_j - x_n) \quad \forall j \quad (58)$$

Proof. Since $\frac{b}{n-1} \leq u_n < \frac{b}{n-2}$, we know that $f_n \leq \frac{b}{n-1}x_n$. So, if $x_n = 0$, the inequality (58) becomes: $f_j \geq \frac{b}{n-1}x_j$, which is valid since $\sum_j x_j \leq n-1$ as $x_n = 0$. When $x_n = 1$ and $x_j = 1$, we have that $f_j \geq f_n$, which is true as $f_j = f_n$. It is also valid for the case when $x_n = 1$ and $x_j = 0$ as $f_n \leq \frac{b}{n-1}x_n$. □

All these valid inequalities can be derived, for each instance, in the preprocessing stage. Before the start of the branch-and-bound algorithm, these cutting planes can be generated if the capacities satisfy the requirements.

2.4 *A branch-and-cut implementation*

We incorporated the cuts derived in the previous section within a branch-and-cut algorithm to solve the multi-commodity network flow problem with equal-split restrictions. We used CPLEX 9.0 callable libraries to implement the branch-and-cut algorithm. Cuts for which we had efficient separation schemes were added using the cut callback function as they were violated. Computations were done on an Intel Pentium 4 Zeon machine with 2.4 GHz speed and 2.0 GB RAM running Linux kernel 2.4.27.

2.4.1 Test Problems

We tested our branch-and-cut implementation on some randomly generated network flow instances. To generate random networks we first fixed the number of nodes and number of demand pairs, and then an arc between two nodes was randomly added with a probability of 0.5. If, after this procedure, the random graph obtained was not connected, more arcs were added to ensure connectedness. Capacities, cost and demands were also randomly generated. The problems are named according to the number of nodes and demand pairs, e.g. n25d10, denotes the problem instance on 25 nodes and 10 source destination demand pairs. Also some dense graphs were generated with probability of an arc chosen to be in the random graph was 0.7. These test instances are denoted by adding a "D" in front of the name, e.g. Dn15d15, is a test instance of dense graph with 15 nodes and 15 demand pairs.

2.4.2 Cuts

We identified single node subsystems (X^1) and added the corresponding valid inequalities. It is easily seen that the cuts (16) are a special case of (52) when the path length is zero (i.e., $\mathcal{P} = \emptyset$). There are exponentially many cuts of type (52) and we do not have an efficient exact separation scheme for this family. Also the cuts become weak in general, as the length of the path $\mathcal{P}$ increases. So in our branch-and-cut procedure we use only the cuts for which the path length $|\mathcal{P}| \leq 2$ and these cuts can be added directly to the CPLEX cutpool. For the family of inequalities (22), we observed after doing initial computational experiments that the inequalities become weaker as the size of set S increases, so we use only the valid

inequalities for $|S| = 1, 2$. The family of valid inequalities (33) are facet defining for only $|S| = 1$ (Proposition 7), so these are the only ones that are used in our branch-and-cut procedure.

We also identified the two node subsystem, as shown in Figure 6, in our equal-split multi-commodity network flow problem and used the cuts (42),(43),(44) and (45). Similarly, we identified subsystems as shown in Figure 7 and added the valid inequalities (46),(47),(48),(49) and (50). We also added cutting planes based on capacity as discussed in Section 2.3.5. All the family of valid inequalities are polynomial in size and were added to the CPLEX cut pool, if the corresponding condition on the capacity was satisfied.

2.4.3 Computational Results

In Table 2.4.3, we demonstrate the effectiveness of our cuts by comparing our branch-and-cut procedure with default CPLEX. The first column is the name of the problem. The second and the third column present time (in seconds) and node counts taken to prove optimality when the proposed MILP formulation was given to CPLEX. The fourth and fifth column present time and node count results for our branch-and-cut method. If for a problem optimality is not within the allotted time of 14400 seconds, i.e. 4 hours, the entry in the time and the node count column is the percentage optimality gap at the end of the run.

When the cuts were used within a branch-and-cut framework, we were able to solve all but two of the problems. It is evident from Table 2.4.3 that both node count and time significantly decreases when our cuts are added. Many of the instances were solved at the root node when the cuts were used. We can see from Table 2.4.3 that time spent per node increases in our approach but that is fully expected as the size of the node linear program increases as more cuts are added. If we consider at the average of the total time spent (not including the times when the problem was not solved) and the average number of nodes required, to solve the instances, the times decrease by a factor of 30 and the node counts decrease by a factor of 90 with our solution strategy. These results clearly demonstrate the effectiveness of the cutting planes derived for the equal-split multi-commodity network flow

Table 1: Node counts and time for instances of multi-commodity network flow problems

<i>Prob</i>	CPLEX		CPLEX + CUTS	
	<i>Time(s)</i>	<i>Node Count</i>	<i>Time(s)</i>	<i>Node Count</i>
Dn15d15	181.7	13343	1.3	13
Dn15d20	2.83	62	0.2	0
Dn20d15	165.49	7203	0.5	0
Dn20d20	43.6	1049	0.4	0
Dn30d15	163.3	1557	0.9	0
Dn30d20	0.27%	0.27%	1.9	0
Dn30d30	11556.1	82455	7.6	10
n25d10	1.34	6	0.1	1
n25d15	21.81	679	2.5	35
n25d20	84.65	2310	13.12	204
n25d30	853.94	20927	18.93	333
n25d35	3.56%	3.56%	1.42%	1.42%
n25d50	1.21%	1.21%	0.18%	0.18%
n30d20	7.03	64	0.69	0
n30d25	1.17%	1.17%	280.5	1531
n30d30	4916.3	177357	25.5	279
n30d35	1137.8	36355	188.74	1040
n30d40	5927.6	150102	356.2	1282
n35d15	16.34	79	0.41	0
n35d20	330.92	6979	1.72	2
n35d25	355.87	5009	0.77	0
n35d30	603.18	6342	63.44	668
n40d25	1849.12	16686	9.56	29
n40d30	0.83%	0.83%	19.05	84
n40d40	10739.3	129895	530.8	3237
n40d50	0.85%	0.85%	52.27	136
Average*	1947.9	32923	61.2	357

*Computed over problems that were solved.

problem.

2.4.4 Relative Strength of Cuts

The computational experiments described in Table 2.4.3 were done by adding all the available cutting plane families in a branch-and-cut method. So the results provided in Table 2.4.3 demonstrate the combined effect of the cutting planes and do not show which cutting planes are more helpful than the others. In order to investigate the relative effectiveness of the different cutting plane families we performed additional computational experiments by dividing our cut families into three different categories:

- I. All families for the single node system. This includes the cut families (16), (22) and (33).
- II. All families for the multiple node systems, this includes both two node systems and multiple node systems. This includes the cut families (42)-(45), (46)-(50) and (52).
- III. All capacity based cuts. This includes the cut families (54)-(58).

We tested the set of random problem instances shown in Table 2.4.3 on all the possible combinations of these three sets (I), (II) and (III), namely:

- 1. Default CPLEX. None of the cutting planes were added and the default CPLEX's branch-and-bound method was used. Same results as column 2 and 3 from Table 2.4.3
- 2. Cutting planes only from set (I) were added in a branch-and-cut fashion.
- 3. Cutting planes only from set (II) were added in a branch-and-cut fashion.
- 4. Cutting planes only from set (III) were added in a branch-and-cut fashion.
- 5. Cutting planes from both of the sets (I) and (II) were added in a branch-and-cut fashion.
- 6. Cutting planes from both of the sets (I) and (II) were added in a branch-and-cut fashion.
- 7. Cutting planes from both of the sets (II) and (III) were added in a branch-and-cut fashion.
- 8. All the cutting planes were added, these are the same results as presented in the Table 2.4.3.

Table 2 and 3 present the computation time and node count results for all the above defined cases. We observe that the default branch-and-bound method of CPLEX, with no cutting planes added, performed worse than, as compared with all the different combinations

of all cutting planes families. We can observe that all the cut families are important in getting the node counts as well as computation time down. Cut sets (I) and (II) are important since on average the computation times and node counts decrease by a factor of 2.5 and 6 when these sets were added. But the most important of the three sets were type (III), which had the most impact both in times and node counts. If we consider the best combination then adding all the families (I), (II) and (III) together and adding (I) and (III) together, led to the most improvement in both computational time as well as nodes.

We can also see similar results from performance profile plots [28] shown in Figures 9 and 10. Figure 9 uses solve time and Figure 10 uses node counts as the performance metric. For a given x on the horizontal axis and a given method, the value y plotted in the performance profile indicates the fraction y of the instances were solved within x times the length of time required by the fastest method. The intercepts of the plot with the vertical axis to the left side indicates the fraction of instances for which the given method was the fastest and the intercept on the right hand side indicates the fraction of instances that were solved to optimality in allotted time. For example, from Figure 9, we can see that adding cut families (I), (II) and (III) together solved around 55% of the instances fastest and around 90% of problems were solved to optimality in allotted time. Also from the figure we can see that the method of adding cut families (I), (II) and (III) together was able to solve 90% of the problem instances within 5 times the length of time required by the fastest solver.

Figures 11 and 12 shows the same performance plots when performance ratio is cutoff at 5. From these figures it is clear that most number of problems were solved by adding cuts of type (I) and (III) together. Cut families (I), (II) and (III) together were able to solve more problems faster than any other combination. Cut families (II) and (III) and (I) and (III) were able to solve more problems with smallest node counts than any other combination.

It is clear that the most important set of family of cuts are of type (III), which represents the family of cutting planes based on capacity. The main reason may be the fact that the cuts of type (III) derived were very specific to the data of the problem. Since they also utilize the equal-split structure of the problem, the cutting planes derived are very strong in cutting off fractional solutions. Cuts of type (I) and (II), are of generic kind (do not

depend on the instance), and are also important. They played a less important part in strengthening the linear programming relaxation of the equal-split multi-commodity flow problem.

Table 2: Time (in seconds) for instances of multi-commodity network flow problem (time limit: 4 hours)

	CPLEX		(I)	(II)	(III)	(I)+(II)	(I)+(III)	(II)+(III)	(I)+(II)+(III)
Dn15d15	181.7		222.63	330.21	0.83	426.97	0.82	1.29	1.3
Dn15d20	2.83		2.19	5.15	0.11	4.85	0.13	0.18	0.2
Dn20d15	165.49		127.68	119.06	0.3	132.15	0.29	0.45	0.5
Dn20d20	43.6		17.91	17.08	0.27	17.66	0.27	0.4	0.4
Dn30d15	163.3		159.26	160.18	0.63	146.91	0.69	1	0.9
Dn30d20	0.27%		5378.93	5754.34	1.23	6855.66	1.19	1.7	1.9
Dn30d30	11556.1		4419.43	4891.8	7.08	5343.72	7.17	11.67	7.6
n25d10	1.34		1.37	1.33	0.11	1.98	0.11	0.13	0.1
n25d15	21.81		8.71	9.93	1.91	15.4	2.49	3.86	2.5
n25d20	84.65		53.3	91.25	27.8	71.05	24.94	19.59	13.12
n25d30	853.94		433.94	511.89	29.16	432.94	35.73	30	18.93
n25d35	3.56%		2.88%	3.41%	1.09%	3.08%	1.58%	2.15%	1.42%
n25d50	1.21%		1.10%	0.97%	0.09%	1.79%	7716.52	0.19%	0.18%
n30d20	7.03		5.07	5.28	0.95	5.69	0.9	1.02	0.69
n30d25	1.17%		3963.27	5773.81	478.83	7447.53	365.2	328.75	280.5
n30d30	4916.3		575.18	678.24	44.05	1112.82	42.41	39.95	25.5
n30d35	1137.8		749.6	1515.23	276.57	1686.37	367.61	279.52	188.74
n30d40	5927.6		1767.67	5156.89	480.99	3166.98	472.87	530.48	356.2
n35d15	16.34		12.71	12.22	0.56	12.29	0.56	0.42	0.41
n35d20	330.92		168.62	175.15	2.5	154.2	2.58	1.73	1.72
n35d25	355.87		238.33	280.36	1.06	267	1.11	0.85	0.77
n35d30	603.18		538.77	436.16	181.15	491.65	103.49	68.09	63.44
n40d25	1849.12		983.59	1253.97	15.54	1514.79	15.48	15.03	9.56
n40d30	0.83%		5389.59	4304.9	30.01	7643.42	28.6	33.54	19.05
n40d40	10739.3		5426.17	5937.38	477.67	6575.56	411.59	542.96	530.8
n40d50	0.85%		0.21%	0.11%	82.04	0.05%	85.53	57.83	52.27
Average	1947.91		795.61	1079.44	77.46	1079.05	74.56	77.43	61.17

(I) -All families for single node systems. (II) -All families for multiple node systems. (III) -All capacity based cuts.

Table 3: Node Counts for instances of multi-commodity network flow problem (time limit: 4 hours)

	CPLEX	(I)	(II)	(III)	(I)+(II)	(I)+(III)	(II)+(III)	(I)+(II)+(III)
Dn15d15	13343	6288	9944	13	10864	13	13	13
Dn15d20	62	67	32	0	39	0	0	0
Dn20d15	7203	2495	1664	0	1856	0	0	0
Dn20d20	1049	115	118	0	117	0	0	0
Dn30d15	1557	909	934	0	917	0	0	0
Dn30d20	0.27%	28714	33325	0	31079	0	0	0
Dn30d30	82455	25613	30320	10	21188	10	10	10
n25d10	6	4	4	0	4	0	0	1
n25d15	679	192	261	24	261	35	35	35
n25d20	2310	727	730	323	687	239	181	204
n25d30	20927	5814	4511	406	4596	375	295	333
n25d35	3.56%	2.88%	3.41%	1.09%	3.08%	1.58%	2.15%	1.42%
n25d50	1.21%	1.10%	0.97%	0.09%	1.79%	33252	0.19%	0.18%
n30d20	64	29	32	0	32	0	0	0
n30d25	1.17%	32836	47892	1909	42296	1928	1780	1531
n30d30	177357	6871	8379	327	8014	323	293	279
n30d35	36355	5405	10853	2274	12272	1813	913	1040
n30d40	150102	12318	24066	1693	10605	1618	2632	1282
n35d15	79	26	25	0	25	0	0	0
n35d20	6979	953	795	2	641	2	2	2
n35d25	5009	1391	1501	0	1397	0	0	0
n35d30	6342	4020	2653	1029	2844	636	700	668
n40d25	16686	7220	7326	28	5789	27	29	29
n40d30	0.83%	26834	21996	80	24580	81	97	84
n40d40	129895	39876	25963	2714	23547	2392	3464	3237
n40d50	0.85%	0.21%	0.11%	140	0.05%	130	162	136
Average	32923	6017	6506	442	5285	374	428	357

(I) -All families for single node systems. (II) -All families for multiple node systems. (III) -All capacity based cuts.

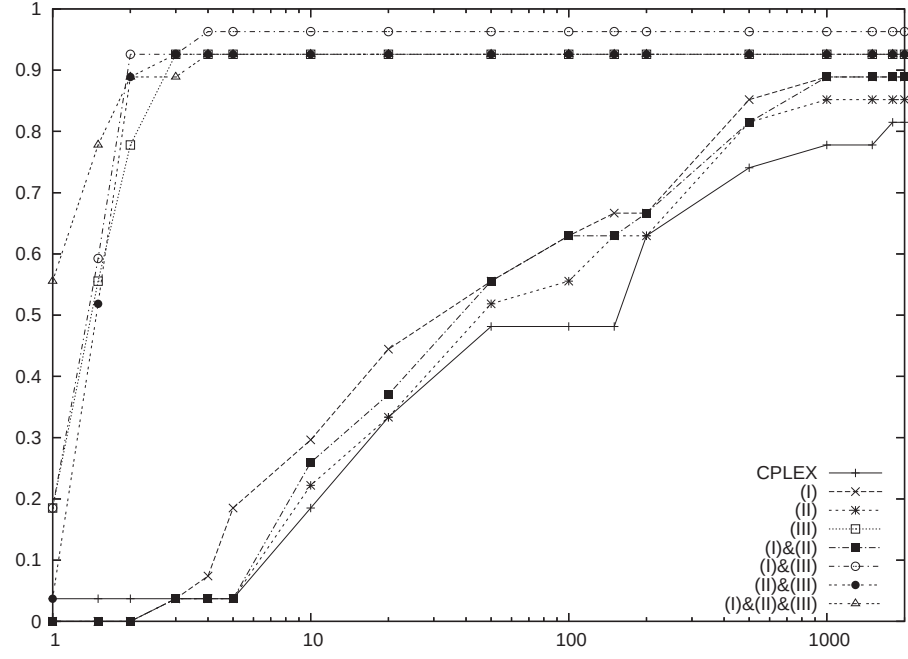


Figure 9: Performance Profile: Solution times

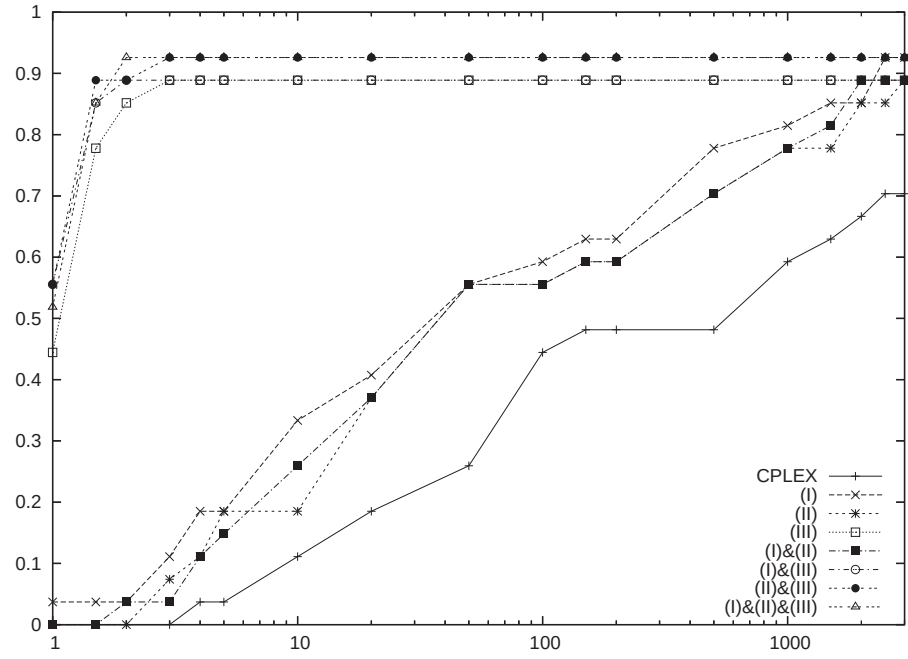


Figure 10: Performance Profile: Node Counts

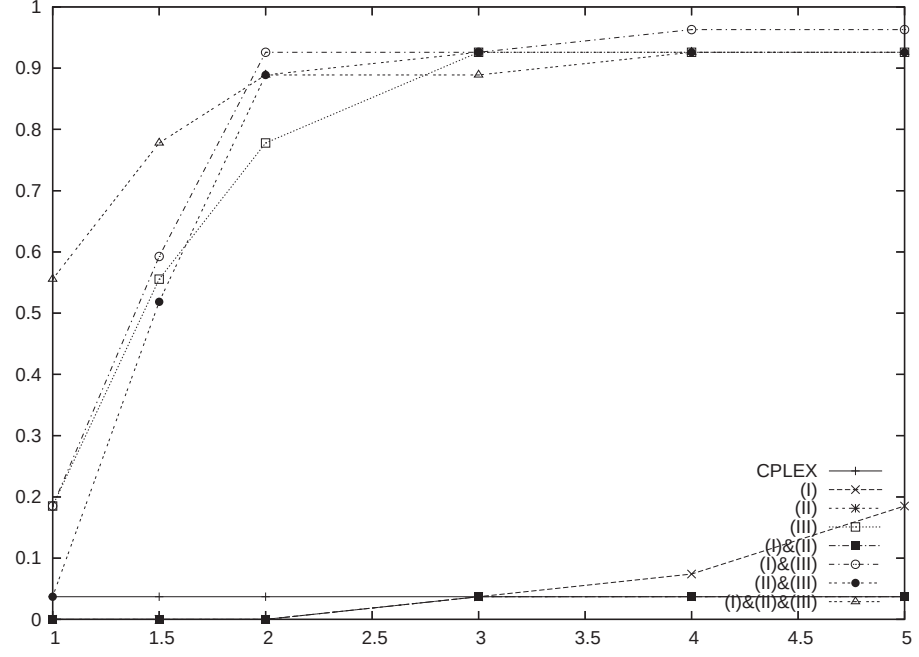


Figure 11: Performance Profile: Solution Times (performance ratio limited to 5)

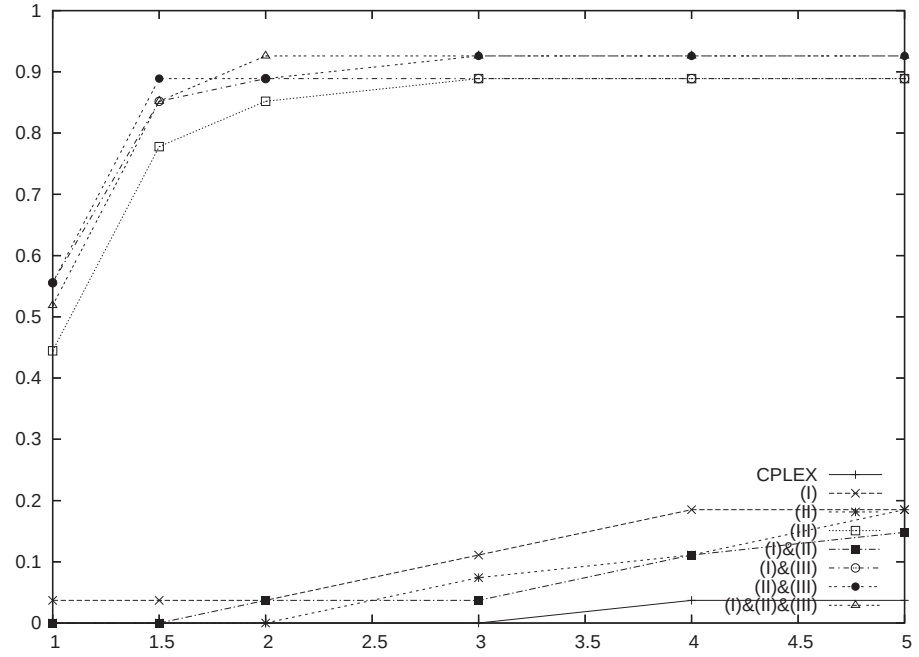


Figure 12: Performance Profile: Node Counts (performance ratio limited to 5)

2.5 Final Thoughts

We provided a detailed polyhedral study of the equal-split structure and used it to derive valid cutting planes for the equal-split multi-commodity flow problem. We also provided

fast separation schemes for most of the families of cutting planes we proposed. Computational results provided indicate very significant improvements in terms of solving instances of equal-split multi-commodity flow problem. We also provided a computational study demonstrating the relative effectiveness of the cutting planes with respect to each other. Our results indicate that the cutting planes derived based on the capacities of the arcs in the network are most important. The study on the cut family provided in Section 2.3.5 is preliminary. Further exploitation of the structure of the capacities with respect the demands may lead to additional improvements.

CHAPTER III

OSPF WEIGHT SETTING PROBLEM

3.1 Introduction

Open Shortest Path First (OSPF) is the most commonly used intra-domain routing protocol in IP networks [12, 51, 69]. Under this protocol every link in the network is assigned a weight, and based on these weights shortest paths between each source and destination are computed. The OSPF protocol then routes flow on shortest paths. If there are multiple shortest paths from some source to destination then the flow is split equally among all the outgoing arcs that are on the shortest paths. This equal-splitting of flow is usually referred to as the Equal-Cost Multipath (ECMP) principle. The weights assigned to the links completely determine the flow in the network and hence the load on each link. The OSPF weight setting problem is to determine a set of weights such that, if the flow is routed based on the OSPF protocol, some measure of network congestion is minimized.

Network congestion can be measured in a variety of ways. Let the load on an arc be defined as total flow over the arc capacity. Then we can use sum of loads over all the arcs as a measure of congestion in the network. Maximum load amongst all the arcs in the network can also be used [57]. Fortz and Thorup [36] used a piecewise linear convex cost function to measure congestion. Other choices of the congestion measure are also possible, see e.g. Awduche et. al. [8].

The OSPF protocol does not support a free distribution of flow between source and destination as in the general flow routing problem. An alternative to OSPF is the emerging Multi-Protocol Label Switching (MPLS) [62] protocol that allows the network operator to route arbitrary amounts of flow on any path. Thus MPLS can implement the optimal solution to the general flow routing problem [7][49]. This is an advantage over shortest path protocols but MPLS does not have a nice compact description of routing between source and destination. Also it is still not clear how to implement the MPLS efficiently. Before going

bankrupt Global Crossing [75] attempted to implement MPLS through a greedy heuristic. So, OSPF will still be widely used as an intra domain routing protocol until an efficient implementation of MPLS.

A variety of optimization-based approaches for the OSPF weight setting problem have been proposed. Fortz and Thorup [36] considered a version of the problem involving a piecewise linear convex cost function to measure congestion. They showed that it is NP-hard to determine an optimal set of weights, and even to approximate an optimal solution within a constant factor. They proposed a local search technique to get good solutions for the problem. Pioro et al. [57] considered the maximum load on any link in the network as the measure of congestion. They showed that the OSPF weight setting problem with this objective is NP-hard even for a single-source destination pair. They presented a mixed-integer linear programming (MILP) formulation for the problem and proposed heuristic methods, for its solution, based on local search, simulated annealing and Lagrangian relaxation. Srivastava et al. [68] used a hybrid of the two objectives as a measure of congestion, and suggested heuristic solution methods. Lin and Wang [45] and Srivastava et al. [68] proposed heuristic algorithms based on Lagrangian relaxation to determine feasible solutions for the weight setting problem. Ericsson et al. [31] presented a genetic algorithm-based heuristic. Buriol et al. [19] extended the genetic algorithm proposed in [31] to a memetic algorithm by adding a local search procedure. Harmatos presented an iterative heuristic in [40]. For more heuristic developments see [46, 52, 59, 60, 73].

Thorup et al. [18] addressed the issue of assigning OSPF weights and multiplicities to each arc, aiming to design OSPF-routed networks with minimum total weighted multiplicity (multiplicity multiplied by the arc length) needed to route the required demand and handle any single arc or router failure. Fortz and Thorup [35] studied the robust version of the OSPF weight setting problem, taking into account single link failure scenarios. They extended the heuristic proposed in [36] for the robust case. A similar problem is discussed by Nucci et al. [54]. Yuan [78] also presented a robust version of this problem that considered both normal and the failure states at the same time and proposed a local search based heuristic technique to solve it. Marbukh [47] presented a game theoretic based approach to

robust traffic engineering problem.

A restricted problem where the aim is to determine a set of weights such that these weights define unique shortest paths for each source-destination pair has also been studied in the literature. In this version of the OSPF weight setting problem there is no splitting of the flow since there is a unique path for each source-destination pair. We call this problem as the *restricted* OSPF weight setting problem. Bley and Koch [16] proposed an integer programming based approach for this restricted problem. The approach is decomposed into two stages: first stage is to find the paths on which flow is routed and the second stage is to find if there exists weights such that these paths are unique with respect to these weights. The second stage requires solving an inverse shortest path problem with the additional restriction of uniqueness of these paths. The inverse shortest path problem has been studied in literature, in context of estimation of travel times on road segments from total end to end travel times, earthquake tomography and genetic sequencing [20, 21, 42]. The uniqueness of shortest paths is not required in these studies. The special case of unique inverse shortest path where the paths are required to be shortest w.r.t the number of arcs, has been studied by Farago et al. [33]. Ben-Ameur and Gourdin [10] discussed structural properties of path sets where all paths are uniquely determined for some common arc lengths. Bley [13] studied the general unique path inverse shortest path problem and presented some inapproximability results.

For the restricted weight setting problem, Bley [14] showed that there does not exist a constant factor approximation, unless $P=NP$. Thorup and Roughan [70] proposed a local search based heuristic method that tries to avoid ties in shortest path routing. Berry et al. [11] proposed an integer programming formulation for similar unsplittable-flow shortest path routing problems. Holmberg and Yuan [41] and Prytz [58] proposed similar integer programming formulations for network design problems with discrete link capacities. See [15, 45, 61] for more discussion on the restricted OSPF weight setting problem.

There has also been some work done on the dynamic version of the OSPF weight setting problem [77], where periodic information about the demands is available, either by monitoring edge-routers or from information about service agreements. See [74, 76, 79] for some

related studies.

The existing studies on the general OSPF weight setting problem are based on developing heuristic solution methods without any quality guarantees. We propose an integer-programming-based solution strategy for this problem so as to obtain solutions along with worst-case quality guarantees. We consider the MILP formulation for the OSPF weight setting problem with a maximum load objective presented by Pioro et al. [57]. Unfortunately this MILP is extremely difficult to solve as-is by state-of-the-art solvers like CPLEX [26]. First we present a decomposition-based method for this problem. Computational results show that these kinds of methods do not perform well on the OSPF weight setting problem. Next, in order to strengthen the linear programming relaxation, we develop a family of valid inequalities and incorporate these within a branch-and-cut algorithm. We propose basic heuristic methods to start the branch-and-cut algorithm with a good solution. We provide numerical results on the performance of our solution methodology on some randomly generated test problems and some problems taken from the literature. Unlike the heuristic methods, the proposed integer programming based approach is able to provide a bound on the worst case optimality gap for the obtained solution. Moreover, the solutions generated by our approach have significantly smaller optimality gaps than those produced by the state-of-the-art MILP solver CPLEX. We also compare the heuristic methods presented with some heuristic algorithms proposed in the literature. Computational results show that our heuristic methods outperformed the ones proposed in [31][57].

3.2 Problem Formulation

In this section we present the MILP formulation of the OSPF weight setting problem from Pioro et al. [57]. The underlying IP network is represented as a directed graph $G = (V, E)$, where the nodes represent routers and arcs represent the links between them. Each arc $e \in E$ has a capacity u_e . We also assume that we have a demand matrix D , where D_{st} represents the flow to be sent from a source node $s \in V$ to a destination node $t \in V$. See [34] for information on how to estimate the demand matrix. Denote by $V_d \subseteq V$, the set of destination nodes. Congestion in the network is measured as the maximum load over

all the arcs in the network. More precisely, if l_e denotes the total flow on arc e , then the congestion is measured as the maximum of (l_e/u_e) over all arcs $e \in E$. This objective allows the flow on an arc to exceed its capacity, since there are no explicit capacity constraints, and hence any set of weight vectors is feasible for the problem. The problem is to determine weights w_e for every arc e such that if the flow is routed, as per the OSPF protocol, between each source destination pair satisfying the demand, it minimizes the congestion (measured as maximum load) in the network. Flow is split according to ECMP rule which is illustrated in Figure 13. Suppose we need to send 1 unit of flow from source s to destination t and the weights on all the arcs are 1, then all the paths $\{s, 1, 3, t\}$, $\{s, 1, 4, t\}$ and $\{s, 2, 5, t\}$ are of equal length. So according to the ECMP principle, the paths $\{s, 1, 3, t\}$, $\{s, 1, 4, t\}$ carry $\frac{1}{4}$ units of flow each, while the path $\{s, 2, 5, t\}$ carry $\frac{1}{2}$ units of flow.

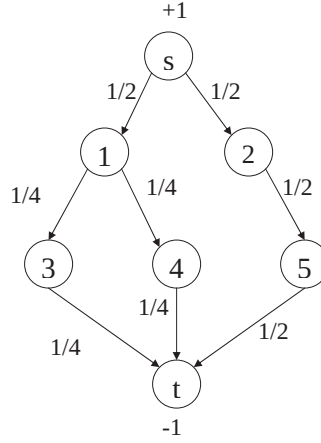


Figure 13: Equal-Cost Multipath principle

OSPF does not distinguish between the flow coming from different sources that is going to the same destination. It is thought of as a single commodity; hence we model this problem as a multi-commodity network flow problem with each commodity being flow going to a destination. The decision variables and constraints of the MILP formulation are presented next.

Decision variables:

- x_e^t binary variable denoting whether arc e is on some shortest path to destination t .
- f_e^t flow on arc e for destination t .
- w_e weight on arc e .
- d_v^t shortest path distance from node v to destination t .
- m_v^t dummy flow variables for splitting the flow from node v to destination t .
- L the maximum load in the network.

Flow conservation constraints:

$$\sum_{e \in (-, t)} f_e^t - \sum_{e \in (t, -)} f_e^t = \sum_{v \in V} D_{vt} \quad \forall t \in V_d \quad (59)$$

$$\sum_{e \in (-, v)} f_e^t - \sum_{e \in (v, -)} f_e^t = -D_{vt} \quad \forall v \in V \setminus \{t\} \quad \forall t \in V_d \quad (60)$$

$$f_e^t \leq M x_e^t \quad \forall e \quad \forall t \in V_d \quad (61)$$

Constraints (59) and (60) ensure that total inflow is equal to total outflow. Constraints (61) force the flow to be sent on only arcs that are chosen to be on the shortest path. Here M is a large number that can be set to $\sum_{v \in V} D_{vt}$.

Feasible distance label constraints:

$$d_u^t \leq d_v^t + w_e \quad \forall e = (u, v) \quad \forall t \in V_d \quad (62)$$

$$d_v^t - d_u^t + w_e \leq M(1 - x_e^t) \quad \forall e = (u, v) \quad \forall t \in V_d \quad (63)$$

$$(1 - x_e^t) \leq M(d_v^t - d_u^t + w_e) \quad \forall e = (u, v) \quad \forall t \in V_d \quad (64)$$

Constraints (62), (63) and (64) ensure that x_e^t is 1 if and only if e is on a shortest path to destination t .

Flow splitting constraints:

$$f_e^t \leq m_v^t \quad \forall e : e = (v, -) \quad \forall v, t \in V_d \quad (65)$$

$$m_v^t - f_e^t \leq M(1 - x_e^t) \quad \forall e \in (v, -) \quad \forall v, t \in V_d \quad (66)$$

Constraints (65) and (66) ensure that flow is equally split amongst all outgoing arcs on the shortest path to each destination.

Congestion constraints:

$$\sum_{t \in V_d} f_e^t \leq L c_e \quad \forall e \quad (67)$$

Constraint (67) relate the total flow on an arc to the congestion measure L .

Using the variables and constraints defined above, a MILP model for the OSPF weight setting problem is:

$$\begin{aligned} \min \quad & L \\ \text{s.t.} \quad & (59), (60), (61), (62), (63) \\ & (64), (65), (66), (67) \\ & x_e^t \in \{0, 1\}, f_e^t \geq 0, w_e \geq 0 \\ & d_v^t \geq 0, m_v^t \geq 0 \end{aligned} \quad (P)$$

For a network with $|E|$ arcs, $|V|$ nodes and k destinations, formulation (P) has $6|E||V| + |V|k + |E|$ constraints and $2(|E| + |V|)k + |E| + 1$ variables, out of which $|E|k$ are binary. As has been discussed by Pioro et al. [57], even for a small sized network, the above MILP is very difficult to solve using a state-of-art MILP solver such as CPLEX [26]. In the following section we present a decomposition based approach to solve the problem.

3.3 A Decomposition based approach

As discussed before, the mixed-integer programming formulation proposed for the OSPF weight setting problem is hard to solve even for small-sized problems. The reason is that the linear programming relaxation for the MILP formulation is weak. So even though we might have a very good feasible solution, it is impossible to tell how good the solution is. We now present a decomposition-based approach to solve the problem. We hope, even if we are not able to solve the problem to optimality, to get better lower bounds. The basic approach is discussed next.

If the constraints (62), (63) and (64) are relaxed (removed from the MILP formulation), it makes the problem easier to solve. But the binary optimal solution x might not be a feasible solution to the OSPF weight setting problem. In order to make sure that we have

a feasible solution to the OSPF weight setting problem we need to check if there exists feasible weights corresponding to the binary solution.

This can be done by solving a feasibility problem on the relaxed constraints; this is a linear programming problem (called the subproblem LP). If the subproblem LP is feasible then the solution is optimal; otherwise we can add a dual-based cut to the first stage IP and continue. Now we provide details on the procedure.

Solve the master integer program:

$$\begin{aligned}
(FS_{IP}) \quad & \min \quad L \\
& \text{s.t.} \quad (59), (60), (61), (65), (66), (67) \\
& \quad \quad \quad x_e^t \in \{0, 1\}, f_e^t \geq 0, m_v^t \geq 0
\end{aligned}$$

Let $\hat{x}_e^t$ be the binary optimal solution to (FS_{IP}) . Then the solution is feasible to the weight setting problem if and only if there exists a solution to the following set of inequalities:

$$d_u^t \leq d_v^t + w_e \quad \forall e = (u, v) \quad \forall t \in V_d \quad (68)$$

$$d_v^t - d_u^t + w_e \leq M(1 - \hat{x}_e^t) \quad \forall e = (u, v) \quad \forall t \in V_d \quad (69)$$

$$(1 - \hat{x}_e^t) \leq M(d_v^t - d_u^t + w_e) \quad \forall e = (u, v) \quad \forall t \in V_d \quad (70)$$

$$d_v^t \geq 0, w_e \geq 0,$$

or the optimal solution value of the following linear program is 0:

$$\begin{aligned}
(SS_{LP}) \quad & \min \quad \sum_{t,e} ((y_1)_e^t + (y_2)_e^t + (y_3)_e^t) \\
& \text{s.t.} \quad d_u^t - d_v^t - w_e - (y_1)_e^t \leq 0 \quad \forall e = (u, v) \quad \forall t \in V_d \quad (71)
\end{aligned}$$

$$d_v^t - d_u^t + w_e - (y_2)_e^t \leq M(1 - \hat{x}_e^t) \quad \forall e = (u, v) \quad \forall t \in V_d \quad (72)$$

$$M(d_v^t - d_u^t + w_e) + (y_3)_e^t \geq (1 - \hat{x}_e^t) \quad \forall e = (u, v) \quad \forall t \in V_d \quad (73)$$

$$d_v^t \geq 0, w_e \geq 0, (y_1)_e^t, (y_2)_e^t, (y_3)_e^t \geq 0.$$

If the value of the optimal solution is 0, then we have found an optimal solution to the OSPF weight setting problem. If the optimal solution value is > 0 , then we know that

for binary solution $\hat{x}$ there do not exist feasible weights. In order to cut off this binary solution we can add a dual-based cut as follows. Let $(\widehat{\pi}_1)_e^t$, $(\widehat{\pi}_2)_e^t$ and $(\widehat{\pi}_3)_e^t$ be the optimal dual solution to subproblem LP corresponding to constraints (71), (72) and (73). Then the following inequality is a valid inequality and it cuts off the solution $\hat{x}$:

$$\sum_{t,e} (M(1 - x_e^t)(\widehat{\pi}_2)_e^t + (1 - x_e^t)(\widehat{\pi}_3)_e^t) \leq 0 \quad (74)$$

We can add the above inequality to FS_{IP} and repeat the procedure until a time limit is reached or feasible solution is found. Along with the dual based cut, we can also add the following inequality that removes this exact binary solution $\hat{x}$.

$$\sum_{i:\hat{x}_i=1} (1 - x_i) + \sum_{i:\hat{x}_i=0} x_i \geq 1$$

The details on the complete algorithm are presented in Figure 14.

```

begin
  feas = FALSE ;
  while feas = FALSE and time limit;
  do
    Solve  $FS_{IP}$  and get optimal solution  $\hat{x}$  ;
    Solve  $SS_{LP}(\hat{x})$ ;
    if  $OPT(SS_{LP}(\hat{x})) = 0$  then
      feas = TRUE ;
      exit;
    end
    else
      Get dual soln  $\hat{\pi}$  to  $SS_{LP}(\hat{x})$ ;
      Add dual cut (74) to  $FS_{IP}$  ;
      feas = FALSE ;
    end
  end
end

```

Figure 14: A Decomposition Algorithm

3.3.1 Results

We implemented the decomposition based method as proposed above using CPLEX 9.0 [26]. Except for a few problems, we were not able to solve problems to optimality. But at every

step of the decomposition based method, we get a valid lower bound for the OSPF weight setting problem. Table 4 presents the comparison between the linear programming lower bound and the lower bound obtained through the decomposition-based approach. It also shows the number of iterations taken for reaching that lower bound. The last column shows the percentage improvement over the linear programming lower bound.

From Table 4 we can see that, except for a few instances, we were not able to solve the problem in the allotted time. For some instances we were able to improve the lower bound significantly (n10d6, n15d8, and n15d6, for example) but for others there was usually a small improvement or no improvement at all. On average the improvement was 5.19%, which is not very significant for the allotted time.

Table 4: Lower bound comparison of the decomposition method with Linear Programming lower bound (bold-face represents optimal soln found). Time limit: 600s

Prob	LP-LB	Decomp-LB	Iterations	Improvement
n5d10	1.5918	1.7019	93	6.47%
n10d5	0.9457	1.0000	44	5.43%
n10d6	0.3455	0.3971	124	13.00%
n10d11	1.2377	1.2377	125	0.00%
n10d12	1.5847	1.6269	47	2.59%
n10d15	0.8503	0.8840	2	3.81%
n15d5	0.5970	0.5971	16	0.01%
n15d6	0.5316	0.5906	218	10.00%
n15d7	0.3870	0.3871	2	0.03%
n15d8	0.4578	0.6023	237	24.00%
n15d9	0.5417	0.5417	130	0.00%
n15d10	0.3695	0.4061	2	9.00%
graph1	2.0701	2.1579	268	4.07%
caldata	0.6597	0.7060	494	6.56%
Pioro7	1.0000	1.0111	20	1.10%
Pioro12w1	1.0000	1.0056	35	0.56%
Pioro12w2	0.7802	0.7926	2	1.56%
Average Improvement	-	-	-	5.19%

3.3.1.1 Implementation within branch-and-bound

The decomposition-based method discussed above is a two stage method where an integer program (first stage) and a linear program (second stage) need to be solved at every iteration. Instead of solving a series of integer programs, we can implement both the stages

within a single branch-and-bound method.

We start with the branch-and-bound method for solving the $FSIP$, and if at a branch-and-bound tree node we find a feasible solution to $FSIP$ (which might not be feasible to relaxed constraints), we check whether it satisfies the relaxed constraints (this is done by solving the second stage LP (SS_{LP})). If the solution is feasible, accept the solution, and continue as usual. Otherwise, add the dual-based cut at that tree node, by solving the second stage LP. Now when this cut is added, the binary solution is no longer feasible and we can continue branching from the node. So the feasible solutions to $FSIP$ are accepted if and only if they are feasible to the whole problem; otherwise branching is continued. This approach was implemented in CPLEX[26], but without much improvement.

Lagrangian-relaxation-based decomposition methods were also implemented for the weight setting problem, but the lower bounds obtained were also not significantly better than the linear programming lower bound. It shows the need to develop valid inequalities for the OSPF weight setting problem in order to strengthen the linear programming relaxation. This is discussed in the next section.

3.4 Valid Inequalities

In Section 2.3 we considered various subsystems of the equal-split network flow problem. Since the same structure is present in the OSPF weight setting problem we can use the valid inequalities derived before for this problem as well. For the weight setting problem, for every node that has positive flow coming in, we can generate the same family of cutting planes.

Let l_v^t be a lower bound on the flow incoming to node v going to destination t ($l_v^t > 0$ for all source nodes v which have flow going to destination t) and let O_v denote the set of outgoing arcs out of node v . Then we can obtain the following set of inequalities from the family (16)

$$k(k+1)m_v^t + l_v^t \sum_{e \in O_v} x_e^t \geq (2k+1)l_v^t \quad k = 1, \dots, |O_v| - 1, \quad \forall v, \quad \forall t \in V_d \quad (75)$$

Similarly, we look for single node systems and obtain valid inequalities (22) and (33).

Also we can obtain two node and multi-node subsystems and obtain valid inequalities (42), (43), (44), (45), (46), (47), (48), (49), (52), like before.

3.4.1 Specific Cuts

We believe that one of the reasons why the linear programming relaxation for the OSPF weight setting problem is so weak is that there are no explicit constraints on the flow that can be routed on each arc. Our objective is to minimize the maximum load and since there are no constraints on capacity this load can be greater than one and no matter what the demand matrix looks like this problem is always feasible. But if we do have a good feasible solution, say with load L^* , then we know that every optimal solution would satisfy the capacity constraints on each arc when capacity is set to $u_e L^*$. Hence if we start the branch-and-cut algorithm with a good starting solution then we can add the following capacity constraints for flow on each arc:

$$\sum_{t \in V_d} f_e^t \leq u_e L^* \quad \forall e \in E \quad (76)$$

Consider a single source node with incoming flow b and let u_{MAX} be the capacity of the arc with the largest capacity among the outgoing arcs from the source node. Let x_i denote the binary variables on outgoing arcs. Then we know, given a feasible solution with load L^* , that the following inequality is valid:

$$\sum_i x_i \geq \left\lceil \frac{b}{u_{MAX} L^*} \right\rceil. \quad (77)$$

The above valid inequality (77) can be strengthened by considering the arc with the second largest capacity, say u_{MAX1} , as follows:

$$\sum_i x_i \geq 1 + \max \left\{ \left\lceil \frac{b - u_{MAX} L^*}{u_{MAX1} L^*} \right\rceil, 0 \right\}. \quad (78)$$

Now consider a node that is not a source node, i.e. there is no exogenous flow into this node. Let variable G denote the sum of the flows coming into the node, i.e. $G = \sum \{f_e :$

e incident with the node}. Then we generate similar cuts to (77):

$$\sum_i x_i \geq \frac{G}{u_{MAX} L^*}. \quad (79)$$

We can not do the rounding as before because G is now a variable, rather than data.

Similarly we can get the following weak valid inequality:

$$\sum_i x_i \geq 1 + \left(\frac{G - u_{MAX} L^*}{u_{MAX} L^*} \right). \quad (80)$$

Now consider a general node in the network. Let $x_i : i = 1, \dots, n$ denote the binary variables for arcs emanating from this node and $y_j : j = 1, \dots, m$ denote the corresponding binary variables for incoming arcs to this node.

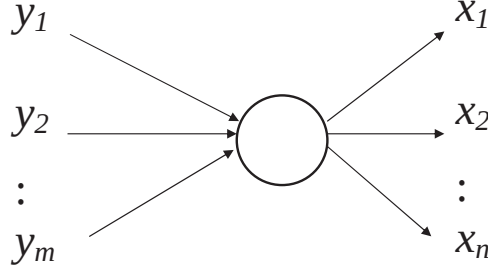


Figure 15: A general node

Then we know that if one of the incoming arcs is chosen, then there is a positive flow coming in and hence at least one of the outgoing arcs has to be chosen as well. This leads to the following set of valid inequalities:

$$\sum_{i=1}^n x_i \geq y_j \quad \forall j = 1, \dots, m. \quad (81)$$

Note that these inequalities are not valid for destination nodes as there is no flow going out from the destination nodes.

Considering in opposite direction, where if one of the outgoing arcs is chosen, then there has to be positive flow coming in and hence at least one of the incoming arcs has to be chosen as well.

$$\sum_{i=1}^m y_i \geq x_j \quad \forall j = 1, \dots, n \quad (82)$$

for all transshipment nodes, i.e. nodes without positive incoming exogenous flow.

3.5 *A branch-and-cut implementation*

We incorporated the cuts derived in the section 3.4 within a branch-and-cut algorithm to solve the OSPF weight setting problem. We used CPLEX 9.0 callable libraries to implement the branch-and-cut algorithm. Cuts were added using the cut callback function as they were violated. Computations were done on an Intel Pentium 4 Zeon machine with 2.4 GHz speed and 2.0 GB RAM running Linux kernel 2.4.27.

3.5.1 Test Problems

We experimented using some problems taken from the literature and some randomly generated problems. To generate random networks we first fixed the number of nodes and number of demand pairs, and then an arc between two nodes was randomly added with a probability of 0.5. If the random graph obtained was not connected, more arcs were added to ensure connectedness. Capacities and demands were also randomly generated. Table 5 provides the number of nodes, arcs, demand pairs and number of binary variables in the MILP models for the problems considered. The problem names starting with ‘n’ are randomly generated. The problems ‘Pioro7,’ ‘Pioro12w1’ and ‘Pioro12w2’ are taken from [57]. The problem ‘snh’ is obtained from [64]. The remaining problems, i.e., ‘graph1’ and ‘cal-data’ are proprietary telecommunication networks.

3.5.2 Cuts

Similar to the multi-commodity flow problem with equal-split we identified single node and two node subsystems and added the corresponding cutting planes. The valid inequalities that could not be separated efficiently were added to the CPLEX cut pool.

Our objective in the OSPF weight setting problem is to minimize the maximum load and there are no explicit capacity constraints on the flow. But if we have a good starting solution say L^* , we know that in any optimal solution, flow on any arc e is going to be at most $u_e L^*$ and this gives a bound on the flow on an arc. We can use this bound as explicit capacity and use capacity-based cuts of type (54),(55),(56) and (57). But the cuts generated

Table 5: Size of the problems

Prob	Nodes	Arcs	Demand pairs	Binary Vars
n5d10	5	12	10	60
n10d5	10	40	5	200
n10d6	10	48	6	288
n10d11	10	42	11	336
n10d12	10	38	12	266
n10d15	10	42	15	378
n15d5	15	98	5	392
n15d6	15	82	6	410
n15d7	15	96	7	672
n15d8	15	84	8	504
n15d9	15	88	9	792
n15d10	15	98	10	784
graph1	10	28	10	168
caldata	14	36	33	504
snh	11	46	9	322
Pioro7	7	24	21	144
Pioro12w1	12	36	66	396
Pioro12w2	12	36	66	396

from this procedure are going to be effective only if we have a very good starting solution. This emphasizes the need to have heuristic methods to get good starting solutions.

Also, based on a good starting solution we can add the valid inequalities (77), (78),(79),(80), (81),(82) as discussed in Section 3.4.1. These cuts would be strong only if we have a good starting solution L^* .

Inequalities that could not be separated efficiently were added directly to the CPLEX cut pool. The rest of them were added, using the cut callback function of CPLEX, when they were violated. At any given fractional node all violated valid inequalities are added (through cut callback).

3.5.3 Heuristics

We implemented some basic heuristic methods to start CPLEX with a good initial solution.

3.5.3.1 Local search

For the OSPF weight setting problem with the objective of minimizing the maximum load, any set of weights on the arcs is feasible (since there are no explicit restrictions on the capacity). So we implemented a weight-adjustment heuristic in which we start with a set of weights (all arc weights could be set to 1 or some other random number). After routing the flow based on these weights following the OSPF protocol, we adjust the weights, increasing weights for the arcs with bigger load and decreasing for arcs with smaller load, in hope of decreasing the maximum load in the network. This procedure is repeated a fixed number of times and the best solution obtained over all the iterations is stored. The exact steps are shown in Figure 16.

```

begin
  Initialize (w_old) ;
   $L_{best} := \infty$ ;
  for  $i = 0$  to  $max\_count$  do
    Route_Flow(w_old);
     $L = \text{Compute\_Load}(w\_old)$ ;
     $L_{max} = \max\{L(e) : e \in E\}$ ;
    if  $L_{max} < L_{best}$  then
       $L_{best} = L_{max}$  ;
       $w\_best = w\_old$  ;
    end
    for  $e = 1$  to  $E$  do
      if  $L(e) \geq 1$  or  $L(e) > \alpha L_{max}$  then
         $w\_new(e) = w\_old(e) + 1$ ;
      end
      else if  $w\_old(e) > 1$  then
         $w\_new(e) = w\_old(e) - 1$ ;
      end
    end
     $w\_old = w\_new$  ;
     $i := i + 1$ ;
  end
  return  $w\_best$  and  $L_{best}$  ;
end

```

Figure 16: A Local Search Heuristic

The weight adjustment algorithm is initialized with weights randomly selected from 1 to 20. The procedure is repeated for a set of different initial weights and the best solution

is recorded. For the computational results provided the parameter α is set to 0.9, and the max count is set to 5000.

3.5.3.2 Integer Programming (IP) heuristic

After experimenting with various different objectives, we realized that the problem with the objective being the sum of the loads rather than maximum deviation was much easier to minimize. And since the feasible region remains the same this IP provides a feasible solution to the problem with the objective of minimizing the maximum deviation. So in the heuristic stage we solved an IP that minimized the sum of loads on each arc and in order to get better starting solution, we also added the maximum load variable (with certain penalty) to our objective. We changed the objective from $\min L_{max}$ to:

$$\min \lambda L_{max} + \sum_{e \in E} L(e)$$

keeping the constraints the same, where λ is the penalty on the maximum load variable.

The more we increased the penalty on the maximum load variable λ , the harder the problem became to solve. If the penalty on the maximum load variable is very large then the problem basically reduces to our original problem. We did some computational experiments to determine the best penalty. Since we did not want to spend too much time on the heuristic IP, we put a time limit of 100 seconds on this procedure and collected the best solution obtained. We also kept track of the best solution obtained in terms of the maximum load amongst all the feasible solutions obtained.

We got the best solution from both the heuristics and started our weight setting problem with the better one. It was observed that the IP heuristic outperformed the local search heuristic most of the time.

An advantage of getting a good starting solution is that the MILP model does not have any capacity constraints on the flow that can be routed on any arc, and good heuristic solutions provide an upper bound on the capacity. Also, this facilitates generation of general flow cover type of cuts which help in the branch-and-cut procedure.

3.5.4 Symmetry

The integer programming formulation for the weight setting problem has a lot of symmetry. Given a feasible solution with certain load, another feasible solution with the same objective can be obtained by changing one x variable from 0 to 1 and keeping the flow zero. In order to get rid of this issue we added lower bounding inequalities, which imply that if the arc is chosen then there has to be a certain positive flow on that arc. These are easy to obtain for the arcs emanating from the nodes with positive exogenous incoming flow. For the simple single node system we can add the following inequalities $f_i \geq \frac{b_i}{n}x_i$. For the other arcs we find a path from the closest source node and add an inequality similar to the one obtained for the multiple node system. This does not completely remove the symmetry problem but adding these inequalities helps in the computations.

3.6 Computational Results

In Table 6 we provide the optimality gap of the best feasible solution returned after 800 seconds. The reason we chose to stop our computations after 800 seconds is because after around 800 seconds the gaps did not improve much problem. So either the problem would be solved to optimality before that time or it would be solved at all. The first column is the name of the problem. The second column provides optimality gaps for the solutions returned by the default CPLEX solver. The third column provides optimality gaps for the solutions from our branch-and-cut procedure when all the above defined cuts are added. The fourth column presents results after adding the heuristic methods and adding all the cuts obtained in the previous sections for penalty of 5 on the maximum load objective (see discussion in Section 3.5.3.2). The rest of the columns present results for the penalty of 10, 20, 50, 100, 200, 500, 1000 and 10000 respectively.

Table 6: Percentage optimality gap for instances of OSPF problem (time limit: 800 secs).

Prob	CPLEX	Cuts	Heuristic + Cuts									
			P(5)	P(10)	P(20)	P(50)	P(100)	P(200)	P(500)	P(1000)	P(10000)	
n5d10	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
n10d5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
n10d6	22.06%	6.00%	10.02%	7.35%	3.15%	3.15%	6.00%	6.00%	6.00%	6.00%	4.64%	
n10d11	13.01%	3.49%	1.18%	1.18%	0.40%	0.15%	0.15%	0.15%	0.15%	0.15%	0.12%	
n10d12	19.63%	12.89%	12.89%	10.04%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
n10d15	39.71%	29.15%	32.14%	19.92%	18.30%	18.30%	18.30%	18.28%	18.28%	18.28%	18.28%	
n15d5	8.15%	23.37%	16.42%	4.71%	0.02%	0.20%	0.45%	0.20%	0.56%	0.56%	0.20%	
n15d6	44.74%	33.33%	16.82%	7.89%	0.00%	0.00%	0.00%	11.22%	3.16%	0.00%	9.23%	
n15d7	79.83%	35.36%	30.87%	24.45%	7.59%	3.39%	7.25%	2.61%	3.03%	5.14%	7.25%	
n15d8	54.07%	36.72%	33.47%	8.14%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
n15d9	53.28%	8.33%	8.33%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
n15d10	76.54%	61.23%	60.88%	38.54%	29.83%	29.83%	29.83%	29.83%	29.83%	29.83%	29.83%	
graph1	12.76%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
caldata	-	4.84%	4.84%	4.83%	2.82%	0.00%	4.84%	0.00%	2.82%	4.84%	0.00%	
snh	-	35.18%	20.49%	19.92%	19.02%	19.92%	19.02%	19.02%	19.92%	19.02%	19.02%	
Pioro7	5.17%	4.36%	0.00%	0.00%	0.00%	3.52%	3.53%	3.53%	3.53%	3.48%	3.48%	
Pioro12w1	-	6.32%	6.32%	6.32%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
Pioro12w2	98.36%	15.58%	12.25%	10.88%	5.11%	4.21%	4.21%	4.21%	4.21%	4.21%	4.21%	
Average Gap*	35.15%	17.56%	14.83%	9.12%	4.79%	4.59%	5.20%	5.28%	5.08%	5.08%	5.35%	

*Computed over the 14 problems for which CPLEX found a feasible solution.

CPLEX: Default branch-and-bound

Cuts: Branch-and-bound with cutting planes derived

P(n): Cuts + Heuristic with penalty n .

As is evident from Table 6 the OSPF weight setting MILP is quite difficult to solve as-is. Within the allotted 800 second time limit, the default CPLEX solver produced solutions with an average optimality gap of 35.7%, and for three of the problems, it was not even able to find a feasible solution. When cuts are used in the branch-and-bound procedure we see that there is a significant reduction in the optimality gaps for the problems considered as seen in column 3, where the average optimality gap reduced by half. When the proposed branch-and-cut scheme is used with a heuristic hot start it produced feasible solutions with optimality gaps significantly reduced from those of default CPLEX. We tested various different sets of penalties, ranging from 5 to 10000. Results as shown in Table 6 indicate that as the penalty size increases the optimality gaps decreased but for penalties bigger than 50 there was no improvement in general. A penalty setting of 50 provided the best results.

Comparing the best results against the default CPLEX branch-and-bound method, we were able to solve seven more problems to optimality and for two more of them the solutions provided were near enough to optimal for any practical purposes(gaps of 0.15% and 0.20%). Secondly, except three instances, we were able to provide solutions with good quality bounds (gap < 10%), while CPLEX was able to do it for only a handful of instances. Overall, the average optimality gap decreased from 35.15% to 4.59% with the addition of cuts and heuristics to the branch-and-bound procedure. Hence with the use of heuristics and cutting planes we were able to provide solutions with seven times better optimality gap, on average, as compared to default branch-and-bound method of CPLEX.

The computational results provided strongly indicate that the cuts and heuristic methods are both important for getting good solutions with tighter optimality gaps.

3.7 Comparison of Heuristic Quality

We implemented two heuristics proposed in literature for getting solutions for the OSPF weight setting problem. This was done for two reasons. Firstly, we wanted to make a comparison with our heuristic methods. Secondly, we believed that if the heuristic methods in the literature provided better solutions, then integrating them with our branch-and-cut

approach would lead to solving more problems and possibly solving them faster.

Both the heuristic methods we proposed were based on local search, one based on Genetic Algorithms [31] and other based on Simulated Annealing [57]. We provide a brief introduction for both the algorithms. For details refer to the corresponding papers. See [43] for general discussion on Simulated Annealing and [9][39] for Genetic Algorithms.

3.7.1 Genetic Algorithm Based Heuristic (GA)

The Genetic Algorithm (GA) presented [31] consists of the following steps:

```

begin
   $t = 0$ ;
  initialize Pop(0);
  evaluate Pop(0) ;
  while Not termination-criteria do
     $t = t + 1$  ;
    select Pop( $t$ ) from Pop( $t-1$ ) ;
    alter Pop( $t$ ) ;
    evaluate Pop( $t$ ) ;
  end
end

```

Figure 17: Genetic Algorithm

Population at step t is represented by $\text{Pop}(t)$. Population was initialized by selecting random integer weights from $[1, 65535]$. Each solution was associated to a fitness value. A given set of OSPF weights completely determine the shortest paths and hence how much flow is sent over which arcs and the load on each arc. The fitness value for each solution was assigned based on the maximum load (among all the arcs) in the network as computed for that solution.

After computing the fitness for each solution the population was divided into three classes, based on fitness; elite (class A) and non-elite (class B & C). The top 20% of the solutions per fitness values were categorized as Class A, the next 70% were non-elite Class B and the rest were assigned to Class C. Crossover was done by mating two parents in the population, one from the elite class (Class A) and the other from the non-elite class (Class B or C). To cross two parent solutions p_1 (elite) and p_2 (non-elite) into a new solution, the

gene for the new solution is inherited from parent p_1 with a certain probability K (> 0.5). It is done by generating a random vector between 0 and 1 of size number of arcs. Also to escape early local convergence a gene of a solution is changed randomly with certain probability during each crossover.

To create the next generation of the GA, all Class A solutions are promoted without change, all Class C solutions are replaced by random solutions, and the remaining solutions are created by crossover.

We implemented this heuristic with the same settings as proposed by Ericsson et al. [31]. Two different sizes of initial population were used, namely 20 and 50, to perform computational results.

3.7.2 Simulated Annealing Based Heuristic (SA)

The Simulated Annealing implemented is based on the heuristic algorithm proposed by Pioro et. al. [57]. The basic algorithm is presented below:

The algorithm is started with random solution w_{old} . At each step, the algorithm selects a neighbor of w_{old} . The neighboring state, w_{new} , is obtained by selecting at random a link and incrementing or decrementing its weight by 1 (the selection from the two possibilities is also random). Then the demands are routed according to the new weight system w_{new} , and the cost of the new state, $C(w_{new})$, is calculated and compared to the cost of the old one, $C(w_{old})$. If the cost of the new state is not greater than that of the old one, the new state is always accepted. If $C(w_{new})$ is greater than $C(w_{old})$, the new state is accepted according to the Metropolis test. The outcome of the test depends on the current temperature T and on the cost difference between the states (C). At the beginning, SA will accept states with relatively large (C) with a high probability, which is then decreased exponentially during the optimization process. This allows for a better scanning of the state space to avoid local optima. At the end of the main loop the temperature is decreased ($a < 1$); the loop is executed until the temperature reaches a predefined lower bound.

The algorithm is implemented based on the settings as proposed by Pioro et. al. [57].

```

begin
  Initialize (w_old) ;
  min_cost:= C (w_old);
  w_best:= w_old ;
  T:= initial temperature;
  while T ≥ temp_lower_bound and min_cost > cost_lower_bound do
    for counter=0 to counter_upper_bound do
      w_new = neighbor(w_old);
      ΔC = C (w_new) - C (w_old);
      if ΔC ≤ 0 then
        w_old = w_new ;
        if C (w_new) < min_cost then
          min_cost = C (w_new);
          w_best =w_new;
        end
      end
      else if random < exp{−ΔC/T} then
        w_old = w_new ;
      end
    end
    T = T × a ;
  end
end

```

Figure 18: Simulated Annealing

3.7.3 Traditional Recommendations

CISCO [25][69] recommends setting the OSPF weights inversely proportional to the capacity of the link (InvCap). The justification behind this strategy is that, smaller the capacity of the arc, the weight should be set larger so fewer shortest paths go through this arc and vice versa for arcs with large capacity.

Another strategy could be setting all the weights to 1 (Unit) [51]. In this case the shortest path would be the path with shortest number of arcs from a source to destination. We implemented both these strategies to demonstrate why these strategies might not work in general and also to show the need for an optimization-based approach.

3.7.4 Proposed Heuristics

For the OSPF weight setting problem, we implemented two basic heuristic methods as discussed in Section 3.5.3. The first of them was a local-search heuristic based on weight

adjustment. Details are provided in Figure 16. The other heuristic was based on the integer programming formulation discussed in 3.5.3.2.

3.7.5 Discussion of Results

Table 7 presents the comparison of various heuristic methods. The first column is the problem name, followed by the solution returned if all the weights are set to 1 (Unit OSPF). The third column denotes the CISCO recommendation [24] for the OSPF weight setting problem, setting weights inversely proportional to capacity. The fourth and fifth columns show the results for the Genetic Algorithm for population size of 20 and 50 respectively. The next column presents result for the Simulated Annealing algorithm. Next column (HEUR) presents the results from the heuristic proposed for the OSPF weight setting problem in this thesis. The next two columns show, in percentages, how much better our solution is as compared with the best solution returned by Unit-OSPF and InvCap. The last two columns show the improvements over Genetic Algorithm and Simulated Annealing respectively.

Table 7: Comparison of different heuristic strategies

<i>Prob Name</i>	<i>Unit</i>	<i>InvCap</i>	<i>GA20</i>	<i>GA50</i>	<i>SA</i>	<i>HEUR</i>	<i>Percentage Improvements by HEUR</i>			
							<i>Imp-Unit</i>	<i>Imp-InvCap</i>	<i>Imp-GA</i>	<i>Imp-SA</i>
n5d10	1.7019	2.2241	1.7019	1.7019	1.7019	1.7019	0.00%	30.68%	0.00%	0.00%
n10d5	1.6029	1.4118	1.0000	1.0000	1.0000	1.0000	60.29%	41.18%	0.00%	0.00%
n10d6	0.7419	1.1912	0.5294	0.5294	0.6667	0.4224	75.64%	181.99%	25.33%	57.82%
n10d11	1.6544	1.4259	1.2525	1.2776	1.4944	1.2396	33.47%	15.03%	1.04%	20.56%
n10d12	3.4008	2.4304	1.8632	1.8632	2.4304	1.7697	92.17%	37.33%	5.28%	37.33%
n10d15	1.9365	1.5484	1.5484	1.2530	1.5974	1.0407	86.07%	48.78%	20.40%	53.49%
n15d5	1.4035	1.8571	0.6667	0.6265	1.0968	0.6125	129.16%	203.23%	2.30%	79.08%
n15d6	1.2381	1.6032	0.8860	0.8860	1.1471	0.7101	74.36%	125.77%	24.77%	61.54%
n15d7	1.0241	1.5606	0.5597	0.5597	1.0241	0.3973	157.75%	292.78%	40.88%	157.75%
n15d8	1.3038	1.8454	1.0206	0.8929	1.3038	0.6023	116.46%	206.37%	48.23%	116.46%
n15d9	0.7500	1.0130	0.7059	0.5909	0.8791	0.5417	38.46%	87.01%	9.09%	62.30%
n15d10	1.9608	2.0706	0.9529	0.9529	1.1176	0.5294	270.37%	291.11%	80.00%	111.11%
graph1	6.6667	4.2647	2.6316	2.6316	6.6667	2.6316	153.33%	62.06%	0.00%	153.33%
caldata	1.0417	1.2639	0.8611	0.8611	0.8611	0.8611	20.97%	46.77%	0.00%	0.00%
snh	6.4063	2.7660	1.7188	1.5566	2.5490	1.4623	338.10%	89.16%	6.45%	74.32%
Pioro7	1.2471	1.4943	1.0619	1.1494	1.3228	1.0619	17.45%	40.72%	0.00%	24.57%
Pioro12w1	8.6111	1.4831	1.0000	1.0000	7.0000	1.0000	761.11%	48.31%	0.00%	600.00%
Pioro12w2	5.5938	1.2128	0.9180	0.8418	5.9043	0.8145	586.79%	48.90%	3.35%	624.91%
Average Improvement							167.33%	105.40%	14.84%	124.14%

Unit - All weights set to 1

InvCap - All weights set to inversely proportional to capacity

GA20, *GA50* - Genetic Algorithm with population size 20 and 50

SA - Simulated Annealing heuristic

HEUR - Heuristic proposed in thesis

Imp-Unit - Improvement of *HEUR* over *Unit*

Imp-InvCap - Improvement of *HEUR* over *InvCap*

Imp-GA - Improvement of *HEUR* over *GA*

Imp-SA - Improvement of *HEUR* over *SA*

The results from the Unit and InvCap columns of Table 7 as compared with our heuristic solution demonstrate that neither Unit OSPF, setting weights on the arcs to 1, nor InvCap OSPF, setting weights inversely proportional to the capacity of the arc, is a good strategy for setting the weights in an OSPF network. Both the construction strategies recommended by CISCO usually yield sub-optimal solutions and leave a lot of room for improvement. Comparing the results from Imp-Unit and Imp-InvCap columns, we can see that the heuristic proposed by us outperforms both the Unit OSPF and InvCap strategies by more than 100%, on average. That is, our heuristic decreases the maximum utilization in the network by half. Among these construction strategies, the InvCap strategy usually provides better solutions as compared with the Unit strategy.

The two local search heuristics, Genetic Algorithm (GA), and Simulated Annealing (SA), also return solutions that are inferior to the solutions provided by our heuristics. Comparing the two columns for Genetic Algorithm, GA20 and GA50, we see that Genetic Algorithm heuristic provides better solution with larger population size, as is expected. But larger population size also leads to larger computation times. Also the Genetic Algorithm almost always outperforms Simulated Annealing (SA). Our heuristic methods return a solution better at least as good as any other methods implemented for all the tested instances. On average, Simulated Annealing was outperformed by more than 100% and Genetic Algorithm by about 15%, when compared with our heuristic method.

After comparing the results we can say that, firstly, construction strategies as recommended by CISCO may not provide good set of weights and in some cases the solution provided can be far away from optimal. So the approaches that provide good solutions become very important. Secondly, the heuristic methods proposed in the thesis always outperform the other heuristic methods tested. Lastly, even though the solutions returned by our heuristics look good, there is no way to know whether the heuristic has found the optimal solution. Integer programming based approaches are needed to make a claim like that.

3.8 Final Thoughts

A few final thoughts on the OSPF weight setting problem. First and foremost, we need integer programming based methods for the weight setting problem to get good solutions with quality guarantees. The linear programming relaxation for the MILP model proposed is weak and there is need to develop cutting planes to strengthen it. The valid inequalities derived for the weight setting problem were based on exploiting the equal-split restriction in the model. A next step should be to consider the shortest path structure in the problem to develop more valid inequalities.

Heuristic methods are as important as the cutting planes to solve the weight setting problem. Decomposition based approaches to the problem do not seem to work as well. Also, due to the nature of the objective function used, there are lot of symmetry issues that need to be looked at.

CHAPTER IV

EQUAL-SPLIT FLOW PROBLEMS ON BIPARTITE GRAPHS

4.1 *Problem Definition*

In the equal-split flow problem on a bipartite graph we are given two sets of nodes, sources and sinks, each source node has a positive incoming flow that has to be routed out to any of the given sinks and a restriction on number of sinks that the flow can be sent to. Every sink has a capacity on flow that can be routed to it. The problem is to send the flow from sources to sinks such that all the flow is routed and capacity of the sinks is not exceeded. The key requirement is that if the flow from a particular source is routed to multiple sinks then it has to be equally split among them. It is easy to see that for a given set of data this problem might not be feasible. Even when the sum of incoming flow is less than the sum of the capacity of the sinks, it might be infeasible due to the equal-split restriction that does not allow the flow to be split arbitrarily. So in order to make the problem feasible we allow the flow assignment to go over sinks capacity. The objective is to minimize the summed deviation over the capacity of all the sinks. See Figure 19.

One of the applications of the equal-split problem on bipartite graphs is network switch design. Network switches [1] are used to route flow in a Local Area Network (LAN). Switches consist of two sets of ports, input ports and output ports. The flow comes into the switch through the input ports and is routed out through one of the output ports. The flow coming from a source s , into the switch through one of the input ports say i , going to destination t is routed through the available output ports (the ports connecting to destination t). The switch has the provision to split the packets among the available output ports (for load balancing purposes). Since the links connecting the output ports to other parts of the network have limited capacities, an arbitrary assignment of output ports may lead to overloading of one of these links (since there are multiple sources trying to send flow). So the assignment of output ports to input ports for different sources may

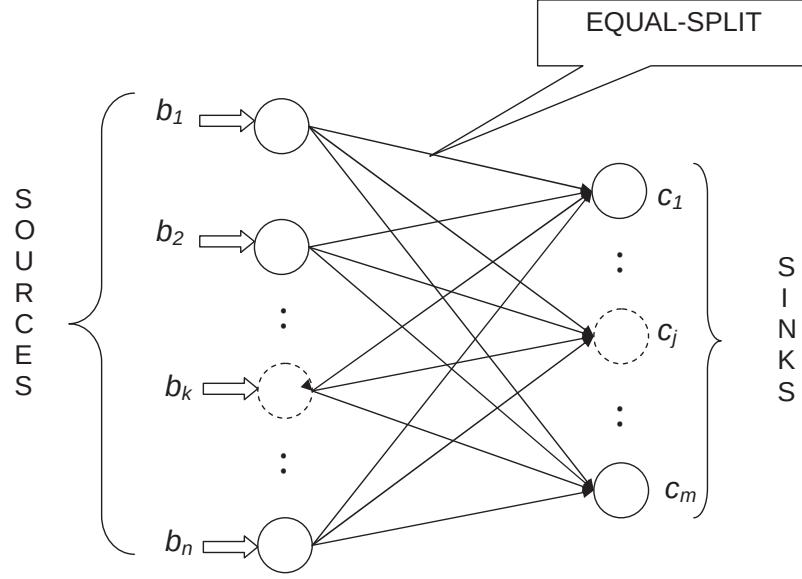


Figure 19: An equal-split problem on bipartite graphs

be considered as a part of the switch design. This problem can be seen as the equal-split problem on bipartite graphs, where the set of input ports form one bipartition of the bipartite graph and the output ports form the other bipartition. We assume that we have an estimate on the incoming flow into each input port (from historical data). The capacities of the outgoing links from the output ports are the capacities of the second bipartition of the graph. The objective is to assign output ports to input ports while trying to minimize the total deviation of exceeding the capacities of the links. Similar equal-split structure is also used in peer-to-peer networks [29] where the resources are equally split among the connected users as one of the methods for allocation of resources.

For the equal-split flow problem on bipartite graphs if the equal-split restriction and the constraint on number of sinks each source can split the flow into is removed, and arbitrary fractional flow is allowed then this problem is easy to solve. Also, if the sum of incoming flow, i.e. sum of exogenous flow into the sources, is less than or equal to sum of the capacity of the sinks, then it is easy to see that the optimal solution to the relaxed problem is always zero. This basically means that there always exists an assignment such that none of the capacities of the sinks are exceeded. But we can not make the same statement for the problem with the equal-split restriction, under the same assumptions on sum of flows and

capacities. Indeed this problem is NP-complete, which we prove next. First we define the problem formally.

This problem can be represented on a bipartite graph, where one bipartition represents the sources N_1 and the other sinks N_2 . We assume that all the arcs between N_1 and N_2 are present. Each source $i \in N_1$ has an incoming flow of b_i and the maximum number of sinks that the flow can be sent to, is denoted by M_i . Each sink $j \in N_2$ has a capacity c_j on the flow that can be assigned to sink j . The problem is to route flow from sources to sinks, and if the flow from a source is assigned to multiple sinks then it has to be equally split among them, while minimizing the summed deviation over the capacity of the sinks.

We prove the hardness result by reducing a variation of the Subset Sum problem to our problem. Subset Sum with Repetitions Problem (SSRP) defined as follow:

(SSRP): Given positive integers $a_1 < a_2 < \dots < a_n$ and $r_1, r_2, \dots, r_n, t$. Determine if there exists integers x_i , $0 \leq x_i \leq r_i$ for $i = 1, \dots, n$ such that $\sum_i a_i x_i = t$.

SSRP is NP-complete since it contains the subset sum problem when ($r_i = 1$ for all i). Also, it has been shown by Alfonsin [4] that the problem is still NP-complete if $r_i = 2$ for all i . We denote the special case of SSRP with 2-SSRP. We are going to provide a reduction from 2-SSRP to our equal-split problem on bipartite graphs.

Theorem 6. *The equal-split flow problem on bipartite graphs is NP-complete.*

Proof. Consider an instance of 2-SSRP. Construct an instance of equal-split problem on bipartite graphs with n sources and 2 sinks. The exogenous flow into source i is $2a_i$ and the capacity of sink 1 is t and the capacity of second sink is $(2\sum_i a_i - t)$. Each source can split the flow among either of the sinks. Now the problem is to determine an equal-split assignment of sources to sinks such that summed deviation over the capacity of the sinks is 0. This is equivalent to assigning t units of flow to sink 1 and rest of the flow to sink 2.

Now if we have a solution x to 2-SSRP, then equal-split assignment defined below:

1. If $x_i = 2$, then assign the source i to sink 1.
2. If $x_i = 1$, then split the flow from source i equally among both the sinks.

3. If $x_i = 0$, then assign the source i to sink 2.

yields an assignment to equal-split problem with deviation of 0.

Similarly, if we have a solution to the equal-split problem on bipartite graphs with deviation 0, then construct the solution to 2-SSRP as follows:

1. If flow from source i is assigned to sink 1 completely, then set $x_i = 2$.
2. If flow from source i is split to sink 1 and 2, then set $x_i = 1$.
3. If flow from source i is assigned to sink 2 completely, then set $x_i = 0$.

Hence the problem is NP-complete. □

4.2 Problem Formulation

We can formulate this problem similar to the multi-commodity flow problem with (x, f, d) variables from Chapter 2, but after performing computational tests we found out that this formulation did not yield good results, even after adding the cutting planes derived for this formulation. So we provide a different formulation for modelling the equal-split restriction. We model the mixed integer linear programming formulation with the help of the following variables.

Decision variables:

- x_{ij} binary variable denoting whether flow from source i is routed to sink j .
- f_{ij} amount of flow routed from source i to sink j .
- y_{ik} binary variable denoting the flow from source i is split into k equal parts
- Δ_j deviation of incoming flow over the capacity of sink j

The mixed integer linear programming (MILP) formulation is given below:

$$(P1) \quad \min \quad \sum_j \Delta_j$$

$$\text{s.t.} \quad \sum_j f_{ij} = b_i \quad \forall i \quad (83)$$

$$\sum_i f_{ij} - \Delta_j \leq c_j \quad \forall j \quad (84)$$

$$\sum_k y_{ik} = 1 \quad \forall i \quad (85)$$

$$\sum_k k y_{ik} = \sum_j x_{ij} \quad \forall i \quad (86)$$

$$f_{ij} \leq \frac{b_i}{k} + b_i(2 - y_{ik} - x_{ij}) \quad \forall i, j, k \quad (87)$$

$$f_{ij} \geq \frac{b_i}{k} - b_i(2 - y_{ik} - x_{ij}) \quad \forall i, j, k \quad (88)$$

$$x_{ij} \in \{0, 1\}, \quad f_{ij} \geq 0 \quad \forall i, j$$

$$y_{ik} \in \{0, 1\} \quad \forall i, k$$

$$\Delta_j \geq 0 \quad \forall j$$

The set of constraints (83) are flow conservation for every source i . The next set of constraints (84) are capacity constraints for every sink j that measure deviation Δ_j . The set of constraints (85) model the requirement that flow can only be split in one way. The next set of constraints (86) make sure that the number of arcs chosen is same as the corresponding split variable y , for every source i . The next two sets of constraints (87) and (88) model the equal-split of the flow on the arcs. Finally, we have the binary and the non-negativity restrictions on the variables.

The optimal value of the linear programming relaxation of $(P1)$, is $\max \left\{ \left(\sum_i b_i - \sum_j c_j \right), 0 \right\}$. Thus linear programming relaxation is weak and it is very hard to solve this problem directly by the CPLEX integer programming solver.

In order to strengthen the linear programming relaxation of $(P1)$, we carry out a polyhedral study and obtain some strong valid inequalities in the next section.

4.3 Valid Inequalities

4.3.1 A simple system

Consider the following simple system of $(P1)$ given by the constraints (85)-(86). It represents the relation between the number of arcs chosen, $\sum_j x_{ij}$, and the binary variables y_{ik} denoting the number of sinks used to split the flow from a given source i . Dropping the index i , the system we study is represented as:

$$Y^1 = \left\{ (x, y) \in \{0, 1\}^n \times \{0, 1\}^n : \right. \\ \left. \sum_k y_k = 1 \right. \\ \left. \sum_j x_j = \sum_k ky_k \right\}$$

First we have the following result for the convex hull of Y^1 , $\text{co}(Y^1)$.

Proposition 17. *The dimension of convex hull of Y^1 , $\text{co}(Y^1)$ is $2n - 2$.*

Proof. The dimension of the convex hull of Y^1 is at most $2n - 2$, because the two equality constraints in the description of Y^1 are linearly independent. In order to prove that the dimension of the convex hull is exactly $2n - 2$ we show $2n - 2 + 1 = 2n - 1$ affinely independent solutions. The following set of $2n - 1$ points are in Y^1 and can be seen to be affinely independent.

$$\left(\begin{array}{ccccc|ccccc} x_1 & x_2 & x_3 & \dots & x_n & y_1 & y_2 & y_3 & \dots & y_n \\ 1 & 0 & 0 & \dots & 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 & 1 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 1 & 0 & 0 & \dots & 0 \\ \hline 1 & 1 & 0 & \dots & 0 & 0 & 1 & 0 & \dots & 0 \\ 1 & 1 & 1 & \dots & 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 & 0 & 0 & 0 & \dots & 1 \end{array} \right)$$

□

Even though the system Y^1 is very simple, its linear programming relaxation is still not integral, i.e. the LP relaxation polyhedron contains fractional extreme points. To see this, consider Y^1 with $n = 3$. Let an objective function be defined as follows: $c_{x_1} = 1$, $c_{x_2} = 1$, $c_{x_3} = 0$, $c_{y_1} = 3$, $c_{y_2} = 0$, $c_{y_3} = 2$. Then the optimal solution is : $x_1 = x_2 = 1, x_3 = 0$ and $y_1 = y_3 = \frac{1}{2}, y_2 = 0$, with objective value $4\frac{1}{2}$ and this is the unique optimal solution for this objective. For this example with $n = 3$, in any binary solution to Y^1 we know that if $y_3 = 1$ then all of the x 's have to be chosen, so the following inequalities are valid: $x_1 \geq y_3$, $x_2 \geq y_3$, $x_3 \geq y_3$, and the solution defined before would not satisfy the inequality $x_3 \geq y_3$. In fact for the above defined objective, after adding the inequality $x_3 \geq y_3$, the optimal solution becomes integral. Observations along similar lines lead us to the following set of valid inequalities.

Proposition 18. *Let $1 \leq l \leq n - 2$ and let S_l denote the set of all subsets of $\{1, \dots, n\}$ of cardinality l , (note $|S_l| = \binom{n}{l}$). Then the following inequalities are valid for Y^1 .*

$$\sum_{j \in I} x_j \geq \sum_{m=1}^l m y_{n-l+m} \quad \forall I \in S_l, 1 \leq l \leq n - 2 \quad (89)$$

Proof. To prove the validity of (89), pick any point in Y^1 and consider p such that $y_p = 1$. Let l be fixed. If $p < n - l + 1$ then the right hand side is zero and the inequality is valid.

So we may assume that $p \geq n - l + 1$, more specifically let $p = n - l + q$ with $q \geq 1$, integer. Then the right hand side is equal to $qy_p = q$. Now $l = n - p + q$ and any subset of $\{1, \dots, n\}$ of size $n - p + q$ would have to have at least q of the x 's with value 1. And this is exactly the inequality.

□

The inequality (89) implies that if $y_p = 1$, i.e. there are p variables x with value 1, then for any subset of $\{1, \dots, n\}$ of size $n - p + 1$ there has to be an x_j with $j \in I$ such that $x_j = 1$. And for all subsets of $\{1, \dots, n\}$ of size $n - p + k$, k of the x_j 's for $j \in I$ has to be set to 1.

Proposition 19. *The family of valid inequalities (89) are facet defining for $\text{co}(Y^1)$.*

Proof. The dimension of convex hull of Y^1 is $2n - 2$, so to prove the facet defining property, we will show $2n - 2$ affinely independent points satisfying the valid inequality at equality. We will show this for a fixed l . Also, without loss of generality we may assume that set I consists of the first l components, i.e. $I = \{1, \dots, l\}$. Then the following set of $2n - 2$ points are affinely independent and satisfy the inequality (89) at equality.

$$\left(\begin{array}{cccc|cccc|cccc|cccc}
x_1 & x_2 & \dots & x_l & x_{l+1} & x_{l+2} & \dots & x_n & y_1 & y_2 & \dots & y_{n-l} & y_{n-l+1} & y_{n-l+2} & \dots & y_n \\
1 & 0 & \dots & 0 & 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\
0 & 1 & \dots & 0 & 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 1 & 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\
\hline
0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & 0 & 1 & \dots & 0 & 1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & 1 & 1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
\hline
0 & 0 & \dots & 0 & 1 & 1 & \dots & 0 & 0 & 1 & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & 1 & 1 & \dots & 1 & 0 & 0 & \dots & 1 & 0 & 0 & \dots & 0 \\
\hline
1 & 1 & \dots & 0 & 1 & 1 & \dots & 1 & 0 & 1 & \dots & 0 & 0 & 1 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 & 0 & 1 & \dots & 0 & 0 & 0 & \dots & 1
\end{array} \right)$$

The first (l) points, second $(n-l)$ points, the third $(n-l-1)$ points and the last $(l-1)$ points, are all affinely independent and total up to $2n-2$, providing required number of points. $\square$

Another interesting property of the above defined family is that it cuts off all the fractional extreme points of the linear programming relaxation of Y^1 .

Proposition 20. *The family of valid inequalities given by (89) cuts off every fractional extreme point of the linear relaxation of Y^1 .*

Proof. To prove the above property, we first claim that all the extreme points of Y^1 have all x components integer.

We have $2n$ variables and at a given solution, a maximum of $2n+2$ inequalities can be active (only one of the bound inequalities can be active at any feasible solution). If some component of y is fractional, then there has to be another component of y fractional as well

because of the constraint $\sum_k y_k = 1$. So if we have an extreme point with fractional x , then y can not be fractional as it would lead to at most $(2n - 1)$ active constraints but for an extreme point we need $2n$. So we may assume that if x is fractional then all components of y are integer. Since all components of y are integer, the right hand side of the equality:

$$\sum_j x_j = \sum_k k y_k$$

is integer and hence there have to exist two fractional x components (can not be more than two for an extreme point). Let their two indices be k_1 and k_2 . This leads to

$$x_{k_1} + x_{k_2} = \text{integer}.$$

Now since $0 < x_{k_1} + x_{k_2} < 2$, we have that $x_{k_1} + x_{k_2} = 1$, which implies $y_1 = 1$. Then this fractional point is a convex combination of following two points

$$x_{k_1} = 1, y_1 = 1, \text{everything else zero},$$

$$x_{k_2} = 1, y_1 = 1, \text{everything else zero}.$$

Hence the claim that all the extreme points of Y^1 have all x components integer.

Now any extreme point with fractional y will have to have at least two fractional components and an extreme point, also can have at most two fractional components. Let the two fractional components be k_1 and k_2 . Then we have

$$\sum_{j=1}^n x_j = k_1 y_{k_1} + k_2 y_{k_2}.$$

We have that $k_2 - k_1 \geq 2$, since $k_1 y_{k_1} + k_2 y_{k_2}$ has to be an integer and $y_{k_1} + y_{k_2} = 1$. It implies that $\sum_j x_j = w \in (k_1, k_2)$, and also we know that w is an integer. Now let $l = n - w$ and the set I be the one that contains all the x_j 's that are zero (there are $(n - w)$ of them). Then the left hand side of the valid inequality (89) for this set is 0. The inequality becomes:

$$0 = \sum_{j \in I} x_j \geq \sum_{m=1}^{n-w} m y_{n-(n-w)+m} = \sum_{m=1}^{n-w} m y_{w+m} = y_{w+1} + 2y_{w+2} + \dots + (n-w)y_n > 0$$

since $k_2 > w$. Hence this point is cut off by one of the inequalities (89). Hence all the extreme points of Y^1 are either integer, or if fractional are cut off by the family of valid inequalities (89). $\square$

We believe that the set of inequalities (89) plus the original inequalities for Y^1 are enough to define the convex hull of Y^1 . The properties shown above support our conjecture.

Conjecture 1. *The family of inequalities (89) plus the original inequalities defining Y^1 are sufficient to define the convex hull of Y^1 , $\text{co}(Y^1)$.*

Separation of (89)

Even though the family of valid inequalities (89) are exponential in number we have a polynomial time scheme for separating them.

Given a fractional solution $(\hat{x}, \hat{y})$, we sort the components of $\hat{x}$ in ascending order of their value. For every l , the right hand side of (89) is fixed and hence we just need to check whether there exists a set $I \subset S_l$ s.t. $\sum_{i \in I} x_i$ is smaller than the right hand side. This can be done by checking the smallest l components of the fraction solution $\hat{x}$, or the first l components of the $\hat{x}$ sorted in ascending order. If the sum of the smallest l components of $\hat{x}$ greater than the right hand side then there does not exist a set I , for that l such that the any of the inequalities are violated; otherwise, the set I defined by the smallest l components provides the violated set. Since sorting requires $O(n \log n)$ time, so this scheme is polynomial.

We have considered a simple subsystem of the equal-split system. This simple system relates the binary variable x that denotes which arcs were chosen, to the binary variables y that denote the number of arcs chosen. The actual flow assignment on the chosen arcs was not considered. In the following section we add the flow assignment to the single node system and consider the same subsystem in (x, y, f) space.

4.3.2 A simple single node system

Here we consider a simple single node subsystem in (x, y, f) space as shown in Figure 20. We assume that the flow can be split among all the sinks, so $M_i = n$ for all sources i .

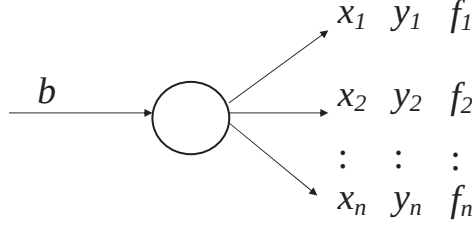


Figure 20: An (x, y, f) single node system.

The single source node system with equal-split and with positive incoming flow b can be represented as:

$$\begin{aligned}
Y^2 = \Big\{ \quad & (x, y, f) \in \{0, 1\}^n \times \{0, 1\}^n \times \mathbb{R}^n : \\
& \sum_k y_k = 1 \\
& \sum_i x_i = \sum_k k y_k \\
& \sum_i f_i = b \\
& f_i \leq \frac{b}{k} + b(2 - y_k - x_i) \quad \forall i, k \\
& f_i \geq \frac{b}{k} - b(2 - y_k - x_i) \quad \forall i, k \\
& 0 \leq x_i \leq 1, f_i \geq 0 \\
& 0 \leq y_k \leq 1 \Big\}
\end{aligned}$$

Note. In our original formulation of the equal-split restriction introduced in Chapter 2 with (x, f, d) variables, the dummy variable d is equivalent to $\sum_k (\frac{b}{k}) y_k$ for the new formulation. Hence all the valid inequalities for the original formulation are still valid for new formulation when we replace d by $\sum_k (\frac{b}{k}) y_k$.

First we have a result for the dimension of the convex hull of (Y^2) .

Proposition 21. *The dimension of convex hull of Y^2 is $3n - 3$.*

Proof. The three equalities in the description of Y^2 are linearly independent hence the dimension of convex hull of Y^2 is at most $3n - 3$. We will show $3n - 2$ affinely independent points in Y^2 to show that the rank is exactly $3n - 3$.

$$\left(\begin{array}{c|c|c} x_1 & x_2 & x_3 & \dots & x_n & y_1 & y_2 & y_3 & \dots & y_n & f_1 & f_2 & f_3 & \dots & f_n \\ \hline 1 & 0 & 0 & \dots & 0 & 1 & 0 & 0 & \dots & 0 & b & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 & 1 & 0 & 0 & \dots & 0 & 0 & b & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & b & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & b \\ \hline 1 & 1 & 0 & \dots & 0 & 0 & 1 & 0 & \dots & 0 & \frac{b}{2} & \frac{b}{2} & 0 & \dots & 0 \\ 1 & 0 & 1 & \dots & 0 & 0 & 1 & 0 & \dots & 0 & \frac{b}{2} & 0 & \frac{b}{2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & 1 & 0 & 1 & 0 & \dots & 0 & \frac{b}{2} & 0 & 0 & \dots & \frac{b}{2} \\ 0 & 1 & 1 & \dots & 0 & 0 & 1 & 0 & \dots & 0 & 0 & \frac{b}{2} & \frac{b}{2} & \dots & 0 \\ \hline 1 & 1 & 1 & \dots & 0 & 0 & 0 & 1 & \dots & 0 & \frac{b}{3} & \frac{b}{3} & \frac{b}{3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 & 0 & 0 & 0 & \dots & 1 & \frac{b}{n} & \frac{b}{n} & \frac{b}{n} & \dots & \frac{b}{n} \end{array} \right)$$

The above defined $(3n - 2)$ points are affinely independent and are feasible to Y^2 . $\square$

We now present valid inequalities for the system Y^2 :

Proposition 22. Let $\gamma_k = \frac{(k-1)(k-2)}{2k}$ and $\eta = \frac{n-3}{2(n-1)}$, then the following inequalities are valid for Y^2 :

$$b \sum_{k=3}^n \gamma_k y_k + \frac{b}{2} x_j - \frac{b}{2} \sum_{i \neq j} x_i - f_j \leq -\frac{b}{2} \quad \forall j \quad (90)$$

$$-b \sum_{k=3}^n \gamma_k y_k + \frac{b}{2} \sum_{i \neq j} x_i + \eta b x_j - \sum_{i \neq j} f_i \leq \eta b \quad \forall j \quad (91)$$

$$\sum_{k=l+1}^n b \frac{(l-k)}{lk} y_k + \frac{b}{l} x_j - f_j \leq 0 \quad \forall j, \quad l \leq n-1 \quad (92)$$

Proof. To prove the validity of (90) we take a point in Y^2 and consider the following cases:

- (i.) $y_1 = 1$. If $x_j = 1$, then LHS = $\frac{b}{2} - b = -\frac{b}{2} = \text{RHS}$, so the inequality is valid. If $x_j = 0$ then $\exists i \neq j$ such that $x_i = 1$; hence LHS = $-\frac{b}{2} = \text{RHS}$, so the inequality is valid.

(ii.) $y_2 = 1$. If $x_j = 1$, then $\text{LHS} = \frac{b}{2} - \frac{b}{2} - \frac{b}{2} = \text{RHS}$, so the inequality is valid. If $x_j = 0$ then $\exists i, k \neq j$ such that $x_i = x_k = 1$; hence $\text{LHS} = -b \leq -\frac{b}{2} = \text{RHS}$, so the inequality is valid.

(iii.) $y_l = 1, l \geq 3$ and $x_j = 1$. We have that (LHS-RHS) equal to

$$\begin{aligned} &= \frac{(l-1)(l-2)}{2l}b + \frac{b}{2} - \frac{b}{2}(l-1) - \frac{b}{l} + \frac{b}{2} \\ &= b \left(\frac{l^2 - 3l + 2 + l - l^2 + l - 2 + l}{2l} \right) = 0. \end{aligned}$$

Hence the inequality is valid in this case.

(iv.) $y_l = 1, l \geq 3$ and $x_j = 0$. We have (LHS-RHS) equal to

$$= \frac{(l-1)(l-2)}{2l}b - \frac{b}{2}l + \frac{b}{2} = b \left(\frac{l^2 - 3l + 2 - l^2 + l}{2l} \right) = -b \frac{(l-1)}{l} \leq 0$$

for all $l \geq 1$, integer.

To prove the validity of (91) we consider a point in Y^2 and consider the following cases:

(i.) $y_1 = 1$. If $x_j = 1$, then $\text{LHS} = \eta b \leq \eta b = \text{RHS}$, so the inequality is valid. If $x_j = 0$, then $\text{LHS} = \frac{b}{2} - b \leq \eta b = \text{RHS}$ (since $\eta \geq 0$), so the inequality is valid.

(ii.) $y_2 = 1$. If $x_j = 1$, then $\text{LHS} = \frac{b}{2} + \eta b - \frac{b}{2} \leq \eta b = \text{RHS}$, so the inequality is valid. If $x_j = 0$, then $\text{LHS} = b - b \leq \eta b$, so the inequality is valid.

(iii.) $y_l = 1, l \geq 3$ and $x_j = 1$. We have (LHS-RHS) equal to

$$\begin{aligned} &= -b \frac{(l-1)(l-2)}{2l} + \frac{b}{2}(l-1) + b \frac{n-3}{2(n-1)} - b \frac{(l-1)}{l} - b \frac{n-3}{2(n-1)} \\ &= \frac{-b(l-1)(l-2) + bl(l-1) - 2bl(l-1)}{2l} = 0 \end{aligned}$$

(iv.) $y_l = 1, l \geq 3$ and $x_j = 1$. We have (LHS-RHS) equal to

$$-b \frac{(l-1)(l-2)}{2l} + \frac{b}{2}l - b - b \frac{n-3}{2(n-1)} = b \frac{(l-2)}{2l} - b \frac{n-3}{2(n-1)} \leq 0$$

for all $3 \leq l \leq n-1$ (since $x_j = 0$, implies $y_n = 0$).

It is easy to see that the inequality (91) is valid when $y_k = 1$ if $k \leq l$. Let $y_k = 1$ with $k \geq l + 1$. If $x_j = 0$ then the inequality is valid as ($l < k$). If $x_j = 1$, we have LHS-RHS $= b \frac{(l-k)}{lk} + \frac{b}{l} - \frac{b}{k} = b \frac{(l-k+k-l)}{lk} = 0$. $\square$

Proposition 23. *Families of valid inequalities (90), (91) and (92) are facet defining for the convex hull of Y^2 .*

Proof. The dimension of convex hull of Y^2 is $3n - 3$, so the proof of the facet defining property is constructed by showing $3n - 3$ affinely independent points satisfying the inequality at equality.

Consider the valid inequality (90) and let $j = 1$. For the proof we have the following set of affinely independent points.

$$\left(\begin{array}{c|c|c} \begin{array}{ccccc} x_1 & x_2 & x_3 & \dots & x_n \end{array} & \begin{array}{ccccc} y_1 & y_2 & y_3 & \dots & y_n \end{array} & \begin{array}{ccccc} f_1 & f_2 & f_3 & \dots & f_n \end{array} \\ \hline \begin{array}{ccccc} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{array} & \begin{array}{ccccc} 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & 0 \end{array} & \begin{array}{ccccc} b & 0 & 0 & \dots & 0 \\ 0 & b & 0 & \dots & 0 \\ 0 & 0 & b & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & b \end{array} \\ \hline \begin{array}{ccccc} 1 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & 1 \end{array} & \begin{array}{ccccc} 0 & 1 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & 0 & \dots & 0 \end{array} & \begin{array}{ccccc} \frac{b}{2} & \frac{b}{2} & 0 & \dots & 0 \\ \frac{b}{2} & 0 & \frac{b}{2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{b}{2} & 0 & 0 & \dots & \frac{b}{2} \end{array} \\ \hline \begin{array}{ccccc} 1 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{array} & \begin{array}{ccccc} 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{array} & \begin{array}{ccccc} \frac{b}{3} & \frac{b}{3} & \frac{b}{3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{b}{n} & \frac{b}{n} & \frac{b}{n} & \dots & \frac{b}{n} \end{array} \end{array} \right)$$

To prove the facet defining property of (91), consider the following set of $(3n - 3)$ affinely independent points.

x_1	x_2	x_3	$\dots$	x_{n-1}	x_n	y_1	y_2	y_3	$\dots$	y_{n-1}	y_n	f_1	f_2	f_3	$\dots$	f_{n-1}	f_n
0	1	1	$\dots$	1	1	0	0	0	$\dots$	1	0	0	$\frac{b}{n-1}$	$\frac{b}{n-1}$	$\dots$	$\frac{b}{n-1}$	$\frac{b}{n-1}$
1	0	1	$\dots$	1	1	0	0	0	$\dots$	1	0	$\frac{b}{n-1}$	0	$\frac{b}{n-1}$	$\dots$	$\frac{b}{n-1}$	$\frac{b}{n-1}$
1	1	0	$\dots$	1	1	0	0	0	$\dots$	1	0	$\frac{b}{n-1}$	$\frac{b}{n-1}$	0	$\dots$	$\frac{b}{n-1}$	$\frac{b}{n-1}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
1	1	1	$\dots$	0	1	0	0	0	$\dots$	1	0	$\frac{b}{n-1}$	$\frac{b}{n-1}$	$\frac{b}{n-1}$	$\dots$	0	$\frac{b}{n-1}$
1	1	1	$\dots$	1	0	0	0	0	$\dots$	1	0	$\frac{b}{n-1}$	$\frac{b}{n-1}$	$\frac{b}{n-1}$	$\dots$	$\frac{b}{n-1}$	0
1	1	0	$\dots$	0	0	0	1	0	$\dots$	0	0	$\frac{b}{2}$	$\frac{b}{2}$	0	$\dots$	0	0
1	0	1	$\dots$	0	0	0	1	0	$\dots$	0	0	$\frac{b}{2}$	0	$\frac{b}{2}$	$\dots$	0	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
1	0	0	$\dots$	1	0	0	1	0	$\dots$	0	0	$\frac{b}{2}$	0	0	$\dots$	$\frac{b}{2}$	0
1	0	0	$\dots$	0	1	0	1	0	$\dots$	0	0	$\frac{b}{2}$	0	0	$\dots$	0	$\frac{b}{2}$
1	1	1	$\dots$	0	0	0	0	1	$\dots$	0	0	$\frac{b}{3}$	$\frac{b}{3}$	$\frac{b}{3}$	$\dots$	0	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
1	1	1	$\dots$	1	1	0	0	0	$\dots$	0	1	$\frac{b}{n}$	$\frac{b}{n}$	$\frac{b}{n}$	$\dots$	$\frac{b}{n}$	$\frac{b}{n}$
1	0	0	$\dots$	0	0	1	0	0	$\dots$	0	0	b	0	0	$\dots$	0	0

The first set consists of n points, the second set consists of $n - 1$ points, the third set consists of $n - 3$ points (it does not have the $(n - 1)^{st}$ point from first set) and last one add up to $3n - 3$. It is easy to see that these points are affinely independent.

To prove the facet defining property of (92), consider the following set of $3n - 3$ affinely independent points.

$$\left(\begin{array}{c|c|c} x_1 & x_2 & x_3 & \dots & x_n & y_1 & y_2 & y_3 & \dots & y_n & f_1 & f_2 & f_3 & \dots & f_n \\ \hline 1 & 0 & 0 & \dots & 0 & 1 & 0 & 0 & \dots & 0 & b & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 & 1 & 0 & 0 & \dots & 0 & 0 & b & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & b & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & b \\ \hline 1 & 1 & 0 & \dots & 0 & 0 & 1 & 0 & \dots & 0 & \frac{b}{2} & \frac{b}{2} & 0 & \dots & 0 \\ 1 & 0 & 1 & \dots & 0 & 0 & 1 & 0 & \dots & 0 & \frac{b}{2} & 0 & \frac{b}{2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & 1 & 0 & 1 & 0 & \dots & 0 & \frac{b}{2} & 0 & 0 & \dots & \frac{b}{2} \\ \hline 1 & 1 & 1 & \dots & 0 & 0 & 0 & 1 & \dots & 0 & \frac{b}{3} & \frac{b}{3} & \frac{b}{3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 & 0 & 0 & 0 & \dots & 1 & \frac{b}{n} & \frac{b}{n} & \frac{b}{n} & \dots & \frac{b}{n} \end{array} \right)$$

□

Separation of (90)-(92)

All the above defined family of valid inequalities (90), (91) and (92) are of linear size and can be checked for violation at a fractional point by going through each of them. It is not clear if we can do better.

We present another family of valid inequalities for Y^2 .

Proposition 24. *Let S_m denote collection of subsets I of $\{1, \dots, n\}$ of size m . Then the following set of inequalities are valid for Y^2 .*

$$\sum_{k=l+1}^n \frac{(k-1)(k-l)b}{k} y_k - b \sum_{i \in I} x_i + \sum_{i \in I} f_i \leq 0 \quad \forall I \in S_{n-l}, l \geq 3 \quad (93)$$

Proof. Pick a point in Y^2 and let $y_m = 1$. For $m < l + 1$, (93) is trivially valid. So we may assume that $m \geq l + 1$, and when $y_m = 1$, for any l , the maximum value of left hand side is:

$$\frac{(m-1)(m-l)}{m}b - (m-l)b + \frac{(m-l)}{m}b = 0 \leq 0.$$

Hence the inequality is valid for all l and all sets S_{n-l} . $\square$

Separation of (93)

The family (93) of valid inequalities is exponential in size but we can separate efficiently. Given a fractional solution $(\hat{x}, \hat{y}, \hat{f})$, compute $(b\hat{x}_i - f_i)$ for each arc i and we sort the arcs in ascending order of $(b\hat{x}_i - f_i)$. Now for every l we need to check if there exists a set I of size $n-l$, such that

$$\sum_{k=l+1}^n \frac{(k-1)(k-l)b}{k} \hat{y}_k - b \sum_{i \in I} \hat{x}_i + \sum_{i \in I} \hat{f}_i > 0.$$

Pick the first $(n-l)$ arcs in I (sorted) and check if

$$\sum_{i \in I} (b\hat{x}_i - \hat{f}_i) < \sum_{k=l+1}^n \frac{(k-1)(k-l)b}{k} \hat{y}_k.$$

If it is we have a violated inequality; otherwise there does not exist any set $I \subset S_l$ for which the inequality is violated for the fixed l . This leads to $O(n \log n)$ time separation scheme where the sorting step ($O(n \log n)$) is the longest.

In the discussion above, we exploited the equal-split structure of a single node system and developed cutting planes for these subsystems, but we did not use the unique bipartite structure of the problem. In the following section we consider the equal-split problem as a whole and perform a similar polyhedral study to obtain valid inequalities.

4.4 Structure based cuts

In this section we develop valid inequalities that utilize the whole structure of the equal-split problem on bipartite graphs. So far we have considered the structure from the source's side, where the restriction of equal-split was exploited to generate valid inequalities. Now we consider the sink side and develop more valid inequalities.

Consider a sink with capacity c . Let there be n sources with incoming flow $b_1, b_2, \dots, b_n$ (refer to Figure 21). Let x_i be the binary variable denoting whether flow from source i

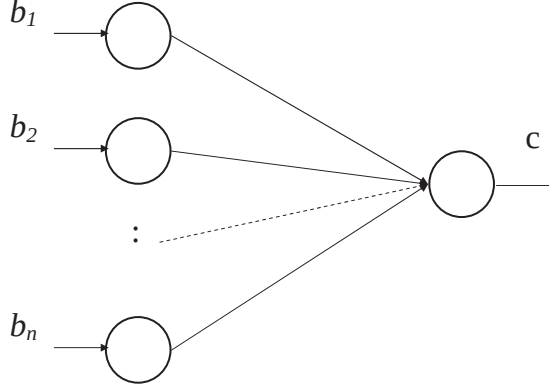


Figure 21: A Single sink node with multiple sources

is sent to the sink or not. Let M_i denote the maximum number of sinks the flow can be assigned to from source i . If source i is assigned to the sink, the minimum flow assignment from this source to sink will be $\frac{b_i}{M_i}$, because at minimum the flow from source i is split into M_i sinks and each sink will be assigned $\frac{b_i}{M_i}$ flow. Based on this observation, it is easy to see the inequality defined below is valid, and it provides a lower bound on deviation (Δ) for the sink:

$$\Delta \geq \sum_i \left(\frac{b_i}{M_i} \right) x_i - c. \quad (94)$$

Also we have the following set of valid inequalities based on M_i .

Proposition 25. *The following inequalities are valid for the equal-split problem on bipartite graphs:*

$$\Delta \geq -\left(c - \frac{b_j}{M_j}\right)x_i - \frac{b_j}{M_j}(1 - x_j) + f_i \quad \forall i, j, i \neq j \quad (95)$$

$$\Delta \geq -\left(c - \sum_{j \neq i} \frac{b_j}{M_j}\right)x_i - \sum_{j \neq i} \frac{b_j}{M_j} - \sum_{j \neq i} \frac{b_j}{M_j}x_j + f_i \quad \forall i \quad (96)$$

Proof. The proof of validity for (95) is presented by considering the following four cases:

- (i.) $x_i = x_j = 0$. In this case $f_i = f_j = 0$, hence $\Delta \geq 0 \geq -\frac{b_j}{M_j} + f_i = \text{RHS}$.
- (ii.) $x_i = 0, x_j = 1$. In this case we have $f_i = 0$, and hence $\Delta \geq 0 \geq f_i = \text{RHS}$.
- (iii.) $x_i = 1, x_j = 0$. In this case we have $\Delta \geq f_i - c = \text{RHS}$.

(iv.) $x_i = x_j = 1$. In this case, $\Delta \geq f_i + f_j - c \geq f_i + \frac{b_j}{M_j} - c = \text{RHS}$ (since when $x_j = 1$, $f_j \geq \frac{b_j}{M_j}$).

The set of inequalities (96) are a form of an aggregated version of (95), hence the proof is similar and is not repeated here. $\square$

We also have the following set of valid inequalities that depend on a specific value of M_i .

$$\Delta \geq \sum_{i:M_i=2} \left(-(b_i - c)x_i + \frac{2(b_i - c)}{b_i} f_i \right) + \sum_{i:M_i=1} (b_i - c)x_i \quad (97)$$

$$\Delta \geq \left(c - \frac{b_i}{2} \right) x_i + \frac{2(b_i - c)}{b_i} f_i - \sum_{j \neq i} \left(c - \frac{b_i}{2} \right) x_j + \sum_{j \neq i} f_j - \frac{b_i}{2} \quad \forall i \text{ (with } M_i = 2) \quad (98)$$

All the above defined inequalities are polynomial in size and also they depend on the data of the instance, so these inequalities are added to CPLEX cut pool at the start of the branch-and-bound method.

4.5 *Branch-and-Cut Results*

We demonstrate the effectiveness of our cuts by performing computational experiments on randomly generated instances of the equal-split flow problem on bipartite graphs. The random instances were created such that the sum of the incoming flow is always less than or equal to the sum of the capacities of the sinks.

We incorporated the cutting planes developed in the previous section within a branch-and-cut algorithm to solve the equal-split problem on bipartite graphs. We used CPLEX 9.0 callable libraries to implement the branch-and-cut algorithm. Computations were done on an Intel Pentium 4 Zeon machine with 2.4 GHz speed and 2.0 GB RAM running Linux kernel 2.4.27.

For the family of valid inequalities that are not polynomial in size, i.e. (89) and (93), we have fast separations schemes so these were added when violated, using the cutcallback function of CPLEX. But for the rest, which were mostly linear in size, we added those cutting planes in the CPLEX cut pool. We tested the effectiveness of cuts by comparing our procedure with solutions given by default CPLEX with no cuts.

4.5.1 Heuristics

We developed basic heuristic methods to get good starting solutions. These heuristic methods were based on the MILP formulation. Denote by M_i the maximum number of sinks that flow from source node i can be split amongst and let $m_{min} = \min_i M_i$.

The first approach to get solutions was to fix the number of sinks for flow from each source i to be equal for all of the sources. For example to require that the flow from every source node to be equally split among 2 sinks, we set $y_{i2} = 1$ for all i and the rest of y 's to zero. This way we solve m_{min} integer programs by fixing y_{ik} to 1 for $k = 1, \dots, m_{min}$ with the same objective and store the best solution obtained. Since the problem solved is a special case of the original problem, the solutions obtained are feasible to the original problem as well. For these integer programs, the only decision variables are x 's and f 's, and hence they are much easier to solve. Also, since y 's are fixed, we know the exact number of arcs chosen from each source, hence we know the exact amount of the flow on the arc if it is chosen. We can use this observation to strengthen the cuts family (94) and this makes the problem much easier to solve. Even then we do not have to solve them to optimality, but we can stop the branch-and-bound after certain amount of time to get a good solution.

The second approach is to fix that flow from each source i has to be split equally amongst M_i sinks for each source i . This can be done by setting y_{iM_i} to 1 for each i and the rest to zero. Here we solve one integer program, as compared to m_{min} integer programs in the previous approach, and store the best solution obtained. Similar to the first approach we can strengthen certain families of cuts which makes this restricted problem easier to optimize.

After experimenting with both the approaches we found out that, invariably the second approach was giving better solutions than the first approach. The second approach also seemed more attractive since we just need to solve one integer program in the second approach as compared with m_{min} in the first approach. So the second approach was used to get starting solutions.

4.5.2 Computational Experiments

Table 8 shows the comparison of lower and upper bounds returned by default CPLEX and when we add our cuts and heuristic. Results are presented in such a way that it is easy to see the improvement in both lower and upper bounds when cutting planes are added. The first column denotes the problem name. The problem En_1-n_2 denotes an instance of equal-split flow problem on bipartite graphs with n_1 source nodes and n_2 sink nodes. The second and the last column present the lower and upper bound obtained by CPLEX at the end of 300 seconds of computation time. Third and fourth column present the best lower and upper bound obtained when cuts and heuristics are used to solve the problem.

Table 8: Lower and upper bounds for instances of equal-split assignment on bipartite graphs (time limit: 300s)

Prob	CPLEX LB	CUTS + HEUR		CPLEX UB
		LB	UB	
E10_10	0.00	0.00	0.00	0.00
E10_11	1.00	6.17	6.17	19.00
E10_12	6.00	14.33	14.33	21.17
E10_13	0.00	0.00	0.00	0.00
E10_14	0.00	6.25	6.25	28.08
E10_15	1.00	13.75	13.75	24.00
E10_20	2.00	50.50	50.50	65.00
E15_10	0.00	7.75	7.75	15.25
E15_15	0.00	0.00	0.00	0.00
E15_20	0.00	5.25	6.15	31.25
E20_10	0.00	0.00	0.00	8.00
E20_15	0.00	10.00	10.00	26.67
E20_20	0.00	8.75	10.17	48.83
E20_25	0.00	16.75	17.75	108.00
E20_30	0.00	0.00	0.00	92.58
E25_20	0.00	0.00	0.00	17.83
E25_25	0.00	0.00	7.83	71.00
E25_30	0.00	0.00	0.00	0.00

Entries in **bold** denote convergence to an optimal solution

The results from Table 8 show that the lower bounds returned by CPLEX at the end of the run are quite bad in general, the main reason being that the LP relaxation is very weak. By adding our cuts we were able to significantly increase the lower bounds for almost

Table 9: Time and node counts comparison

Prob	CPLEX		Cuts + Heur	
	Node Count	Time(s)	Node Count	Time(s)
E10_10	240	1.50	0	4.01
E10_11	86543	300.00	762	16.47
E10_12	65324	300.00	19	7.78
E10_13	30	1.12	0	5.06
E10_14	38413	300.00	434	20.11
E10_15	31001	300.00	1655	40.43
E10_20	15565	300.00	5	7.12
E15_10	37247	300.00	16150	68.32
E15_15	155	8.91	0	4.72
E15_20	8053	300.00	4388	300.00
E20_10	33334	300.00	3766	106.49
E20_15	11261	300.00	5	10.25
E20_20	6571	300.00	8855	300.00
E20_25	2100	300.00	1527	300.00
E20_30	753	300.00	3	15.19
E25_20	1844	300.00	0	6.80
E25_25	2939	300.00	3438	300.00
E25_30	1000	235.32	0	13.91

Entries in **bold** represent that optimal solution was found in 300 second

all of the problems. Not only did the lower bounds improve, our heuristic methods with the branch-and-bound procedure were able to significantly decrease the upper bounds as compared with upper bounds returned by CPLEX. Comparing the results from different columns we can see that we were able to solve more problems to optimality and we were able to provide relatively tight bounds for most of the rest.

Table 9 presents the actual node count and computational times for the random instances tested. The time limit was 300 seconds. If the problem was not solved to optimality in allotted time, then the corresponding entry in time column is 300 and the node count entry is the number of nodes explored in 300 seconds. Not only were we able to solve more problems by using heuristic methods and adding cuts but we were also able to solve them in less time and exploring lesser number of nodes, as is evident from Table 9.

Tables 8 and 9 show that the proposed heuristics and cutting planes significantly improved the performance of the branch-and-bound method for solving equal-split problems

defined on bipartite graphs. In the following section we will consider an alternate formulation for this problem.

4.6 An Alternate Formulation

For the equal-split problem on bipartite graphs, it is not hard to see that flow variables are not needed for the integer programming formulation. Consider a single source with exogenous flow b and n sinks. Let the set $S = \{1, 2, \dots, n\}$ be the set of sinks connected to the source. Now if we know that the set $I \subset S, I \neq \emptyset$, is the set of sinks used to route flow from the source, then the flow on the arcs connecting the source to the sink is automatically determined, and is given by:

$$f_i = \begin{cases} \frac{b}{|I|} & \text{if } i \in I, \\ 0 & \text{otherwise.} \end{cases} \quad (99)$$

So once the assignment is known, the flow on all the arcs is determined as well. Based on this realization we propose a new formulation for the equal-split flow problem for bipartite graphs that does not contain any continuous flow variables.

Consider a source with n sinks. Let x_j^k be the binary variable that will be 1 if flow is routed to sink k and a total $j - 1$ other sinks. Then the following set defines the equal-split structure for a single source:

$$Z = \left\{ (x) \in \{0, 1\}^{n^2} : \sum_{k=1}^n \sum_{j=1}^n \left(\frac{x_j^k}{j} \right) = 1, \right. \quad (100)$$

$$\left. \begin{aligned} \sum_{k=1}^n x_j^k &\geq j x_j^k \quad \forall j, k \\ 0 &\leq x_j^k \leq 1 \end{aligned} \right\}. \quad (101)$$

It is easy to see that the above formulation correctly models the single source node with equal-split.

Now for the equal-split flow problem on bipartite graphs, we now show how the deviation is calculated for each worker. Consider a sink k , and let there be n_1 sources that can send flow to sink k . Let us denote by $(x_j^k)_i$, binary decision variables for source i , as defined above for the single source. Then the deviation Δ_k is given by:

$$\Delta_k = \max \left\{ \left(\sum_{i=1}^{n_1} \sum_{j=1}^n \frac{b_i}{j} (x_j^k)_i - c_k \right), 0 \right\}$$

which can be written as:

$$\Delta_k \geq \sum_{i=1}^{n_1} \sum_{j=1}^n \frac{b_i}{j} (x_j^k)_i - c_k \quad (102)$$

$$\Delta_k \geq 0 \quad (103)$$

when the objective is such that each Δ_k will be set as small as possible.

Using the constraints defined above the equal-split problem on bipartite graphs can be modeled as:

$$\begin{aligned} \min \quad & \sum_k \Delta_k \\ \text{s.t.} \quad & \sum_{k=1}^n \sum_{j=1}^n \left(\frac{(x_j^k)_i}{j} \right) = 1, \forall \text{ sources } i \\ & \sum_{k=1}^n (x_j^k)_i \geq j(x_j^k)_i \quad \forall i, j, k \\ & \Delta_k \geq \sum_{i=1}^{n_1} \sum_{j=1}^n \frac{b_i}{j} (x_j^k)_i - c_k \quad \forall \text{ sinks } k \\ & 0 \leq (x_j^k)_i \leq 1, \text{ binary, } \forall i, j, k \end{aligned} \quad (P2)$$

In the formulation defined above, we got rid of the flow variables f and binary variables y , but we added a lot more binary variables. If there are n_1 sources and n_2 sinks, then the number of binary variables in the previous formulation (P1) is $2n_1n_2$, and the number of binary variables in the new formulation (P2) is $n_1n_2^2$. Also there are no continuous variables in the second formulation. So it is not clear which formulation will be better in

terms of solving the integer program. In terms of the linear programming relaxation, both formulations provide the same LP lower bound for the objective of minimizing the sum of deviations for each worker.

For the set defined by Z for a single node, let Z_{LP} be the linear programming relaxation, where the binary restrictions on x variables are removed and replaced by lower and upper bounds of 0 and 1 respectively. We can show that this polyhedron is integral.

Theorem 7. $Z_{LP} = \text{co}(Z)$.

Proof. We will prove this result by contradiction. Assume that above is not true. Then there exist an extreme point of Z_{LP} such that one of the components is fractional. Let $\hat{x}$ be such a fractional extreme point represented by (x_j^k) , $j = 1, \dots, n$, $k = 1, \dots, n$.

Let j_1 be one of the fractional components, by this we mean that there exists a k such that $x_{j_1}^k$ is fractional. We claim that there exists k_1 and k_2 s.t. $x_{j_1}^{k_1}, x_{j_1}^{k_2} > 0$, and are strictly fractional. To see this consider different cases of j_1 . If $j_1 > 1$ then it can be easily seen from (101). If $j_1 = 1$, then either there exists such k_1 and k_2 fractional or else there exists $j_2 > 1$ which has the above desired property and we can substitute j_1 by j_2 and so that j_2 will have the desired properties.

If $j_1 = 1$, then we know that $x_1^{k_1}$ and $x_1^{k_2}$ are strictly positive and less than 1. So we can find an ϵ such that both the points $a = (\dots, x_1^{k_1} + \epsilon, \dots, x_1^{k_2} - \epsilon, \dots)$ and $b = (\dots, x_1^{k_1} - \epsilon, \dots, x_1^{k_2} + \epsilon, \dots)$ are feasible and $\hat{x} = \frac{1}{2}a + \frac{1}{2}b$, contradicting the fact that $\hat{x}$ is an extreme point.

Now if we have multiple choices of $j_1 > 1$, pick j_1 such that there exists k_1, k_2 s.t. $0 < x_{j_1}^{k_1}, x_{j_1}^{k_2} < 1$ and both of the constraints

$$\sum_{k=1}^n x_{j_1}^k \geq j_1 x_{j_1}^{k_1} \quad (104)$$

$$\sum_{k=1}^n x_{j_1}^k \geq j_1 x_{j_1}^{k_2} \quad (105)$$

are not active. If there exists such a j_1 then we can find ϵ , small enough to satisfy all the constraints, s.t. both the points $a = (\dots, x_{j_1}^{k_1} + \epsilon, \dots, x_{j_1}^{k_2} - \epsilon, \dots)$ and $b = (\dots, x_{j_1}^{k_1} -$

$\epsilon, \dots, x_{j_1}^{k_2} + \epsilon, \dots$) are feasible and $\hat{x} = \frac{1}{2}a + \frac{1}{2}b$, contradicting the fact that $\hat{x}$ is an extreme point.

Suppose there is no j satisfying (104) and (105), then we can see from (101) that for all j such that there exists k_1, k_2 s.t. $0 < x_j^{k_1}, x_j^{k_2} < 1$, we have that: $x_j^k = 0$ or F_j , since all the constraints (101) are active.

From (100) and the above property, we know that there has to exist j_1 and j_2 such that

$$x_{j_1}^k = 0 \text{ or } F_{j_1} \text{ and } x_{j_2}^k = 0 \text{ or } F_{j_2}.$$

Let I_{j_1} and I_{j_2} be the number of fractional k 's in x_{j_1} and x_{j_2} . Then we can find ϵ_1 and ϵ_2 such that the following points are feasible:

$$\begin{aligned} a &= (\dots, x_{j_1}^k + \epsilon_1, \dots, x_{j_2}^k - \epsilon_2, \dots) \\ b &= (\dots, x_{j_1}^k - \epsilon_1, \dots, x_{j_2}^k + \epsilon_2, \dots) \end{aligned}$$

where

$$\frac{I_{j_1}\epsilon_1}{j_1} = \frac{I_{j_2}\epsilon_2}{j_2} \text{ (to maintain feasibility of (100)).}$$

Both ϵ_1 and ϵ_2 can be chosen appropriately. Now $\hat{x} = \frac{1}{2}a + \frac{1}{2}b$, contradicting the assumption that $\hat{x}$ is an extreme point. $\square$

4.6.1 Valid Inequalities for (P2)

Consider a sink with capacity c and n different sources with incoming flow b_i that can be assigned to the sink. See Figure 22.

For notational convenience we represent the binary variables from source i to the sink by x_j^i , which denotes the binary decision of sending flow from source i to the sink under consideration and the total number of sinks used is j . Then we have that the deviation for the sink Δ , satisfies the following inequality:

$$\Delta \geq \sum_{i=1}^n \sum_{j=1}^{M_i} \left(\frac{b_i}{j} \right) x_j^i - c$$

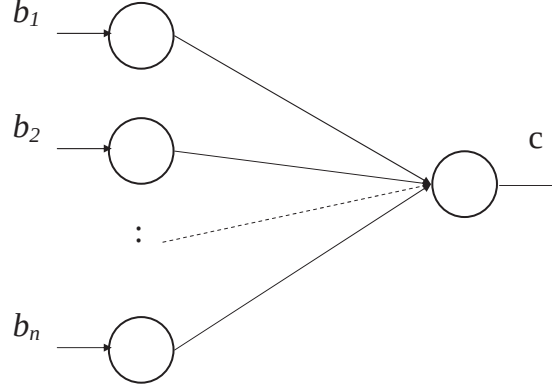


Figure 22: A simple subsystem

Let us denote for each i the term $\left(\frac{b_i}{j}\right)$ by α_{ij} . Then the above inequality becomes:

$$\Delta \geq \sum_{i=1}^n \sum_{j=1}^{M_i} \alpha_{ij} x_j^i - c. \quad (106)$$

Now we will present a family of valid inequalities for the subsystem shown in Figure 22 and hence for the bipartite system.

Proposition 26. *Let L be an integer between 0 and c , then the following is a valid inequality for the bipartite system (P2):*

$$\begin{aligned} \Delta &\geq \sum_{i \neq k} \sum_{j=1}^{M_i} \max\{0, (\alpha_{ij} - L)\} x_j^i + \sum_{j: \alpha_{kj} \geq c-L} \max\{(c-L), (\alpha_{kj} - L)\} x_j^k + \\ &+ \sum_{j: \alpha_{kj} < c-L} (\alpha_{kj} x_j^k) - (c-L) \quad \forall k \in \{1, \dots, n\} \end{aligned} \quad (107)$$

Proof. To prove the validity of (107), consider any binary point of the equal-split bipartite polyhedron. To check the validity it is sufficient to check for a point where for each i , there exists j such that $x_j^i = 1$. Since the right hand side will only decrease if one of the x 's is zero, so validity of other solutions would follow from this case. Pick a $k \in \{1, \dots, n\}$. Two cases arise:

(I) $\alpha_{kj} \geq c - L$. In this case the right hand side is given by:

$$\sum_{i=1, i \neq k}^n \max\{0, (\alpha_{ij} - L)\} + \max\{(c-L), (\alpha_{kj} - L)\} - c + L$$

Two subcases arise:

(a) $\max\{(c - L), (\alpha_{kj} - L)\} = (c - L)$. We have that

$$\sum_{i=1, i \neq k}^n \max\{0, (\alpha_{ij} - L)\} + (c - L) - c + L \leq \sum_{i=1, i \neq k}^n \max\{0, (\alpha_{ij} - L)\} + \alpha_{kj} - c + L \quad (108)$$

since $\alpha_{kj} \geq c - L$. If $\sum_{i=1, i \neq k}^n \max\{0, (\alpha_{ij} - L)\} = 0$, then the left hand side of (108) is 0, and the inequality $\Delta \geq 0$ is trivially valid. Otherwise if $\sum_{i=1, i \neq k}^n \max\{0, (\alpha_{ij} - L)\} > 0$, then it is easy to see that

$$\sum_{i=1, i \neq k}^n \max\{0, (\alpha_{ij} - L)\} + \alpha_{kj} - (c - L) \leq \sum_{i=1}^n \alpha_{ij} - c \leq \Delta.$$

(b) $\max\{(c - L), (\alpha_{kj} - L)\} = (\alpha_{kj} - L)$. We have that the right hand side of the valid inequality (107) in question is:

$$\sum_{i=1, i \neq k}^n \max\{0, (\alpha_{ij} - L)\} + (\alpha_{kj} - L) - c + L = \sum_{i=1, i \neq k}^n \max\{0, (\alpha_{ij} - L)\} + \alpha_{kj} - c.$$

It is easy to see that

$$\sum_{i=1, i \neq k}^n \max\{0, (\alpha_{ij} - L)\} + \alpha_{kj} - c \leq \sum_{i=1}^n (\alpha_{ij}) - c \leq \Delta.$$

(II) $\alpha_{kj} < c - L$. In this case the right hand side of (107) is:

$$\sum_{i=1, i \neq k}^n \max\{0, (\alpha_{ij} - L)\} + \alpha_{kj} - c + L$$

which we can see is less than:

$$\leq \sum_{i=1, i \neq k}^n (\alpha_{ij}) - L + \alpha_{kj} - c + L \leq \Delta.$$

This completes the proof of validity of (107). $\square$

The size of family of valid inequalities (107) depend on the data in the problem, i.e. the capacities of the sinks c , hence it is exponential. But L should be chosen such that $c - L = \alpha_{kj}$ for some k, j . And if L is chosen in this way, it leads to a polynomial size and can just be added to the cut pool right at the start of the branch-and-bound method. Once an inequality (107) is derived, then more inequalities can be developed from the derived inequalities.

4.6.2 Comparison of two formulations

In this section we compare the effectiveness of the two formulations ($P1$) and ($P2$) in solving the equal-split problem on bipartite graphs.

As already mentioned before, the linear programming relaxation optimal solution value is the same for both formulations. Also, formulation ($P2$) contains more binary variables than ($P1$) and also does not have any continuous variables. But the constraints required to model the single node equal-split subsystem for ($P2$) are sufficient for convex hull but this is not true for ($P1$).

In order to make a comparison between the two formulations, we compare the optimal LP solution and its characteristics. Since the LP relaxation solution value is the same for both the formulations, we compare the fractional components of the optimal LP solution. We compare various statistics as shown in Table 10. We first consider the number of fractional components. Since there are different numbers of binary variables in the two formulations, we also consider the percentage of fractional variables. Also, to see how fractional the optimal solution is we compare the total fractionality of the solutions. For a given fractional solution x , total fractionality is calculated as: $\sum_i \min\{x_i, 1 - x_i\}$. The average fractionality is calculated by dividing the total fractionality by the number of fractional variables. This tells on average how far the fractional solution is from an integer solution.

Table 10 presents the above-mentioned statistics. The first column is the problem name. The next five columns present statistics for the LP optimal solution for formulation ($P1$). The last five columns present the same statistics for formulation ($P2$). From Table 10 we can see that even though ($P2$) has many more binary variables than ($P1$), the total number of fractional variables at the LP solution are less. This also means that the percentage of fractional variables in ($P2$) are less as compared with ($P1$). From these results we can say that formulation ($P2$) seems to have fewer fractional components, even though it has more binary variables, so the optimal LP solution is closer to an integer solution for ($P2$) as compared with ($P1$). But, if we compare the average fractionalities across the two formulations there is not much difference across the two formulations.

Table 10: Comparison of two formulations

<i>Prob</i>	(P1)					(P2)				
	Frac	Total	% Frac	$\sum$ Frac	$(\sum \text{Frac})/\text{Tot}$	Frac	Total	% Frac	$\sum$ Frac	$(\sum \text{Frac})/\text{Tot}$
E10_10	23	200	11.50	6.70	0.291	14	1000	1.40	3.74	0.267
E10_11	21	220	9.55	5.72	0.273	16	1210	1.32	2.96	0.185
E10_12	37	240	15.42	9.97	0.270	17	1440	1.18	3.48	0.204
E10_13	17	260	6.54	4.32	0.254	11	1690	0.65	3.26	0.296
E10_14	33	280	11.79	8.70	0.264	19	1960	0.97	4.70	0.247
E10_20	43	400	10.75	10.54	0.245	28	4000	0.70	5.72	0.204
E10_15	29	300	9.67	6.99	0.241	22	2250	0.98	4.39	0.200
E15_10	43	300	14.33	12.36	0.287	16	1500	1.07	3.04	0.190
E15_15	39	450	8.67	12.42	0.318	16	3375	0.47	4.48	0.280
E15_20	59	600	9.83	13.60	0.230	25	6000	0.42	6.15	0.246
E20_10	56	400	14.00	18.23	0.325	15	2000	0.75	3.55	0.236
E20_15	64	600	10.67	14.28	0.223	24	4500	0.53	6.31	0.263
E20_20	62	800	7.75	14.88	0.240	32	8000	0.40	7.56	0.236
E20_25	73	1000	7.30	15.11	0.207	38	12500	0.30	10.22	0.269
E20_30	81	1200	6.75	19.48	0.240	40	18000	0.22	7.00	0.175
E25_20	76	1000	7.60	20.10	0.265	28	10000	0.28	6.86	0.245
E25_25	43	1250	3.44	13.52	0.315	41	15625	0.26	11.83	0.289
E25_30	93	1500	6.20	21.73	0.234	41	22500	0.18	10.12	0.247

Frac: Number of fractional components in optimal LP solution

Total: Total number of binary variables

% Frac: Percentage fractional variables

$\sum \text{Frac}$: Sum of fractional components calculated as $\sum_i \min\{x_i, 1 - x_i\}$

$(\sum \text{Frac})/\text{Tot}$: Sum of fractional components divided by total number of fractional variables

We now present results on how effective the two formulations were at solving the equal-split flow problem on bipartite graphs. CPLEX branch-and-bound was used to solve the two mixed integer programming formulations.

Table 11: Comparison of two formulations: lower and upper bounds were returned at the end of 300s of computation time

<i>Prob</i>	<i>(P1)</i>		<i>(P2)</i>	
	LB	UB	LB	UB
E10_10	0.00	0.00	0.00	0.00
E10_11	1.00	19.00	3.67	6.83
E10_12	6.00	21.17	14.33	14.33
E10_13	0.00	0.00	0.00	0.00
E10_14	0.00	28.08	6.25	9.42
E10_20	2.00	65.00	50.50	50.50
E10_15	1.00	24.00	11.50	13.75
E15_10	0.00	15.25	7.75	7.75
E15_15	0.00	0.00	0.00	0.00
E15_20	0.00	31.25	5.00	15.50
E20_10	0.00	8.00	0.00	0.50
E20_15	0.00	26.67	9.76	11.50
E20_20	0.00	48.83	8.54	21.50
E20_25	0.00	108.00	15.00	21.41
E20_30	0.00	92.58	0.00	20.50
E25_20	0.00	17.83	0.00	0.00
E25_25	0.00	71.00	0.00	42.42
E25_30	0.00	0.00	0.00	0.00

Entries in **bold** represent that optimal solution was found in 300 second

Table 11 presents the best lower and upper bounds reported by CPLEX branch-and-bound for each of the formulations after 300 seconds of computation time. Table 12 reports the solve times and node counts for each of the formulations. If the problem was not solved to optimality in the allotted time (300s), the corresponding entry in the Time column is 300 and the entry in the Node Count column is left blank.

From Table 11 we can see that formulation (*P2*) was much more effective in solving the equal-split problem on bipartite graphs. CPLEX branch-and-bound with formulation (*P2*) was able to solve four more problems as compared with formulation (*P1*). More importantly the lower and upper bound returned at the end of the run for formulation (*P2*) were significantly better than the bounds for formulation (*P1*). Also from Table 12 we can

see that for the problems that were solved to optimality, both node counts and solve times were significantly better for formulation $(P2)$ as compared with $(P1)$. This shows that as an integer programming formulation, with no additional cuts or heuristics added, formulation $(P2)$ performs better than formulation $(P1)$.

Table 12: Comparison of two formulations: Node counts and solve times

<i>Prob</i>	<i>(P1)</i>		<i>(P2)</i>	
	Node Count	Time	Node Count	Time
E10_10	240	1.5	56	0.19
E10_11	86543	300.00	264266	300.00
E10_12	65324	300.00	13740	12.20
E10_13	30	1.12	8	0.10
E10_14	38413	300.00	134958	300.00
E10_20	15565	300.00	101	0.53
E10_15	31001	300.00	136464	300.00
E15_10	37247	300.00	17000	17.68
E15_15	155	8.91	16	0.17
E15_20	8053	300.00	86127	300.00
E20_10	33334	300.00	234194	300.00
E20_15	11261	300.00	129512	300.00
E20_20	6571	300.00	71903	300.00
E20_25	2100	300.00	46454	300.00
E20_30	753	300.00	67791	300.00
E25_20	1844	300.00	56647	94.39
E25_25	2939	300.00	38221	300.00
E25_30	1000	235.32	900	11.62

Entries in **bold** represent that optimal solution was found in 300 second

We studied the equal-split structure as modeled by formulation $(P1)$ and developed cutting planes. We also implemented a basic heuristic method, based on the IP formulation, to get good starting solution. Computational results demonstrated the effectiveness of the cutting planes and heuristics in solving instances of equal-split problem on bipartite graphs. For formulation $(P2)$, we did not provide an extensive study of the equal-split polyhedron as modeled by $(P2)$. But, we did provide a family of valid inequalities for $(P2)$. Also, the same heuristic implemented for $(P1)$ can be used for $(P2)$ as well. The results after adding cutting planes and heuristics for $(P2)$ were better than the CPLEX default branch-and-bound on the formulation. In order to make a comparison on how effective the cutting

planes and heuristics were for formulation ($P1$) as compared with the second formulation we present computational results next.

Table 13 presents the best lower and upper bounds by the branch-and-bound after heuristics and cuts were used in both formulations. Table 14 presents the node counts and solve times for the results obtained in Table 13. The time limit was 300s, so if optimal solution is not found in the allotted time for a problem the corresponding entry for solve time is 300s and node count entry is the number of nodes explored in 300s.

Table 13: Comparison of two formulations with cutting planes and heuristics: lower and upper Bounds after 300s of computation time

<i>Prob</i>	<i>(P1)</i>		<i>(P2)</i>	
	LB	UB	LB	UB
E10_10	0.00	0.00	0.00	0.00
E10_11	6.17	6.17	4.93	6.17
E10_12	14.33	14.33	14.33	14.33
E10_13	0.00	0.00	0.00	0.00
E10_14	6.25	6.25	6.25	9.83
E10_20	50.50	50.50	50.50	50.50
E10_15	13.75	13.75	11.50	14.25
E15_10	7.75	7.75	7.75	8.00
E15_15	0.00	0.00	0.00	0.00
E15_20	5.25	6.15	5.00	6.50
E20_10	0.00	0.00	0.00	1.67
E20_15	10.00	10.00	10.00	10.00
E20_20	8.75	10.17	8.75	11.17
E20_25	15.75	17.75	15.75	17.75
E20_30	0.00	0.00	0.00	0.00
E25_20	0.00	0.00	0.00	0.00
E25_25	0.00	7.83	0.98	9.67
E25_30	0.00	0.00	0.00	0.00

Entries in **bold** represent that optimal solution was found in 300 second

Looking at the results from Table 13, we can see that, with the help of cuts and heuristics, formulation ($P1$) was able to provide better results than ($P2$). More problems were solved to optimality and for except one, the bounds provided for the problems not solved to optimality in allotted time by formulation ($P1$) were stronger than formulation ($P2$). Results from Table 14 show that even though more problems were solved with ($P1$) in the allotted time, but the problems that were solved by ($P2$) were solved faster than by ($P1$).

Table 14: Comparison of two formulations with cutting Planes and heuristics: node counts and computation times

<i>Prob</i>	<i>(P1)</i>		<i>(P2)</i>	
	Node Count	Time	Node Count	Time
E10_10	0	4.01	0	0.30
E10_11	762	16.47	115551	300.00
E10_12	19	7.78	9909	28.60
E10_13	0	5.06	0	1.05
E10_14	434	20.11	72053	300.00
E10_20	1655	40.43	67	5.60
E10_15	5	7.12	62165	300.00
E15_10	16150	68.32	199389	300.00
E15_15	0	4.72	0	3.10
E15_20	4388	300.00	10	300.00
E20_10	3766	106.49	194930	300.00
E20_15	5	10.25	36	1.56
E20_20	8855	300.00	41642	300.00
E20_25	1527	300.00	26179	300.00
E20_30	3	15.19	0	5.14
E25_20	0	6.80	0	1.38
E25_25	3438	300.00	16730	300.00
E25_30	0	13.91	0	2.16

Entries in **bold** represent that optimal solution was found in 300 second

We believe that the cutting planes derived for formulation $(P1)$ were very effective in providing good bounds. Since we did not perform an extensive polyhedral study for formulation $(P2)$, the overall performance of $(P2)$, with cutting planes, was not as strong as $(P1)$. But the results do suggest the potential for improvement if a more detailed study of formulation $(P2)$ is done.

We finish off this section with a note on whether either of these formulations $(P1)$ and $(P2)$ can be used to model the equal-split structure in multi-commodity flow problems or OSPF weight setting problem. The general multi-commodity flows equal-split problem can be modeled as per formulation $(P1)$ but we can not use $(P2)$ to model equal-split in the general problem, because $(P2)$ assumes that incoming flow into every node is fixed, which is not the case in general networks.

4.7 *Final Thoughts*

We studied the equal-split problem on bipartite graphs and presented two different integer programming formulations for it. We studied, in detail, the polyhedral structure of the first formulation ($P1$) and demonstrated with computational experiments that the cutting planes developed were very effective in solving and providing good bounds for this problem. For the second formulation ($P2$), we provided a limited polyhedral results. Next we performed a comparison between the two formulations on the effectiveness of solving the problem. Computational results indicated that the second formulation ($P2$) as-is (with no additional cutting planes or heuristics) was more effective in solving the problem than ($P1$) as-is. When the proposed heuristics and cutting planes were added to both the formulations, ($P1$) performed better than ($P2$).

Because of the nature of the objective function, cutting planes derived from the equal-split structure without taking into account the capacities of the workers were not as effective as the ones that used the capacities. The cutting planes developed performed very well in terms of solving the problems faster and providing good bounds but we feel that in order to do better we need to study the structure of incoming flows to the sources (b) compared to the capacities of the workers (c). We performed a somewhat similar study in Section 4.4, but it is not clear how to do this in general.

The second formulation ($P2$) seems very promising from the computational results presented. A detailed polyhedral study of this formulation would lead to more significant improvements. Since the basic structure both formulations try to model is the same, so the cutting planes already derived for the first formulation ($P1$) should also be valid for the second formulation ($P2$) but in a different space. How the families of valid inequalities derived for ($P1$) look in terms of the variables of ($P2$) can be another direction of further study.

CHAPTER V

CONCLUSIONS

5.1 Summary of work done

We performed an extensive polyhedral study of networks with equal-split structure. We started by studying the multi-commodity network flow problem with equal-split. We considered single node, two node and multiple node subsystems of this problem and developed valid inequalities based on these subsystems. We incorporated these valid inequalities in a branch-and-cut scheme to solve the problem. Computational experiments, performed on randomly generated problems, indicated that the cutting planes developed strengthened the mixed integer linear programming formulation. On average, computation time for our data set decreased by a factor of 30 and node counts decreased by a factor of 90, when compared with default CPLEX 9.0. We also performed extensive computational experiments to test the relative strength of the various cutting plane families with respect to each other.

Next, we considered the OSPF weight setting problem. We proposed a mixed integer linear programming formulation for the problem. A decomposition based approach was presented as a solution methodology for the weight setting problem. The approach did not provide good results, so we developed valid cutting planes by considering the specific structure of the problem to strengthen the linear programming relaxation. Heuristic methods for getting solutions to hot start the branch-and-bound method were also proposed. We tested our solution methodology on some randomly generated instances and some instances taken from the literature. Computational results show significant improvements over default CPLEX, and our solution methodology was able to provide solutions with much better quality guarantees as compared to other approaches. Also, we implemented two heuristic methods proposed in the literature and CISCO's recommendations on how the weights should be set in a network. We compared the results provided by these procedures with the results from heuristic methods proposed by us. Results indicate that our methods

outperformed all the other methods in almost all the cases.

Finally, we considered an equal-split flow problem on a bipartite graph. We provided a different formulation for the equal-split structure for this problem since the original formulation did not perform well. We also considered simple subsystems of this problem and performed polyhedral study to generate cutting planes for the problem. Heuristic methods based on the integer programming formulations were proposed. Computational experiments performed on some randomly generated instances show that the cutting planes we proposed were very effective in closing the optimality gap. When compared with default CPLEX we were able to solve more problems and for the problems that were not solved to optimality, our approach provided solutions with much better optimality gap. An alternate formulation for modelling the equal-split structure in the bipartite graph was proposed next. A convex hull property for a subsystem of this formulation was proved. A comparison between the two formulations, in terms of how effective they were in solving the equal-split flow problem on bipartite graphs, was provided.

5.2 Remaining Technical Issues

In this section we present some remaining technical issues and propose some research directions that might lead to improved computational results and understanding.

5.2.1 Equal-split multi-commodity network flow capacity cuts

The computational results performed to test the relative effectiveness of the various cutting plane families indicated that cutting planes derived from capacities, or structure, were the most important ones. The cutting planes derived were specific for certain special cases, and adding these to the branch-and-cut method led to very significant improvements. So we believe a more detailed study to derive cutting planes, specific to capacities, will lead to significant additional benefits.

5.2.2 OSPF weight setting symmetry

Valid inequalities derived for the OSPF weight setting problem were based on exploiting the equal-split restriction. Another component of the flow is that it has to be routed on

shortest paths, equally split among the arcs on the shortest path. We believe that better understanding of the relationship between the paths and the flow and using the information to derive valid inequalities for this problem will help solve the problems faster and provide better optimality gaps.

As discussed before there is a lot of symmetry in this problem and it is not clear how to get rid of it. Another direction for further study can be to understand the symmetry and propose solutions to remove it from the formulation or propose an alternate formulation that does not have similar issues.

5.2.3 Equal-split flow on bipartite graphs: extended polyhedral study

Because of the nature of the objective function of the flow problem on bipartite graphs cutting planes derived from the equal-split structure taking capacities of sinks into account were more effective compared with the general ones. We performed a somewhat similar study in Section 4.4, but it is not clear how to do this in general and can be one of the further directions to pursue.

Among the two formulations we proposed for this problem, the second one ($P2$) performed better, in computational tests provided, for getting solutions. For the first formulation ($P1$) a detailed polyhedral study of some subsystems was provided and valid inequalities were obtained, which were shown to be very effective in solving the problem. A similar polyhedral study was not performed for the second formulation and we believe that the cutting planes obtained from this would lead to additional benefits. Also since the structure modeled by both the formulations is the same, the cutting planes derived for the first formulation should also be valid for the second formulation but in a different space. So a further direction can be to study the relation between the two formulations and see how the various cutting plane families relate.

To conclude, we want to mention the numerous applications of the equal-split assumption. Some of them include fair allocation in cooperative games in Economics [17], embedded systems design for electronic circuits [38], coalition formation in natural resource tendering [71]. Also, it can be used in school districting to help local communities allocate students to

schools. Equal-split requirement on the flow is also assumed for throughput maximization in a multi-hop packet radio network using ALOHA [55]. We can use similar approaches to the ones presented in the thesis for solving these applications of the equal-split flow assumption.

REFERENCES

- [1] “Cisco 12000 Gigabit Switch Router.” Product overview, www.cisco.com, Feb 2000.
- [2] AHUJA, R., ORLIN, J., SECHI, G., and ZUDDAS, P., “Algorithms for simple equal flow problem,” *Management Science*, vol. 45, pp. 1440–1455, 1999.
- [3] AHUJA, R., MAGNANTI, T., and ORLIN, J., *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 2003.
- [4] ALFONSÍN, J. L. R., “On variations of Subset Sum Problems,” *Discrete Applied Mathematics*, vol. 81, pp. 1–7, January 1998.
- [5] ALI, A. I., KENNINGTON, J. L., and SHETTY, B., “The equal flow problem,” *European Journal of Operations Research*, vol. 36, pp. 105–115, 1988.
- [6] ALON, N., YUSTER, R., and ZWICK, U., “Color-coding: a new method for finding simple paths, cycles and other small subgraphs within large graphs,” in *Proceedings 26th Annual ACM Symposium on Theory of Computation*, vol. 30, pp. 326–335, ACM, 1994.
- [7] APPLEGATE, D. and THORUP, M., “Load optimal MPLS routing with $n + m$ labels,” in *Proceedings 22nd IEEE Conf. on Computer Communications*, 2003.
- [8] AWDUCHE, D. O., MALCOM, J., AGOGBUA, J., O’DELL, M., and MCMANUS, J., “Requirements for traffic engineering over MPLS,” Tech. Rep. RFC 2702, 1999.
- [9] BEAN, J., “Genetic algorithms and random keys for sequencing and optimization,” *ORSA Journal on Computing*, vol. 6, pp. 154–160, 1994.
- [10] BEN-AMEUR, W. and GOURDIN, E., “Internet routing and related topology issues,” *SIAM Journal on Discrete Mathematics*, vol. 15, pp. 18–49, 2003.
- [11] BERRY, L., KOHLER, S., STAEHLE, D., and TRANA-GIA, P., “Fast heuristics for optimal routing in IP networks,” tech. rep., 2000.
- [12] BLACK, U., *IP Routing Protocols, RIP, OSPF, BGP, PNNI and CISCO routing protocols*. Prentice Hall, 2000.
- [13] BLEY, A., “Inapproximability results for the inverse shortest paths problem with integer length and unique shortest paths,” in *Proceedings of the Second International Network Optimization Conference(INOC 2005)*, no. 1, 2005.
- [14] BLEY, A., “On the approximability of the minimum congestion unsplittable shortest path routing problem,” in *Proceedings of 11th Conference on Integer Programming and Combinatorial Optimization (IPCO 2005)*, 2005.
- [15] BLEY, A., GROTCHEL, M., and WESSALY, R., “Design of broadband virtual private networks: Model and heuristics for the B-Win,” in *DIMACS Workshop on Robust Communication Networks and Survivability*, no. 53, 1998.

- [16] BLEY, A. and KOCH, T., "Integer programming approaches to access and backbone IP-network planning." preprint ZIB ZR-02-41, 2002.
- [17] BRANZEI, R., DIMITORV, D., and TIJS, S. H., "The Equal Split-off Set for Cooperative Games." CentER Discussion Paper No. 2004-110, November 2004.
- [18] BURIOL, L. S., RESENDE, M., and THORUP, M., "Survivable IP network design with OSPF routing," *Networks*, vol. 49, no. 1, pp. 51–64, 2007.
- [19] BURIOL, L., RESENDE, M., REBEIRO, C., and THORUP, M., "A memetic algorithm for OSPF routing," in *Proceedings of the 6th INFORMS Telecom*, pp. 187–188, 2002.
- [20] BURTON, D., *On the Inverse Shortest Path Problem*. PhD dissertation, Facultes Universitaires ND de la Paix, Namur, Belgium, Department of Mathematics, 1993.
- [21] BURTON, D. and TOINT, B. P. P., "The inverse shortest path problem with upper bounds on shortest path costs," Tech. Rep. 93/03, 1993.
- [22] CALVETE, H., "Network simplex algorithm for the general equal flow problem," *European Journal of Operations Research*, vol. 150, pp. 585–600, 2003.
- [23] CARRARESI, P. and GALLO, G., "Network models for vehicle and crew scheduling," *European Journal of Operations Research*, vol. 16, pp. 139–151, 1984.
- [24] CISCO, "Configuring OSPF," 1997.
- [25] CISCO, "OSPF design guide." White Paper, 2006.
- [26] CPLEX, "ILOG CPLEX 9.0 Reference Manual." Information available at www.cplex.com, 2003.
- [27] CROWDER, H., JOHNSON, E., and PADBERG, M., "Solving large scale zero-one linear programming problems," *Operations Research*, vol. 31, pp. 803–834, 1983.
- [28] DOLAN, E. D. and MORÉ, J. J., "Benchmarking optimization software with performance profiles," *Mathematical Programming, Ser. A*, vol. 91, pp. 201–213, 2002.
- [29] EGER, K. and KILLAT, U., "Fair resource allocation in Peer-to-Peer networks," in *Proceedings 2006 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS06)*, (Calgary, Canada), pp. 39–45, 2006.
- [30] EISELE, W. L. and FRAWLEY, W. E., "Estimating the impacts of access management with micro-simulation," in *Proceedings 6th National Conference on Access Management*, (Kansas City, KS, USA), August 2004.
- [31] ERICSSON, M., RESENDE, M., and PARDALOS, P., "A genetic algorithm for weight setting problem in OSPF routing," *Journal of Combinatorial Optimization*, vol. 6, pp. 299–333, 2002.
- [32] EVEN, S., TAI, A., and SHAMIR, A., "On the complexity of timetable and mutli-commodity flow problems," *SIAM Journal on Computing*, vol. 5, pp. 691–703, 1976.
- [33] FARAGO, A., SZENTESI, A., and SZVIATOVSKI, B., "Allocation of administrative weights in PNNI," in *Proceedings of Networks*, 1998.

- [34] FELDMAN, A., GREENBERG, A., LUND, C., REINGOLD, N., REXFORD, J., and TRUE, R., "Deriving traffic demands for operational IP networks: Methodology and experience," in *IEEE/ACM Transactions on Networking*, no. 9, pp. 265–279, 2001.
- [35] FORTZ, B. and THORUP, M., "Robust optimization of OSPF/IS-IS weights," in *Proceedings International Network Optimization Conference*, pp. 225–230, October 2003.
- [36] FORTZ, B. and THORUP, M., "Increasing internet capacity using local search," *Computational Optimization and Applications*, vol. 29, pp. 13–48, 2004.
- [37] GAREY, M. and JOHNSON, D., *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: Freeman, 1979.
- [38] GIVARGIS, T. and VAHID, F., "Interface exploration for reduced power in core-based systems," in *Proceedings of the 11th international symposium on System synthesis*, (Washington, DC, USA), pp. 117–122, ACM, IEEE Computer Society, 1998.
- [39] GOLDBERG, D., *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.
- [40] HARMATOS, J., "A heuristics algorithm for solving the static weight optimization problem in OSPF networks," in *Proceedings of GLOBECOM*, 2001.
- [41] HOLMBERG, K. and YUAN, D., "Optimization of internet protocol network design and routing," tech. rep., 2001.
- [42] HUNG, C., *On the Inverse Shortest Path Problem*. PhD dissertation, Georgia Institute of Technology, Atlanta, Industrial and Systems Engineering, 2003.
- [43] JOHNSON, D., ARAGON, C., MCGEOCH, L., and SCHEVON, C., "Optimization by simulated annealing: and experimental evaluation," *Operations Research*, vol. 39, no. 1, 1991.
- [44] LAND, A. H. and DOIG, A. G., "An automatic method for solving discrete programming problems," *Econometrica*, vol. 28, pp. 493–520, 1960.
- [45] LIN, F. and WANG, J., "Minimax open shortest path first routing algorithms in networks supporting the SMDS services," in *Proceedings of IEEE International Conference on Communications*, no. 2, pp. 666–670, 1983.
- [46] MAHAJAN, R., SPRING, N., WETHERALL, D., and ANDERSON, T., "Inferring link weights using end-to-end measurements," in *ACM SIGCOMM Internet Measurement Workshop*, 2002.
- [47] MARBUKH, V., "Robust Traffic Engineering: Game theoretic perspective," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, Dec 2002.
- [48] MEYERS, C., *Network flow problems and Congestion Games: Complexity and Approximation Results*. PhD dissertation, Massachusetts Institute of Technology, 2006.
- [49] MITRA, D. and RAMAKRISHNA, K., "A case study of multiservice multipriority traffic engineering design for data networks," in *Proceedings IEEE GLOBECOM*, 1999.

- [50] MONIEN, B., "How to find long paths efficiently," *Annals of Discrete Mathematics*, vol. 25, pp. 239–254, 1985.
- [51] MOY, J., "The OSPF specification," tech. rep., 1999.
- [52] MURPHY, J., HARRIS, R., and NELSON, R., "Traffic engineering using OSPF weights and splitting ratios," in *Proceedings of INTERNETWORKING*, (Australia), 2002.
- [53] NEMHAUSER, G. and WOLSEY, L., *Integer and Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization, New York, NY: John Wiley & Sons, 1989.
- [54] NUCCI, A., SCHROEDER, B., BHATTACHARYYA, S., TAFT, N., and DIOT, C., "IGP link weight assignment for transient link failures," in *ITC18*, (Berlin, Germany), 2003.
- [55] PAPAVALASSIOU, T. D., SILVESTER, J. A., and WANG, J. L., "Throughput optimization in single commodity slotted ALOHA multi-hop packet radio networks," *Military Communications Conference*, vol. 2, pp. 377–382, Oct 1989.
- [56] PAQUET, C. and TEARE, D., "Configuring IS-IS protocol." CICS Press, March 2003.
- [57] PIORO, M., SZENTSI, A., HARMATOS, J., JUTTNER, A., GAJOWNICZEK, P., and KOZDROWSKI, S., "On open shortest path first related network optimization problems," *Performance Evaluation*, vol. 48, no. 4, pp. 201–223, 2002.
- [58] PRYTZ, M., *On optimization in design of telecommunications networks with multicast and unicast traffic*. PhD dissertation, Royal Institute of Technology, Stockholm, Sweden, 2002.
- [59] RAMAKRISHNAN, K. G. and RODRIGUES, M., "Optimal routing in shortest path data networks." Bell Labs Technical Journal 6(1), 2001.
- [60] RASTOGI, R., BREITBART, Y., and M. GAROFALAKIS, A. K., "Optimal configuration of OSPF aggregates," in *IEEE/ACM Transactions on Networking*, vol. 11.
- [61] RODRIGUES, M. and RAMAKRISHNA, K., "Optimal routing in data networks," in *Presentation at International Telecommunications Symposium (ITS), Rio de Janeiro, Brazil*, 1994.
- [62] ROSEN, E., VISHWANATHAN, A., and CALLON, R., "Multiprotocol label switching architecture," tech. rep., 2001.
- [63] SAHNI, S., "Computationally related problems," *SIAM Journal on Computing*, vol. 3, pp. 262–279, 1975.
- [64] SAKAUCHI, H., NICHIMURA, Y., and HASEGAWA, S., "A self-healing network with an economical spare channel assignment," in *Proceedings of IEEE Global Telecommunications Conference*, pp. 438–443, 1990.
- [65] SCHRIJVER, A., *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.
- [66] SHEPARDSON, F. and MARSTEN, R., "A lagrangian relaxation algorithm for the two-duty period scheduling problem," *Management Science*, vol. 26, pp. 2274–2281, 1980.

- [67] SRINATHAN, K., GOUNDAN, P., KUMAR, M. A., NANDAKUMAR, R., and RANGAN, C. P., "Theory of equal flows in networks," in *Proceedings of the Eighth International Computing and Combinatorics Conference*, no. 2387 in Lecture Notes in Computer Science, (Singapore), pp. 514–524, Springer-Verlag, 2002.
- [68] SRIVASTAVA, S., AGARWAL, G., MEDHI, D., and PIORO, M., "Determining feasible link weight systems under various objectives for OSPF networks," in *IEEE eTransactions on Network and Service Management*, no. 2(1), 2005.
- [69] THOMAS, T., "OSPF Network Design Solutions." CISCO Press, 1998.
- [70] THORUP, M. and ROUGHAN, M., "Avoiding ties in shortest path routing," Tech. Rep. 1131, 2003.
- [71] TISDELL, J., "Coalition formation in natural resource tendering." Faculty of Environmental Sciences, Griffith University.
- [72] TURNQUIST, M. and MALANDRAKI, C., "Estimated driver cost for transit operations planning," in *Joint National Meeting of ORSA/TIMS*, (Dallas), 1984.
- [73] WANG, Y., WANG, Z., and ZHANG, L., "Internet traffic engineering without full mesh overlaying," in *Proceedings of INFOCOM*, 2001.
- [74] WANG, Z. and CROWCROFT, J., "Analysis of shortest path routing algorithms in a dynamic network environment," *Computer Communication Review*, vol. 22, no. 2, pp. 63–71, 1992.
- [75] XIAO, X., HANNAN, A., BAILEY, B., and NI, L., "Traffic engineering with MPLS in the internet," *IEEE Network Magazine*, vol. 14, no. 2, pp. 28–33, 2000.
- [76] YE, T., HARRISON, D., MO, B., SIKDAR, B., KAUR, H. T., KALYANARAMAN, S., SYZMANSKI, B., and VASTOLA, K. S., "Network management and control using collaborative on-line simulation," in *Proceedings IEEE ICC*, (Helsinki, Finland), June 2001.
- [77] YE, T., KAUR, H. T., KALYANARAMAN, S., VASTOLA, K. S., and YADAV, S., "Dynamic optimization of OSPF weights using online simulation," in *Proceedings of IEEE INFOCOM*, 2002.
- [78] YUAN, D., "A bi-criteria optimization approach for robust OSPF routing," in *Proceedings IP Operations and Management*, pp. 91–98, October 2003.
- [79] YUKSAL, M., SIKDAR, B., VASTOLA, K. S., and SYZMANSKI, B., "Workload generation for ns Simulation of Wide Area Networks and the internet," in *Proceedings of Communication Networks and Distributed Systems Modelling and Simulation Conference*, (San Diego, CA, USA), pp. 93–98, 2000.

VITA

Amandeep Parmar was born in Punjab, India. After graduating from Apeejay School, in Jalandhar, 1998, he attended Indian Institute of Technology, Delhi, from where he received a Bachelor of Technology in Mechanical Engineering in June 2002. Next he enrolled in the School of Industrial and Systems Engineering at Georgia Institute of Technology in August of 2002, where he completed his Master's in Operations Research in 2003, and his doctoral research on integer programming methods for network flow problems with equal-split.