Modeling and Solving Variations of the Network Loading Problem

by

John M. Bossert
Diploma, Cambridge University (1997)
B.S.E., Princeton University (1996)

Submitted to the Sloan School of Management in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Operations Research at the Massachusetts Institute of Technology
September 2002
© 2002 Massachusetts Institute of Technology. All rights reserved.

Signature of Author ........................................... John M. Bossert
Sloan School of Management
Interdepartmental Program in Operations Research
August 16, 2002

Certified by .................................................. Thomas L. Magnanti
Thomas L. Magnanti
Dean of Engineering and Institute Professor, MIT
Thesis Supervisor

Accepted by .................................................. James B. Orlin
James B. Orlin
Edward Pennell Brooks Professor of Operations Research, MIT
Co-Director, Operations Research Center, MIT
Modeling and Solving Variations of the Network Loading Problem

by

John M. Bossert

Submitted to the Sloan School of Management on August 16, 2002, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Operations Research

Abstract

We examine three variations of a class of network design problems known as Network Loading Problems (NLP). For each variation we develop a tailored branch and bound solution approach equipped with heuristic procedures and problem specific cutting planes.

The first variation formulates a logistics problem known as Pup Matching that involves matching semitrailers to cabs that are able to tow one or two of the trailers simultaneously. Theoretically, we show that four heuristics each yield a 2-approximation, and we specify facet defining conditions for a cut family that we refer to as odd flow inequalities. Computationally, we solved to optimality 67 percent of test instances randomly generated from realistic data. The average minimum heuristic error among solved instances was 1.3 percent. Cutting planes reduced the average relative difference between upper and lower bounds prior to branching from 18.8 percent to 6.4 percent.

The second problem variation concerns three NLP generalizations (segregated, nested, and general compartments) that we refer to collectively as Compartmentalized Network Loading (CNLP). We model these problems, extend to the case of segregated compartments convex hull results of Magnanti, Mirchandani, and Vachani on single arc and three node problems, and employ the routine of Atamtürk and Rajan to efficiently separate certain (residual capacity) inequalities for all three CNLP models. On randomly generated instances, we conducted four series of tests designed to isolate the computational impact of problem parameters including graph density and model type.

The third variation, Single Commodity Network Loading (SCNLP), requires loading discrete capacity units sufficient to satisfy the demand for standard network flow (multiple source, multiple destination problem). We cast the limiting case of large capacity within the constrained forest framework of Goemans and Williamson, characterize the optimal solution to the single cut special case, and describe cutset, residual capacity, and three partition inequalities for this variation. We solved five randomly generated 15 node SCNLP instances in an average of 19.1 CPU seconds, but only three of five similarly defined NLP instances.

Thesis Supervisor: Thomas L. Magnanti
Title: Dean of Engineering and Institute Professor
to my parents
Acknowledgments

I want to first acknowledge the help and support of my adviser, Tom Magnanti. Tom has provided a great model of someone achieving much by doing what he loves, and I am proud to be among his students. I want to thank him in particular, though, for something he did not do, namely actively steering my course of study. The experience of seeking my own path more than compensates for any reduced output. I also thank the other members of my thesis committee, Cindy Barnhart and Andreas Schulz. Their rigor, and my wariness of it, improved this thesis, yet they also promoted a very friendly atmosphere for its completion. I thank Lisa Magnano Bleheen for seemingly privileged access to Tom and for always welcoming me, and I thank the current staff of the OR Center for expert navigation of the MIT bureaucracy. Finally, I thank the students of the OR Center. I hope to always work with talented and motivated people.
# Contents

1 Introduction and Literature Review ........................................... 13

2 Pup Matching ............................................................................ 21
   2.1 Introduction ........................................................................ 22
   2.2 Formulation, Notation, and Complexity ................................. 25
      2.2.1 Modeling Assumptions ............................................... 25
      2.2.2 Problem Statements .................................................. 27
      2.2.3 Integer Programming Formulations ............................... 28
   2.3 Waiting Rings ..................................................................... 30
   2.4 Initial Computations ............................................................ 37
   2.5 Heuristic Solution Approaches ............................................ 38
      2.5.1 Matching 2 Pups ......................................................... 39
      2.5.2 Matching Approximations ......................................... 41
      2.5.3 Shortest Path Heuristics ............................................. 46
   2.6 Valid Inequalities ............................................................... 48
   2.7 Computational Results ....................................................... 55

3 Compartmentalized Network Loading ........................................... 58
   3.1 Introduction ........................................................................ 59
   3.2 Notation and Formulation .................................................... 60
   3.3 Linear Programming ............................................................ 63
      3.3.1 Linear Programming Relaxation of the CNLP ................. 64
      3.3.2 Compartmentalized Multicommodity Flow ..................... 64
   3.4 Cutset Inequalities ............................................................... 67
      3.4.1 Formulation of Cutset Inequalities ................................. 68
      3.4.2 Single Edge CNLP ...................................................... 71
   3.5 The CNLP Single Arc Design Problem and Residual Capacity Inequalities ........................................... 72
      3.5.1 Segregated Compartments ............................................ 74
      3.5.2 Nested and Mixed Compartments ................................. 81
   3.6 Problems with a Single Origin-Destination Pair ....................... 86
   3.7 Three Node Network Loading ............................................... 89
      3.7.1 Single Compartment Facilities ..................................... 90
      3.7.2 Segregated Compartments .......................................... 92
      3.7.3 Two Compartments, Nested or General ......................... 93
List of Figures

2.1 A conventional tractor trailer and a "tandem" of two semitrailers known as pups. .................................................. 22
2.2 Arc length represents the cost of sending a cab as well as one or two pups from the tail node to the head node. ................. 23
2.3 Shipping network for which the NLP Pup Matching formulation fails. Node numbers and arc costs are specified. ............... 31
2.4 Optimal routing to the NLP formulation of Example 1. The solution assigns a single cab to each arc crossed by a pup path, yet each pup can advance only one arc. .................................................. 31
2.5 Schematic of a 3 pup waiting ring. .......................................................... 33
2.6 Path reassignment breaks the ring of Figure 2.5. ......................... 34
2.7 The underlying graph for several Pup Matching test problems. .......... 38
2.8 Solution of cost 196 to the problem of delivering 1 pup from the origin node to each of the other 55 nodes. The numbers indicate cab loadings. 39
2.9 General structure of an optimal solution to two Pup Matching. Each arrow represents a path. .................................................. 40
2.10 The optimal directed Steiner tree connecting destinations 2, 3, and 4 to origin 1 does not yield the optimal Pup Matching solution. ...... 43
2.11 Instance n of a sequence with an inimum Pup Matching performance ratio of 2. .................................................. 44
2.12 Family of Single Commodity Pup Matching instances with an inimum performance ratio of 2. ........................................ 45
2.13 Pup A is to be routed from 1 to 6 and pup B from 4 to 6. The performance ratio of each successive shortest path heuristic is $\frac{2}{1+\epsilon}$. .............. 47
2.14 Arc costs are shown. Routing pups A and B together from node 1 to node 4 leads to suboptimality. ................................ 48
2.15 Results of the branch and bound procedure on city blocks problems. The portion of the graph above the zero line depicts lower bound improvement from sequential application of the three cutting plane families, and the portion below the zero line indicates the error of the best heuristic. ......................... 56

3.1 One unit of flow must be sent from node 1 to node 2 in one compartment, and another unit of flow must be sent from node 1 to node 3 in another compartment. Under the specified edge costs, the shortest path solution of direct routing is suboptimal for the LP relaxation. .............. 65
3.2 Transformation of the compartmentalized multicommodity flow problem to the standard multicommodity flow problem by replacing each arc with a network structure. ........................................ 67
3.3 Transformation of an undirected multicommodity flow problem to the
standard directed version, as depicted in Ahuja, Magnanti, and Orlin
[2]. ................................................................. 67
3.4 Max flow feasibility problem to determine the right hand side of a
CNLP cutset inequality. ........................................ 69
3.5 Optimal \( z \) values to (P1) and (P2) as functions of \( b^1 \) and \( b^2 = b - b^1 \),
respectively. ...................................................... 79
3.6 Commodity disaggregation underlying the SSADP to SADP transform-
ation. The SSADP instance has two classes with commodities \( a, b, c \)
and \( i, ii, iii, iv \), respectively. The single SADP class has commodities
\( 1, 2, 3, 4, 5, 6 \). .................................................. 81
3.7 Commodity subsets defined by compartment subsets \( S \) and \( T \). .......... 86
3.8 Branch and bound solution time vs. graph density for test instances
of 10 and 12 nodes. ............................................. 101
3.9 Solution time vs. number of compartments for complete graphs of 8
and 10 nodes. .................................................... 102
3.10 Final integrality gap vs. number of compartments for complete graphs
of 8 and 10 nodes. ............................................... 102
3.11 Solution time vs. demand density. ................................ 103
3.12 Final integrality gap vs. demand density. .......................... 104
3.13 Solution times and average number of branch and bound nodes for
each facility type. ............................................. 104
3.14 Average integrality and heuristic gap for each facility type. .......... 105

4.1 The BDSF2 LP relaxation of this instance strictly contains the BDSF1
LP relaxation. ..................................................... 118
4.2 The LP round up heuristic provides a solution of cost \( C \). The optimal
solution sends all flow via node 1 for a cost of \( 1 + (C - 1)c \). ........ 122
4.3 Graph of Example 7. Cutset inequalities do not imply three partition
inequalities. ....................................................... 125
4.4 Instance illustrating that odd flow inequalities are not valid for the
multicommodity NLP on an undirected graph, even for nonbifurcated
flow. ............................................................... 127
4.5 Transformation of the LP relaxation of the Single Cut Problem to a
directed shortest path problem. Edge and arc labels indicate flow costs. 129

A.1 Pup Matching Graph to solve the Three Dimensional Matching in-
stance illustrating the 3DM-PMD transformation. Nonzero arc costs
are shown. ......................................................... 152
A.2 A gimmick controlling token flow to effectively assign a variable either
ture or false. ..................................................... 155
A.3 A gimmick effectively transmitting a variable assignment to those clauses the assignment makes true. ............................... 156
A.4 A gimmick to control the potential activation of clause $C = L_1 \lor L_2 \lor L_3$ by any one of its three constituent variables. .................. 157
A.5 An example of a SAT to WR transformation. ..................... 160
List of Tables

4.1 Computational results, full formulation of SCNLP .................. 135
4.2 Computational results, cut formulation of SCNLP .................. 136
4.3 Computational results, NLP formulation ......................... 137
Chapter 1

Introduction and Literature Review

This thesis considers three variations of a class of network design problems known as Network Loading (NLP). This chapter introduces the Network Loading Problem and reviews some relevant research literature. In Chapter 2 we formulate as a special case a logistics problem known as Pup Matching. Chapter 3 examines generalizations of the NLP through three models that we refer to collectively as Compartmentalized Network Loading. Network Loading problems usually have an underlying multicommodity flow structure with point-to-point demand for each commodity. Chapter 4 develops a single commodity variation of the problem with multiple sources and destinations. Throughout Chapters 2 - 4 we emphasize valid inequalities and their computational use in branch and bound solution approaches. Chapter 5 concludes by summarizing our contributions and posing some questions for further research.

The Network Loading Problem models capacitated network design characterized by the installation or loading of edge capacity in discrete units or facilities. More specifically, an NLP is characterized by a discrete set of capacity values, each corresponding to a different facility type, with a loading of an integral number of facilities
of each type on each edge of an underlying network. The composite capacity of a feasible solution permits simultaneous flow of given demands, and we are to minimize the combined facility installation and flow costs.

A number of practical problems fit within the network loading framework. Telecommunications network design or capacity expansion is probably the most apparent and studied application. See, for example, [35], [7], [45]. Designers can install capacity on a link in various discrete increments, each corresponding to a different facility such as an Optical Carrier 1 (OC-1), worth 51.85 megabits per second, or an OC-3, worth 155.52 Mbps. Similarly, we can cast certain logistics problems as network loading since shipping capacity, such as tractor trailers, occurs in discrete quantities. Chapter 2 considers one such application.

Different applications lead to different models, and we might consider a host of variations. The underlying network and the capacity might each be directed or undirected. A single facility NLP involves only one capacity value. Demand in a single commodity problem can be satisfied by flow originating at any source, while a multi-commodity problem pairs a priori origin and destination nodes. That is, each demand must be satisfied by flow from a prespecified origin.

We can formulate a single facility multicommodity NLP on an undirected network with no flow costs as follows.

\[
\min \sum_{(i,j) \in A} b_{ij}z_{ij}
\]

subject to:

\[
\sum_{j \in N} f_{ji}^k - \sum_{j \in N} f_{ij}^k = \begin{cases} -u^k, & \text{if } i = O^k \\ u^k, & \text{if } i = D^k \\ 0, & \text{otherwise} \end{cases}, \forall k \in Q, \forall i \in N
\] (1.1)

\[
\sum_{k \in Q} (f_{ij}^k + f_{ji}^k) \leq Cz_{ij}, \forall (i,j) \in A
\] (1.2)
\[ z_{ij} \geq 0, \text{ integer, } \forall (i, j) \in A \] (1.3)
\[ f_{ij}^k \geq 0, \forall (i, j) \in A, \forall k. \] (1.4)

Node set \( N \) and edge set \( A \) define the underlying graph. \( Q \) is the set of commodities, and \( u^k \) is the demand for commodity \( k \in Q \). Demand for commodity \( k \) must flow from origin node \( O^k \) to destination \( D^k \). Each facility provides capacity \( C \), and variable \( z_{ij} \) represents the number of facilities loaded on edge \( (i, j) \in A \), each at cost \( b_{ij} \). Variable \( f_{ij}^k \) represents the flow of commodity \( k \) from node \( i \) to node \( j \). Constraints (1.1) enforce commodity flow balance, and constraints (1.2) require sufficient capacity on each edge. Constraints (1.3) and (1.4) impose nonnegative integrality and nonnegativity on the decision variables. The objective is the cost of loaded facilities.

Rothfarb and Goldstein [41] considered a relative of the NLP that they referred to as the Telpak Problem. They modeled a network flow problem with nonconvex piecewise linear flow costs on each arc. Epstein [19] outlined a similar flow formulation of the NLP that eliminates the facility or design variables but involves step flow cost functions with jumps corresponding to combinations of facility capacities. Rothfarb and Goldstein cast the single origin Telpak Problem as a linear program with adjacency conditions (see, for example, Bradley, Hax, and Magnanti [15]) and developed a procedure to obtain a local optimum.

Gavish and Altinkemeci [22] formulated as a nonlinear mixed integer program a backbone network design problem that is similar to the NLP since it assigns simultaneously capacity and flow routes. The nonlinearity arose from the incorporation of delay costs corresponding to link congestion. They required nonbifurcated flow and employed a path formulation with cut constraints. Their Lagrangian relaxation
solution approach decomposes the problem and achieved Lagrangian duality gaps on the order of 10% on test problems of about 30 nodes.

Leung, Magnanti, and Singhal [31] modeled a point-to-point route planning problem as a mixed integer nonlinear program. They proposed a heuristic solution approach that decomposes the problem into two subproblems. The "routing" subproblem is a network loading variation with both loading and flow costs. The authors approached the subproblem via Lagrangian relaxation and reported modest success in a small computational test of the subproblem.

Magnanti and Mirchandani [33] arguably initiated the polyhedral study of the Network Loading Problem. Although they did not refer to it as such, the authors presented the NLP with a single origin and destination as a generalization of the shortest path problem. The authors solved the single facility problem and "common break-even point" versions of the two-facility and three-facility problems by sufficiently loading a shortest path between the origin and destination nodes. They also showed that versions of the two facility loading problem are strongly \( \mathcal{NP} \)-hard. In chapter 3, we extend their analysis to the problems we name Compartmentalized Network Loading.

Magnanti, Mirchandani, and Vachani [34] defined the convex hull of two special cases of the single facility NLP. The first is the single arc problem with both flow and design costs that arises as a Lagrangian subproblem under relaxation of the balance constraints (1.1). The authors introduced a family of cuts that they referred to as arc residual capacity inequalities and showed that their addition to this single arc problem defines its convex hull. The second special case is the single facility problem
on a graph of only three nodes. The authors applied the Gomory-Chvátal rounding procedure to cut conditions to define another family of valid inequalities that they referred to as three partition inequalities. They showed that these inequalities, as well as the cut condition for each of the three nodes, define the convex hull of the problem projected onto the space of the design variables $z$. In chapter 3, we extend these results to the compartmentalized problems, and, in chapter 4 we adapt them to a single commodity variation of the NLP.

In a companion paper, Magnanti, Mirchandani, and Vachani [35] developed a branch and bound solution approach for the two facility Network Loading Problem that employs the residual capacity and three partition inequalities, as well as cutset inequalities. They stated facet defining conditions for both the residual capacity and cutset inequalities and also proved that addition of the residual capacity inequalities yields the same lower bound as the Lagrangian problem defined by relaxation of the flow conservation constraints. An extensive computational test on problems of up to 15 nodes illustrates the ability of the three cut classes to reduce integrality gaps.

Bienstock and Günlük [13] developed a cutting plane and branch and bound approach for a telecommunications capacity expansion problem that they modeled as a two facility network loading variation with directed flow and bidirected capacity. (That is, a facility permits both forward and reverse flow, and flow in one direction does not reduce capacity for flow in the other.) They described three families of valid inequalities. Their cut-set inequalities are similar to those developed in [35]. Flow-cut-set inequalities generalize cut-set inequalities to include nonzero coefficients on flow variables as well as design variables, one arises for each partition of the arcs.
defining a cut. The third family of inequalities includes several variations of three partition inequalities. Bienstock and Günlük showed that one variation defines the convex hull of the feasible facility loadings for the single facility version of their problem on $K_3$, the complete three node graph, to achieve a result similar to that of [34]. The authors reported that the cutting planes significantly reduced inequality gaps and branch and bound solution times on problems derived from two real data sets.

Bienstock et al. [12] considered two solution approaches for a single facility NLP with no flow costs on a directed graph with directed capacity. The first approach uses cutset, flow-cutset, and 3 partition inequalities in combination with branch and bound. The second approach uses the metric inequalities of Onaga and Kakusho [38] to project the problem to only the design or facility variables. The latter formulation includes infinitely many constraints. Bienstock et al. outlined several heuristics to identify violated cuts and so obtain a lower bound on the solution value. They also used heuristics to obtain an upper bound. Computational tests indicated that the approaches are fairly comparable, though the full formulation approach seemed superior on larger problems. In chapter 4, we consider the analogous approaches to our single commodity NLP variation, but we have obtained only bounds from our formulation that includes design variables only.

Stoer and Dahl [45] formulated a network loading variation with survivability constraints. The model contains only binary design variables, each corresponding to a capacity increment, and uses metric inequalities to ensure capacity sufficient for multicommodity flow. The authors developed a set of cuts similar to knapsack cover inequalities that they referred to as band inequalities. They derived other inequalities
from connectivity requirements and reported preliminary computational results from a cutting plane algorithm.

Mirchandani [37] used projection to derive a formulation with finitely many constraints of a two facility NLP in terms of only the design variables. The formulation contains a metric inequality for each extreme ray of the projection cone. Mirchandani also derived several classes of facets of the polyhedron defined by his projection formulation.

Barahona [7] considered both bifurcated and nonbifurcated (or splittable and unsplittable flow) versions of the single facility NLP with only design costs. He too developed a solution approach that initially considers only design variables, though this formulation, based on cutset inequalities, is designed to find only a network loading that yields a good lower bound. He used a full formulation to then find provably good feasible solutions.

Atamtürk and Rajan [5] studied the polyhedral structure of single arc, single facility variants of the NLP. They presented a linear time separation routine for the residual capacity inequalities of [34]. They also introduce two new classes of inequalities applicable to nonbifurcated versions of the NLP. In chapter 4, we employ this procedure to solve the residual capacity inequalities of the compartmentalized problems. Atamtürk [4] studied the polyhedral structure of the network loading subproblem on directed graphs corresponding to a single cutset. He showed that a class of inequalities introduced by Chopra, Gilboa, and Sastry [17] defines the convex hull of the subproblem for the single commodity - single facility case, and generalized the inequalities for the single commodity - multifacility and multicommodity - multifacility
Epstein [19] focused on several heuristic procedures to obtain feasible solutions to the Network Loading Problem. He bounded the error of the simple heuristic of rounding up facility loadings of the LP relaxation and developed two heuristics that output tree solutions. Each tree heuristic provides an $O(n)$ approximation, when the node set has cardinality $n$. In a computational test, Epstein compared these three heuristics and the $O(log(n))$ approximation algorithm of Mansour and Peleg [36] that loads all capacity on a subgraph known as a light-weight, low-stretch spanner. Mansour and Peleg showed that unless $\mathcal{P} = \mathcal{NP}$, this approximation is the best possible, up to a multiplicative constant. In chapter 4, we apply their analysis to the single commodity NLP. Epstein’s tests indicated that no heuristic is dominant, although the spanner heuristic appears most robust.

Agarwal [1] described a relatively sophisticated local search heuristic for the NLP. The procedure iteratively improves an initial solution by rerouting flows associated with a single edge per the solution to a dynamic programming subproblem. He reported solutions within 1% of optimal for instances with several facilities defined on sparse graphs of 50, 75, and 99 nodes.

Salman et al [42] and Hassin, Ravi, and Salman [27] described constant factor approximation algorithms for the single source, single facility, nonbifurcated NLP, and Guha, Meyerson, and Munagala [26] and Talwar [46] described constant factor approximations for the single source, multi-facility NLP. Very recently, Ravi and Sinha [40] developed approximation algorithms for several $\mathcal{NP}$-hard logistics problems that integrate elements of facility location and network loading.
Chapter 2

Pup Matching

This chapter considers the logistics problem of Pup Matching as a special case of Network Loading. Section 2.1 introduces and motivates the problem. Section 2.2 describes modeling assumptions that reduce the real problem to a concise combinatorial optimization problem, and then develops an incomplete integer programming formulation as a special case of the NLP. Section 2.3 shows that infeasibilities we describe as waiting rings fully account for the incompleteness of the NLP formulation. Section 2.4 accounts our initial computational difficulties with branch and bound to motivate the need for the problem specific heuristics and cuts that we describe in Sections 2.5 and 2.6, respectively. Finally, Section 2.7 summarizes computational results from both test problems defined on grid-like graphs and problems based upon realistic data. The results illustrate the ability of the cuts to reduce integrality gaps and their corresponding importance to provably solving the NLP formulation of Pup Matching.
2.1 Introduction

Trucking is a large industry. The Department of Transportation reported that in 1998 the U.S. trucking industry had revenues of just under $200 billion, and its 7.7 million trucks carried over a trillion ton-miles of freight. Therefore, even modest percentage gains in operational efficiency can translate into substantial monetary savings.

Most tractor trailers consist of a cab and a single trailer about 48 feet long, but some cabs can accommodate in tandem up to two relatively short semitrailers, each about 28 feet long, known as pups. See Figure 2.1. In these situations, the cost to a carrier of towing two pups from one location to another is essentially the same as that of towing just one along the same route, half that of towing either three or four, and so forth. Pup matching is the problem of minimizing these stepwise discontinuous costs by matching or pairing pups behind cabs in the most efficient manner.

![Diagram of a conventional tractor trailer and a "tandem" of two semitrailers known as pups.]

Figure 2.1: A conventional tractor trailer and a "tandem" of two semitrailers known as pups.

As an example, in the shipping network represented in Figure 2.2, the arc lengths represent the cost of sending a cab towing one or two pups from the terminal represented by the tail node to the terminal represented by the head node. Suppose that a carrier must send one pup from node 1 to node 4 and a second pup from node 2 to
node 4. If each cab could tow only one pup, it would be optimal to send each pup along its shortest path and incur a cost of 5 for each. However, since pups can be paired, the carrier can achieve the optimal cost of 9 by sending both pups singly to node 3 and then pairing them to the same cab along the arc from node 3 to node 4.

Figure 2.2: Arc length represents the cost of sending a cab as well as one or two pups from the tail node to the head node.

Pups provide not only increased towing capacity over conventional tractor-trailers, but also greater flexibility through options to shift pups among cabs. The problem of optimally exploiting this flexibility seems worth studying.

Barnhart and Ratliff [9] modeled and efficiently solved two different truck/rail intermodal trailer routing problems. Both problems consider full length trailers. However, the latter resembles pup matching since its rail costs are per flatcar, and each flatcar can accommodate up to two trailers. Each origin-destination path, though, includes at most one rail segment. Consequently, each trailer travels paired with at most one other trailer, and a weighted matching algorithm can solve the problem. The matching problems that we formulate permit each trailer to pair with a different trailer over each arc of its O-D path, and direct application of a matching algorithm cannot solve the problem. However, the Matching Approximation that we introduce in Section 2.5 is similar to the solution technique of Barnhart and Ratliff.
Barnhart and Kim [8] developed an integer programming formulation of a specialized pup matching problem they referred to as the core inter-group line-haul problem. This problem involves construction of cyclic driver routes to service trailer pickups and deliveries at the end-of-line terminals associated with a single consolidation center within a logistics network. That is, drivers, or, equivalently, cabs, must be routed over circuits within the network so that the corresponding towing capacity permits each pickup trailer to advance from its origin to the consolidation node and each delivery to advance from the consolidation node to its destination node. The objective is linear in the number of cabs traversing each arc. Barnhart and Kim proposed an approximate solution approach that uses two weighted matching subroutines, and they demonstrated the effectiveness of this approach using both randomly generated data and data provided by a large LTL (less than truckload) carrier. Both their formulation and solution approach permit infeasibilities that we describe as waiting rings in Section 3.2.

Li, McCormick, and Simchi-Levi [32] considered a class of problems more general than Pup Matching that they referred to as point-to-point delivery and connection problems. The problems involve sending a single item from each of \( p \) origins to \( p \) destinations. Up to \( C \) items at once can share each unit (typically a truck) of capacity loaded on an arc, and costs are linear in the number of units loaded. Pup matching under our assumptions corresponds to the special case of \( C = 2 \). The authors considered problems with predefined and unfixed origins and destinations, and on directed and undirected graphs. They also considered the special cases with large values of \( C \) – the problems of connecting origins and destinations as cheaply
as possible. The authors showed that all variations are strongly $\mathcal{NP}$-hard, and they described a polynomial time algorithm for the special case of point-to-point delivery with a fixed value of $p$.

2.2 Formulation, Notation, and Complexity

This section outlines assumptions we used to reduce the real pup matching problem to a concise problem statement. Section 2.2.1 describes the modeling assumptions, and Section 2.2.2 states the resulting problem in instance-problem format. Section 2.2.3 presents an initial but incomplete integer programming formulation.

2.2.1 Modeling Assumptions

We assume that the motor carrier in question operates on a well defined logistics network that is adequately summarized as a directed graph with a known cost of sending a cab and driver, as well as one or two pups, along each arc of the network. We assume these costs either include or dominate all other relevant costs, including those incurred switching pups from one cab to another. We also assume that each pup is closed before leaving its origin, not opened until reaching its destination, and that the carrier is concerned only with the costs of transporting the closed pups. That is, our problem addresses no load consolidation issues.

The preceding assumptions restrict the scope of the problem. We also make several simplifying assumptions. First, we ignore any time constraints imposed upon the shipment of the pups, and search for the minimum cost shipping strategy that sends
the pups to the required destinations. Additionally, we ignore constraints on driver and cab resources such as driver availability and cab rebalancing. We effectively assume immediate availability of a loaded cab at each arc tail node, and, in turn, that the carrier can move a pup along any outgoing arc of its current node for no cost other than that attributed to moving along the arc, the marginal cost of which might be 0.

The model of Barnhart and Kim [8] requires cyclic routing of each cab and enforces trailer balance at each node. A model might require such constraints to satisfy driver work rules or to ensure a longer term deployment of resources capable of meeting future shipping requirements. The adequacy of our simplifying assumptions depends on the application, but, hopefully, our models capture at least a core structure common to this family of applications.

Within our modeling framework, we might consider two problem variations. The first requires shipment of a pup between a specified origin-destination pair. The second variation requires that each destination node receive one or more pups, but without regard to their origin, perhaps because each pup contains the same commodity. This second variation identifies but does not pair origin and destination nodes. We consider the former variation the primary case, and focus mostly on it in the remainder of this chapter. Chapter 4 considers a generalization of the second variation.
2.2.2 Problem Statements

The problem statements refer to the collective towing capacity allocated over a network as a “loading.” A feasible loading permits specification of an origin-destination path for each pup, and for each arc of such a path, an indication of another pup, if any, that travels with it behind the same cab. We term such a specification a routing. A routing includes both paths and pairs. Feasibility of a loading and an accompanying routing corresponds to the existence of a dispatching sequence of the loaded cabs that implements the pup routing. We refer to two pups assigned to traverse one or more arcs together as pairs or matches. We use the latter two terms interchangeably.

The preceding assumptions lead to the following problem statements.

**Pup Matching (PM)**

**Instance:** A directed network \( G = (N, A) \), a set \( K \) of pairs of elements of \( N \), and a cost function \( c : A \rightarrow R^+ \).

**Problem:** Find the minimum cost capacity loading of \( G \) that permits a multicommodity flow with one unit flow from the first to the second node of each of the pairs \( K \). Each unit of capacity loaded on arc \( a \in A \) costs \( c(a) \) and permits 1 unit of flow or 2 units flowing together to traverse arc \( a \).

**Single Commodity Pup Matching (SCPM)**

**Instance:** A directed network \( G = (N, A) \), sets \( K_o \) and \( K_d \) of elements from \( N \) satisfying \( |K_o| = |K_d| \), and a cost function \( c : A \rightarrow R^+ \).

**Problem:** Find the minimum cost capacity loading of \( G \) that permits flow satisfying unit supply at each node in \( K_o \) and unit demand at each node in \( K_d \). One unit of loading on \( a \in A \) costs \( c(a) \) and permits one unit of flow or 2 units together to traverse arc \( a \).

The “togetherness” requirement in the problem statements reflects the fact that two units of flow must be available on an arc simultaneously to be able to use a single unit of loaded capacity. (See our discussion of waiting rings in Section 2.3.)
Since the problem statements permit a pup to be matched to more than one other pup and over more than one arc, matching costs are not well defined, and we cannot solve the problems by directly applying a weighted nonbipartite matching algorithm. In fact, Pup Matching is at least as hard as Three Dimensional Matching and so \( \mathcal{NP} \)-complete.

**Theorem 1** Pup Matching, posed as the decision problem of determining whether a feasible cab loading with cost no greater than a specified value exists, is \( \mathcal{NP} \)-complete.

**Proof.** See Appendix A.1 or [32].

The proof of Theorem 1 in the appendix shows that Pup Matching remains \( \mathcal{NP} \)-complete when restricted to the case of a single origin. Furthermore, since Pup Matching and Single Commodity Pup Matching are equivalent in this single origin case, \( \mathcal{NP} \)-completeness of Single Commodity Pup Matching follows immediately.

**Corollary 1** Single Commodity Pup Matching posed as the decision problem of whether some feasible solution does not exceed a specified cost is \( \mathcal{NP} \)-complete.

### 2.2.3 Integer Programming Formulations

We formulate Pup Matching as a special case of the Network Loading Problem that casts pups in the role of commodities and cabs in the role of capacity providing facilities. The model includes the following data:

- \( G = (N, A) \) : the shipping network,
- \( c_{ij} \) : cost to send one cab, as well as one or two pups, on arc \((i, j) \in A\),
- \( O^k, D^k \) : origin and destination nodes, respectively, for pup \( k, k = 1, 2, \ldots K \),

and the following variables:

- \( f_{ij}^k \) : binary variable, with a value of 1 indicating that pup \( k \) is routed on arc \((i, j)\),
\( z_{ij} \): integer variable, the number of cabs assigned to arc \((i, j)\).

Using this notation, we formulate the model as follows.

**NLP formulation of Pup Matching**

\[
\min \sum_{i,j \in A} c_{ij} z_{ij}
\]

subject to:

\[
\sum_{j \in N} f_{ij}^k - \sum_{j \in N} f_{ji}^k = \begin{cases} 
1, & \text{if } i = O^k \\
-1, & \text{if } i = D^k \\
0, & \text{otherwise}
\end{cases}, \forall i \in N, k = 1, 2, \ldots K, \tag{2.1}
\]

\[
\sum_{k \leq K} f_{ij}^k \leq 2z_{ij}, \forall (i, j) \in A \tag{2.2}
\]

\[
z_{ij} \geq 0, \text{ integer, } \forall (i, j) \in A \tag{2.3}
\]

\[
f_{ij}^k \text{ binary, } \forall (i, j) \in A, k = 1, 2, \ldots K \tag{2.4}
\]

The objective minimizes cab loading cost. Constraints (2.1) enforce pup flow balance for each pup at each node, and constraints (2.2) require sufficient arc capacity. Constraints (2.3) and (2.4) enforce nonnegative and binary integrality, respectively.

The model for Single Commodity Pup Matching includes the same data, but each pup is not assigned an origin-destination pair \(\text{à priori}\). That is, a pup must leave each node \(O^k\) and a pup must arrive at each node \(D^k\), but not necessarily the same one.

The model uses the following variables:

- \(f_{ij}\): nonnegative integer variable, number of pups routed on arc \((i, j)\),
- \(z_{ij}\): nonnegative integer variable, number of cabs assigned to arc \((i, j)\),

and is formulated as:

**NLP formulation of Single Commodity Pup Matching**

\[
\min \sum_{i,j \in A} c_{ij} z_{ij}
\]

subject to:

\[
\sum_{j \in N} f_{ij} - \sum_{j \in N} f_{ji} = s_i, \forall i \in N \tag{2.5}
\]

\[
f_{ij} \leq 2z_{ij}, \forall (i, j) \in A \tag{2.6}
\]
\[ z_{ij} \geq 0, \text{integer}, \forall (i, j) \in A \] \\
\[ f_{ij} \geq 0, \text{integer}, \forall (i, j) \in A. \]  

In this model, \( s_i \) is the net supply of pups at node \( i \). The objective again minimizes the loading cost, and constraints (2.5) and (2.6) enforce pup flow balance and arc capacity. In this case, both sets of variables are nonnegative integers, though integral loadings \( z_{ij} \) and the underlying network flow structure yield gratis integral flows \( f_{ij} \).

### 2.3 Waiting Rings

The NLP formulation fails to explicitly enforce the constraint that both capacity units of a cab loading be used together, since it permits two pups traversing an arc separately to each exhaust one unit of capacity. More specifically, the flow variables \( f \) define pup paths but not pairings, so a solution to the NLP formulation typically corresponds to many routings, and the formulation implicitly assumes that two pups assigned to the same arc can always be matched to a single cab. Example 1 illustrates that this assumption is not necessarily valid, and that, as a consequence, it might not be possible to implement a Pup Matching solution for the optimal cost of its NLP formulation.

**Example 1** 3 pups must be redistributed in a network with topology and arc costs as shown in Figure 2.3. Pup A is to travel from node 1 to node 3, pup B from node 6 to node 2, and pup C from node 5 to node 4.

Figure 2.4 depicts an optimal routing determined by the flow variables of the NLP Pup Matching formulation that requires only 1 cab on each arc crossed by a pup path. When pup A reaches node 2, it must wait for pup C if the routing is to
Figure 2.3: Shipping network for which the NLP Pup Matching formulation fails. Node numbers and arc costs are specified.

Figure 2.4: Optimal routing to the NLP formulation of Example 1. The solution assigns a single cab to each arc crossed by a pup path, yet each pup can advance only one arc.
be implemented for the loaded capacity. Similarly, when pup C arrives at node 3, it must wait for B. Finally, when pup B arrives at node 4, it must wait for A. No pup can advance more than a single arc. Breaking this gridlock requires allocation of additional cabs. The following definition generalizes this class of infeasibilities.

**Definition 1** Suppose that a pup A has arrived at some node but cannot advance along its assigned path until its assigned pair, pup B, for the next arc of that path has also arrived. Suppose further that B must wait at its present node until some other pup, C, has arrived, and similarly, pup C must wait for pup D, pup D for pup E . . . pup Q for pup R. If this precedence chain closes in the sense that pup R waits upon pup A, none of the pups in the chain can advance according to the assigned routing, and the routing is thus infeasible. We refer to the pups involved in this gridlock and the portion of each such pup's origin-destination path between the node where it waits and the node where it completes travel with the pup that waits on it, as a **waiting ring**. The waiting ring of Figure 2-4 is defined by pups A, B, and C, and their subpaths among nodes 2, 3, and 4.

A waiting ring is a property of a routing and is independent of the dispatch sequence and travel times. In Example 1, no matter how quickly pup A arrives at node 2 relative to pups B and C, it cannot advance according to the assigned routing until pup C arrives at node 2, and pup C never arrives at node 2. Also, by construction, the pups forming a ring are distinct. A ring closes upon hitting a particular pup a second time when tracing back waiting relationships as in the definition. If we were to continue tracing back the relationships, the chain might again close, but it would then define a different, perhaps overlapping, ring. The time constraints of Li, McCormick, and Simchi-Levi [32] eliminate waiting rings. However, their paper did not formulate these constraints mathematically, as it did not formulate the point-to-point delivery problems as network loading problems.

The waiting relationships in a ring are not necessarily as direct as those shown in
the example. Figure 2.5 shows schematically a more general three pup waiting ring. Pup A enters the ring at node 1 and exits at node 4; B enters at node 3 and exits at node 6; and C enters at node 5 and exits at node 2. Two pups might be matched along several successive arcs. In the figure, A and C are matched from node 1 to node 2, but might pass through several other nodes on the way. Also, a pup in the ring might be more than one arc or matching away from the pup upon which it waits, as represented by the squiggly lines in the figure. Pup A waits at node 1 for C, which cannot advance because it waits for B, but C is not available to join A directly upon leaving B.

![Diagram of a 3 pup waiting ring](image)

**Figure 2.5:** Schematic of a 3 pup waiting ring.

Pups in Single Commodity Pup Matching are interchangeable. The following lemma shows that this interchangeability permits path reassignments that remove waiting rings without incurring additional cost.
Lemma 1 Waiting rings arising in Single Commodity Pup Matching can be eliminated without an increase in cost.

Proof. Each pup involved in a ring can effectively advance by maintaining its current origin to ring subpath and taking the destination node and corresponding ring to destination subpath previously assigned to the pup in the ring upon which it waits. Figure 2.6 depicts the effect of this reassignment for pups A, B, and C on the ring of Figure 2.5, with, for example, pup A leaving the ring on the subpath previously assigned to pup C.

This reassignment breaks the ring since each involved pup can advance singly to the node from which it leaves the ring, and, similarly, any pair of the pup along the excised portion of its subpath (a squiggly path in Figure 2.5) can also advance singly. Furthermore, the cab requirements outside the ring are unchanged. In particular, the reassignment creates no new rings since system states in which all the pups defining the broken ring have yet to enter the ring, or all the pups have left the ring, are unchanged, and each such pup can now advance singly through the remnant of its original ring subpath. □

![Diagram of pup movement](image)

Figure 2.6: Path reassignment breaks the ring of Figure 2.5.

We show next that rings fully account for the discrepancies between the NLP formulations of PM and SCPM and the original combinatorial problems stated in Section 2.2.2, in the sense that if we can construct a ring free routing from an NLP solution,
we can also construct a complete pup dispatching sequence that demonstrates feasibility to the combinatorial problem. Theorem 2 constructs such a dispatching sequence by effectively advancing tokens.

**Theorem 2** If some routing of a solution to the NLP formulation of either Pup Matching or Single Commodity Pup Matching contains no waiting ring, the solution is feasible to the corresponding combinatorial problem.

**Proof.** Consider the following dispatching procedure, given a ring free routing to the NLP formulation. (Recall that a routing includes both paths and pairs.)

1. For each pup, initialize a label for its current node to its origin.

2. Choose a pup whose current node and destination node are not equal, and determine whether the pup can advance. If the pup is assigned to travel singly to its next node, it can advance. If it is to travel with another pup, and the current nodes of the two pups are equal, both can advance. Otherwise, the first pup cannot advance.

3. If the pup chosen can advance, update its current node and that of its pair, if any, to the next node on the assigned path.

4. If the pup chosen in step 2 cannot advance, the match for its next arc is currently at another node. Determine whether that pup can advance. If so, proceed as in step 3. If not, determine whether the match of the latter pup can advance, and so forth, until a pup that can advance is chosen, and go to step 3.

5. If each pup has reached its destination, terminate. Otherwise, return to step 2.

Step 3 eventually finds a pup that can advance by the assumption of a ring free routing. The process eventually moves each pup to its destination since a finite number of arcs comprise the path of each pup. Termination proves the result. □

This dispatching result does not indicate how to construct a routing. A routing specifies a path for each pup while the flow variables of an NLP solution might trace a cyclic walk. We can remove any such cycles without introducing infeasibility, since we can convert any dispatching sequence corresponding to the solution before cycle removal to a routing of the acyclic solution by sending singly any former pairs of a pup within a cycle. The latter routing would not require additional cab loadings,
and, with no new pup pairings, cannot create a new waiting ring. Given an acyclic NLP solution, we can construct a routing by assigning matchings for each arc with a flow of more than one pup. However, Theorem 3 below shows that the problem of determining whether some ring free routing corresponds to such an acyclic NLP solution is \( \mathcal{NP} \)-complete.

Lemma 1 and Theorem 2 together imply that the NLP formulation leads to an optimal solution to the Single Commodity Pup Matching Problem. In turn, equivalence of PM and SCPM in the single origin case implies correctness of the NLP formulation for that special case of PM.

**Corollary 2** The NLP formulation of Single Commodity Pup Matching determines the optimal loading cost.

**Corollary 3** The NLP formulation of Pup Matching determines the optimal loading cost if all pups share a single origin.

Since Pup Matching remains \( \mathcal{NP} \)-complete in the single origin case, exactness of the NLP formulation for this special case of PM, as well as for the general SCPM, implies that the NLP formulations are also \( \mathcal{NP} \)-complete.

**Corollary 4** The NLP formulation of either Pup Matching or Single Commodity Pup Matching, posed as the decision problem of whether there exists a feasible solution with cost not exceeding a specified value is \( \mathcal{NP} \)-complete.

Finally, we show that the decision problem of whether a ring free routing corresponds to a given feasible solution to an NLP formulation of Pup Matching is \( \mathcal{NP} \)-complete. We refer to the problem as the Waiting Ring Problem and state it as:

**Waiting Ring Problem (WR)**

**Instance:** A directed network \( G = (N, A) \), a set of \( K \) (acyclic) paths on \( G \), and an integral capacity loading on each arc in \( A \) satisfying the
property that the number of paths traversing each arc is no more than twice the loading on that arc.

**Problem**: If each unit of loading can be used once to advance one or two tokens along its assigned arc, determine whether there exists a utilization sequence of the loadings that advances one token from the head node to the tail node of each of the $K$ paths.

**Theorem 3** The Waiting Ring Problem is $NP$-complete.

We give a proof via transformation of Satisfiability in Appendix A.2. The following corollary of WR complexity is a special case of the negative result of Karp and Papadimitriou [30], and indicates that we cannot reasonably expect to determine a set of inequalities, even nonlinear inequalities, that guarantees a solution with a ring free routing to the NLP formulation of Pup Matching.

**Corollary 5** A set of constraints that eliminates waiting rings and that we could search in nondeterministic polynomial time would imply that $NP = co-NP$.

**Proof**. Such constraints would permit solution of the complement to the Waiting Ring Problem (i.e., the problem of whether every routing contains a ring) in nondeterministic polynomial time by checking all such constraints. Consequently, such constraints would imply that the Waiting Ring Problem is in $co-NP$ as well as $NP$, which, in turn, would imply that $NP = co-NP$. See Karp and Papadimitriou [30].

The polyhedral results and computational studies of sections 6 and 7, respectively, consider the NLP formulations of PM and SCPM. Corollary 5 and our observation of few waiting rings on initial Pup Matching test instances seem to justify focus on the incomplete formulation.

### 2.4 Initial Computations

By illustrating the computational difficulty of the NLP formulation of Pup Matching, this brief section motivates the tailoring of the branch and bound procedure summa-
rized in Sections 2.5 and 2.6. We first applied the default CPLEX branch and bound routine to a series of fabricated problems including several defined on the grid-like graph shown in Figure 2.7. The graph represents a set of city blocks, and each edge in the figure corresponds to two arcs, one in each direction.

![Graph](image)

Figure 2.7: The underlying graph for several Pup Matching test problems.

**Example 2** Deliver a pup from the origin node indicated in the lower left corner of Figure 2.7 to each of the other 55 nodes. Each arc cost is 1.

The objective equals the number of cab loadings needed to complete the deliveries. Given this problem, we might quickly find a solution of cost 196 as shown in Figure 2.8: the horizontal flow occurs only on the lower most lateral street, and the numbers indicate cab loadings. Although 196 is the optimal solution, the unmodified branch and bound code was able to improve its lower bound from the LP relaxation value of 182 to only 184 with several days’ computation time.

### 2.5 Heuristic Solution Approaches

Pup Matching poses the trade-off between directly routing pups and efficiently utilizing loaded capacity. This section outlines and analyzes several heuristic approaches
Figure 2.8: Solution of cost 196 to the problem of delivering 1 pup from the origin node to each of the other 55 nodes. The numbers indicate cab loadings.

to address this trade-off. Section 2.5.1 develops an exact algorithm for matching only 2 pups that we apply in Section 2.5.2 to develop matching based approximations. This approach yields the optimal solution to Pup Matching subject to the additional constraint that we can pair each pup with at most one other pup. Section 2.5.3 outlines several heuristics based on shortest path calculations. These procedures dynamically modify arc lengths to steer pups toward unused capacity and, hopefully, achieve efficient capacity utilization.

2.5.1 Matching 2 Pups

Since each cab can tow two pups, matching two pups reduces to a connectivity problem, that is, its optimal solution is the cheapest subgraph with a directed path connecting each origin-destination pair. We next observe that in some optimal solution these two paths merge along only one subpath.

**Lemma 2** In some optimal solution to the two Pup Matching Problem, the two O-D paths coincide only along some (possibly empty) directed subpath. That is, some optimal solution has nonoverlapping $O_1 - D_1, O_2 - D_2$ paths or the general structure shown in Figure 2.9, where the arrows represent disjoint paths.


Figure 2.9: General structure of an optimal solution to two Pup Matching. Each arrow represents a path.

**Proof.** The result follows from the observation that if the two pup paths contain two distinct directed paths \( P \) and \( Q \), between two nodes \( p \) and \( q \) of the network, then routing both pups on the cheaper of \( P \) and \( Q \) costs no more than routing one pup on each path. \( \square \)

In turn, the following algorithm yields an optimal solution to the two Pup Matching Problem. We can solve the two pup SCPM by applying Algorithm 1 once for each of the two possible origin-destination pairings.

**Algorithm 1.**

1. Run an all pairs shortest path algorithm on the network. Let \( d(i, j) \) be the shortest distance between nodes \( i \) and \( j \).

2. Let \( O_1, O_2, D_1, D_2 \) be the origin and destination nodes. For each pair of nodes \( p \) and \( q \), calculate \( l_{p,q} = d(O_1, p) + d(O_2, p) + d(p, q) + d(q, D_1) + d(q, D_2) \).

3. The optimal solution corresponds to \( \min_{p,q} \{ l_{p,q}, d(O_1, D_1) + d(O_2, D_2) \} \).

**Proof of Correctness.** Given initial and final junction nodes \( p \) and \( q \), each optimal subpath \( O_1 - p, O_2 - p, p - q, q - D_1 \), and \( q - D_2 \) is, by contradiction, a corresponding shortest path. So, \( l_{p,q} \) is the optimal solution value given that the pups travel together only from \( p \) to \( q \). Now, either the pups travel together or they do not, and, if they do not, it is optimal to send each on its shortest path. Step (3) considers both possibilities. \( \square \)
The two Pup Matching Problem is a directed variation of the Generalized Steiner Network Problem, that of finding the cheapest forest linking each of a given set of node pairs. (See, for example, Goemans and Williamson [23].) If the two pups share a common origin or destination, Pup Matching reduces to the Directed Steiner Network Problem, that of finding a minimum cost subgraph containing a directed path from a specified source node to each node in a specified subset of nodes. (See, for example, Winter [47].)

2.5.2 Matching Approximations

In Section 2.2.2 we noted that direct application of a weighted matching algorithm does not solve the Pup Matching Problem because each pup may be matched to more than one other pup. Matching costs would be well defined and independent, however, if each pup could be paired with at most one other pup along its entire origin-destination path. Specifically, if a solution pairs pups A and B under such a restriction, their optimal routing, independent of the routing of the other pups, is given by solving the 2 pup problem defined by the same network and only pups A and B. In this setting, we would be able to solve pup matching by applying a weighted matching algorithm to a graph with a node corresponding to each pup and edge weights given by optimal 2 pup matching costs. The following algorithm formalizes the approach.

Matching Approximation.

1. For every pair of pups, solve the 2 pup problem with Algorithm 1 of Section 2.5.1.
2. Using the results of step (1), populate a cost matrix with a row and column for each pup and a matrix element for each possible pair of pups. If the number of pups is odd, add a dummy pup with matching costs equal to the shortest origin-destination path lengths of the pups.

3. Solve the weighted nonbipartite perfect matching for the cost matrix calculated in step (2).

**Corollary 6** Pup Matching under the additional constraint that each pup may be paired with at most one other pup is polynomially solvable.

A pup matched with the dummy is paired with no real pup and travels along its shortest path. Each other pup is routed according to the solution of the 2 pup problem that determines its matching cost. Some such pups might be matched in name only and actually be assigned to travel singly (as specified on the solution to the 2 pup matching problem).

For single origin pup matching, the heuristic of feasibly loading the optimal directed Steiner tree that connects the common origin to each of the destination nodes might seem an attractive extension of the optimality result for 2 pups. Example 3 and Figure 2.10 show that the Steiner tree does not define the optimal solution for single origin problems with more than two pups. Furthermore, Lemma 3 implies that the matching heuristic provides a lower bound on the best tree solution to the single origin problem.

**Example 3** In the network of Figure 2.10, route pups A, B, C, and D from node 1 to nodes 2, 3, and 4 as indicated. The optimal solution loads a single cab on each arc for a cost of 6. A tree solution must load 2 cabs on either arc 1-2 or arc 1-3 and costs at least 7.

**Lemma 3** If the pup paths of the single origin problem form a tree, then some routing pairs each pup with no more than one other pup.
Figure 2.10: The optimal directed Steiner tree connecting destinations 2, 3, and 4 to origin 1 does not yield the optimal Pup Matching solution.

**Proof.** Given the tree forming paths, we can construct such a pairing by first pairing the two pups whose paths from the source node share the greatest distance (or cost) along each of their common arcs, then pairing the two remaining pups whose paths share the greatest distance, and so forth. At each iteration of the assignment, resolve ties in common arc distance by choosing the tied pair with the greatest number of arcs in common and then, if a tie remains, choosing arbitrarily. Suppose this routing strategy assigns pups A and B to traverse an arc singly. Since there is only one path from the source node to each node of the tree, the common path of pup A and its pair, if any, and the common path of pup B and its pair, if any, is no greater than that pups of A and B. So, the routing procedure would pair A and B. □

**Corollary 7** The matching heuristic provides a lower bound on the best tree solution to the single origin problem.

Finally, we will prove that the Matching Approximation (MA) has an absolute performance ratio of 2. If $A$ is an approximation algorithm for a minimization problem $\pi$, $A(I)$ is the solution value returned by $A$ on instance $I \in \pi$, and $OPT(I)$ is the optimal value for instance $I$, then the performance ratio of $A$ on the instance $I$ is defined as $R_A(I) = \frac{A(I)}{OPT(I)}$. The absolute performance ratio of $A$ is defined as $r_A = \inf (r | R_A(I) \leq r, \forall I \in \pi)$.

**Theorem 4** The absolute performance ratio of the Matching Approximation, $r_{MA}$, is 2 for both Pup Matching and its NLP formulation.

**Proof.** The ratio is no greater than 2 because the Matching Approximation can do no worse than routing each pup singly on its shortest origin-destination path.
Specifically, if $m_i$ is the cost of the $i$th match of the approximation, $d_j$ is the length of the shortest origin-destination path of pupil $j$, and $d'_j$ is the origin-destination path length of pupil $j$ in an optimal solution, then for any instance $I$ of Pup Matching, $MA(I) = \sum_i m_i \leq \sum_{j<k} d_j \leq \sum_{j<k} d'_j \leq 2OPT(I)$.

On the other hand, Figure 2.11 depicts a sequence of instances with a limiting performance ratio of 2. Instance $n$ consists of $n+1$ nodes and $n+1$ arcs connected as in the figure, as well as $n$ pupils. For $i = 1, 2, \ldots (n-1)$, nodes $i$ and $i+1$ form an origin-destination pair for one pupil, and nodes 1 and $n$ are the origin-destination pair for the final pupil. The optimal solution routes each pupil along its unique O-D path among nodes $1, 2, \ldots n$, pairs the final pupil with each of the other $n-1$ pupils over a single arc, and achieves a cost of $n-1$ since it uses only one cab on each arc. The pairing restriction, however, blinds the Matching Approximation to the efficiency of sending the last pupil on path $1-2-\ldots- n$. Specifically, it sends the final pupil via node $A$, routes all pupils singly, and loads a single cab on each arc for a total cost of $2n-4$.

![Figure 2.11: Instance $n$ of a sequence with an infimum Pup Matching performance ratio of 2.](image)

A priori pairing of the origins and destinations of an SCPM instance defines a PM instance. Solving an assignment problem to pair the origins and destinations, and then applying the Matching Approximation to the resulting PM instance yields a 2-approximation that we refer to as the Double Matching Approximation.

**Double Matching Approximation (DMA).**

1. Perform an all pairs shortest path algorithm on the network. Replace each arc $(i, j)$ with $d(i, j)$, the length of the shortest $i-j$ path, and set $d(i, i) = 0$ for all $i$.

2. Using the resulting distance matrix, determine the least cost matching of origin nodes to destination nodes.

3. Apply the Matching Approximation to the Pup Matching problem defined by the origin-destination assignments in (2).
Theorem 5 \textit{The absolute performance ratio of the Double Matching Approximation, }\( r_{DMA} \text{, is 2.} \)

Proof. The absolute performance ratio is no worse than 2 because the Double Matching Approximation (DMA) can do no worse than routing the pups singly over the paths of minimum total distance with one path joining each origin node to each destination node. That is, if \( l_j \) is the length of the shortest path between the \( j \)th origin-destination pair assigned in step (2), and \( l'_k \) is the length of the path assigned pup \( k \) in an optimal routing, then for any instance \( I \) of SCPM, the cost \( DMA(I) \) of the solution to the double matching heuristic satisfies the inequalities \( DMA(I) \leq \sum_j l_j \leq \sum_k l'_k \leq 2OPT(I) \).

On the other hand, Figure 2.12 depicts a family of instances with a limiting performance ratio of 2. The designations below the nodes specify origin and destination locations, and the numbers above the arcs are the arc lengths. The Double Matching Heuristic first assigns origin-destination pairs \((0, n), (0, 1), (1a, 2), (2a, 3) \ldots ((n-1)a, n)\). To see the optimality of this assignment, consider the subgraph defined by removing path \((0, A, n)\). Induction implies that all assignments on this subgraph have the same cost, \(2n - (n - 1)\epsilon\), since we can convert any assignment for subgraph \( n \) to an assignment for subgraph \( n - 1 \) with cost decrease of \(2 - \epsilon\). (Specifically, consider an arbitrary assignment on subgraph \( n \). This assignment must pair the origin at node \((n-1)a\) to a destination at node \( n \) since only node \( n \) lies to the right of node \((n-1)a\). Subgraph \( n - 1 \) has two destinations at node \( n - 1 \). By omitting from the assumed assignment O-D pair \((n-1)a, n)\) and shifting its other node \( n \) destination to node \( n - 1 \), we can define an assignment on subgraph \( n - 1 \). This new assignment \( n - 1 \) costs \(2 - \epsilon\) less than the initial assignment on subgraph \( n \), and, by assumption, all assignments on subgraph \( n - 1 \) have the same cost.) Creating O-D pair \((0, n)\) using path \((0, A, n)\) reduces the cost and, in turn, leaves only the prior assignment. Given this assignment, the proof follows as in the proof of Theorem 4. \(\square\)

![Figure 2.12: Family of Single Commodity Pup Matching instances with an infimum performance ratio of 2.](image-url)
2.5.3 Shortest Path Heuristics

Pup Matching heuristics based on successive shortest path calculations might seem more intuitive than the matching approximations. Perhaps the simplest strategy routes each pup on its shortest path with arc lengths given by facility loading costs. This strategy is equivalent to solving the LP relaxation and rounding up the number of cabs loaded on each arc. Alternatively, one might choose a pup, route it on its shortest path, modify the arc costs to reflect marginal costs, and repeat the procedure until we have routed all of the pups. If the first pup were routed on some arc, the marginal cost of that arc to the second pup would be 0. A third option combines the preceding two by routing the first several pups according to loading costs and the rest according to marginal costs. The delayed reduction of marginal costs might allow subsequent pups to better exploit unused capacity.

Clearly none of these procedures is optimal. Furthermore, the three heuristics do not necessarily output feasible solutions to the Pup Matching problem since each might produce a waiting ring. Each would generate the 3 node ring of Example 1. However, the heuristics generate feasible solutions to the NLP formulation of Pup Matching and yield 2-approximations for that formulation.

**Proposition 1** Each of the three successive shortest path heuristics provides a 2-approximation for the NLP formulation of the Pup Matching Problem.

**Proof.** The cost of each pup route never exceeds that of its shortest path, and half the sum of the shortest paths is a lower bound on the optimal solution. So, the heuristics have an absolute performance ratio no worse than 2.

On the other hand, the performance ratio of each heuristic is \( \frac{2}{1+\epsilon} \) on the family of instances (parameterized by \( \epsilon \)) illustrated in Figure 2.13. Pup A is to be routed from node 1 to node 6, and pup B is to be routed from node 4 to node 6. Each heuristic routes pup A on path 1-2-6 and pup B on path 4-5-6, independent of the routing
order, for a cost of 4. The optimal solution routes both pups to node 3 then sends them together to node 6 for a cost of $2 + 2\varepsilon$. □

![Diagram](image)

Figure 2.13: Pup A is to be routed from 1 to 6 and pup B from 4 to 6. The performance ratio of each successive shortest path heuristic is $\frac{2}{1+\varepsilon}$.

Epstein [19] described two shortest path based heuristics for the two facility NLP on an undirected graph that he refers to as Edge Rounding and Path Rounding. The former is equivalent to our first shortest path heuristic of routing on shortest paths and rounding up. Epstein outlined instances illustrating that Edge Rounding and Path Rounding have absolute performance ratios equal to the capacity of the larger facility.

It seems reasonable to first dispatch pups sharing the same origin and destination along their common shortest path, and then consider the matching problem over the remaining pups. As the following example shows, however, this preprocessing procedure can lead to a suboptimal solution, even if we solve the latter pup matching problem exactly.

**Example 4** Four pups, A, B, C, and D are to be routed on a network with topology and costs as shown in Figure 2.14. A and B are to travel from node 1 to node 4, C is to travel from node 1 to node 3, and D is to travel from node 1 to node 2.

The optimal routing achieves a cost of 22 by sending A and C to node 2, and B and D to node 3. The preprocessing procedure routes A and B directly from node 1 to node 4, yielding a cost of 24.
Figure 2.14: Arc costs are shown. Routing pups A and B together from node 1 to node 4 leads to suboptimality.

The shortest path heuristics can be applied to an SCPM problem by first pairing origins and destinations through some heuristic procedure, then applying the shortest path heuristic to the resulting PM problem. Solving the assignment problem with origin-destination pairing costs given by shortest path lengths then routing each pup on its shortest path (the second heuristic), yields the same solution as solving the SCPM LP relaxation then rounding up any fractional loading variables.

2.6 Valid Inequalities

To tighten the lower bound provided by the LP relaxation of the NLP formulation, we append cuts from three families of valid inequalities – cutset inequalities, residual capacity inequalities, and a class that we refer to as odd flow inequalities.

Cutset inequalities (see Magnanti, Mirchandani, and Vachani [35], Barahona [7], and Bienstock and Günlük [14]) bound the capacity loaded across a cut to accommodate the flow that must cross the cut. For Pup Matching, these inequalities assume
the following relatively simple form:

\[
\sum_{i \in S, j \notin S} z_{ij} \geq \left\lfloor \frac{D_S}{2} \right\rfloor, \forall S \subset N. \tag{2.9}
\]

In this expression, \(D_S\) is the number of pups that must leave node set \(S\), that is, the number of pups with origin in \(S\) and destination in \(N \setminus S\). The left side of the inequality is the number of cabs loaded on the cut defined by nodes \(S\). This quantity is integral and each cab has capacity 2. Consequently, the loading must be at least the ceiling of half the net demand. Since we are unable to efficiently solve the cutset separation problem, as in Balakrishnan, Magnanti, Sokol, and Wang [6], we append the inequalities (one for inflow, one for outflow) for each cut defined by a single node and then iteratively use a Gomory-Hu tree (see, for example, [24]) to identify other promising cuts.

A residual capacity inequality (see Magnanti, Mirchandani, and Vachani [34], [35]) constrains the loading requirement on a single arc, and one is defined for every commodity subset on every arc. The inequality tightens the corresponding arc capacity inequality (2.2) for flow values near the demand of the commodity subset (in this case, the number of pups in the subset, since each pup has unit demand). For the Network Loading formulation of Pup Matching, the residual capacity inequalities on arc \((i, j)\) reduce to:

\[
z_{ij} \geq \sum_{k \in \mathcal{L}} f_{ik} - \left\lfloor \frac{|L|}{2} \right\rfloor, \tag{2.10}
\]

for an odd cardinality subset of pups \(L\). The arc capacity inequalities (2.2) imply the residual capacity constraints for even cardinality subsets.
Atamtürk and Rajan [5] have shown how to separate the residual capacity inequalities for a single arc of a Network Loading Problem with \( q \) commodities and with facilities of an arbitrary capacity in \( O(q) \) time. We can also separate the residual capacity inequalities for Pup Matching in \( q \log q \) time by directly checking the inequality for commodity subsets \( L \) of maximum flow for each possible odd cardinality, since the RHS is maximized by the commodity subset defined by the largest \( f^k_{ij} \) values.

**Lemma 4** A given fractional solution violates the residual capacity inequality for a given arc of a Pup Matching problem only if it violates the inequality for a commodity subset \( L \) of maximum flow for some odd cardinality \( |L| \).

Our solution procedure checks the inequalities identified by the \( q \log q \) routine. Given that cutting plane addition typically accounted for a small fraction of overall solution time on larger instances, implementing the routine of Atamtürk and Rajan seems unnecessary.

Although the cutset and residual capacity inequalities improve the lower bounds, they do not lead to efficient solutions of all the city blocks test problems, including Example 2. In trying to prove by other means optimality of the Example 2 solution of value 196 diagrammed in Figure 2.8, we discovered a set of inequalities that constrain flow on arcs incident to a node with odd demand.

If total pup flow on an arc is odd, some capacity loaded on that arc must remain unused and we could tighten its capacity constraint. In general, the flow on a given arc can be even or odd. However, if the net demand at a node is odd, then the total inflow or total outflow must be odd, and the node must be incident to at least one
unit of unused pup capacity, half a cab's worth. Odd flow inequalities exploit this observation to tighten the sum of arc capacity constraints over the set of arcs incident to a node of odd demand.

**Theorem 6** The following odd flow inequalities are valid for the NLP formulation of Pup Matching for each node \( i \in N \) with odd net demand:

\[
\sum_{a \in A_i} z_a - \frac{1}{2} \sum_{\delta \in K} \sum_{a \in A_i} f^k_a \geq \frac{1}{2}.
\]

In this expression, \( A_i \) denotes the set of arcs (incoming and outgoing) incident to node \( i \).

The odd flow inequalities are a special case of the generalized cutset inequalities that Chopra, Gilboa, and Sastry [17] introduced in the context of the single facility, single O-D pair NLP with both flow and loading costs. Atamtürk showed that the generalized cutset inequalities yield the convex hull of the NLP variation of shipping a fixed amount of demand across a single directed cut. We return to the single cut problem in Section 4.5. Our solution procedure appends odd flow inequalities corresponding to only single nodes, though the same logic applies to any subset of nodes with odd net demand.

We next show that under strong connectivity conditions, the odd flow inequalities define facets of the convex hull of feasible solutions to the NLP formulation.

**Theorem 7** If \( G = (N, A) \) is strongly connected (contains a directed path between each pair of nodes), the total net demand of some node \( i \in N \) is odd, and node \( i \) and those nodes adjacent to it form a clique, then the corresponding odd flow inequality defines a facet of the convex hull of feasible solutions to the NLP formulation of Pup Matching.
Proof.

Validity.
The odd flow inequality forces a solution to include at least 1 unit of unused capacity, or, equivalently, half a cab's worth, on the arcs incident to node $i$. A solution can fully utilize the capacity on a set of arcs only of the flow on every such arc is even. However, the flow on at least one arc incident to node $i$ must be odd because the odd demand forces either the total inflow or the total outflow to be odd.

More formally, we can derive the odd flow inequality as a rank 1 Gomory-Chvátal cut:

Addition of the capacity constraints for the arcs $A_i$ incident to node $i$ yields:

$$2 \sum_{a \in A_i} z_a - \sum_{k \in K} \sum_{a \in A_i} f_a^k \geq 0.$$ 

Substitution via the node $i$ flow balance constraint yields:

$$2 \sum_{a \in A_i} z_a - 2 \sum_{k \in K} \sum_{a \text{ leaving } i} f_a^k \geq d_i.$$ 

Division by 2 and rounding yields:

$$\sum_{a \in A_i} z_a - \sum_{k \in K} \sum_{a \text{ leaving } i} f_a^k \geq \left\lceil \frac{d_i}{2} \right\rceil.$$ 

Subtracting half of the node $i$ flow balance yields the odd flow inequality.

Face Definition.

We first show that the odd flow inequality can hold at equality. The clique assumption permits modification of any flow satisfying the flow balance constraints to some other solution satisfying the properties that some arc $(j, i)$ carries all node $i$ inflow, and, similarly, some arc $(i, l)$ carries all node $i$ outflow. Since $d_i$ is odd, either $\sum_k f_{jl}^k$ or $\sum_k f_{il}^k$ is odd. The odd flow inequality then holds at equality if $z_a = \left\lfloor \frac{\sum_k f_{jl}^k}{2} \right\rfloor$, $\forall a \in A_i$. Also, for some feasible solution, the odd flow inequality does not hold at equality, because we can add extra loadings to any feasible solution to obtain another feasible solution.

Therefore, the odd flow inequality is a nonempty proper face of the convex hull of feasible solutions.

Facet Definition.

Let $P$ be the set of feasible solutions to the network loading problem, and let $L = \{(z, f) \in \text{conv}(P) \mid \text{the odd flow inequality holds at equality}\}$. Suppose some other inequality $\beta z + \gamma f \geq \delta$ (***) satisfies the property that $L \subseteq \{(z, f) \mid (***) \text{ holds at equality}\}$. We will show that (***) is a linear combination of the odd flow inequality and the flow balance equalities, implying that $\dim(L) = \dim(\text{conv}(P)) - 1$ and that $L$ is a facet of $\text{conv}(P)$.
(a) $\beta_a = 0$ for all arcs $a$ not incident to node $i$, because for any $(z, f) \in L$, the solution given by increasing $z_a$ by 1 is also in $L$.

(b) $\beta_a = \beta$ for some constant $\beta$ for all $a$ incident to node $i$.
Let $(z_0, f_0)$ be the feasible solution described previously, with all the flow into node $i$ via some other node $j$, all the flow from node $i$ via some node $l$, and $z_a = \left\lceil \frac{\sum_k f^k_a}{2} \right\rceil$, $\forall a \in A$. Modify $(z_0, f_0)$ by sending one additional unit of flow of some commodity $k$ around the cycle $(j, i) \rightarrow (i, l) \rightarrow (l, j)$ (If $l = j$, ignore the arc $(l, j)$, and if $l \neq j$, arc $(l, j)$ exists by the clique assumption.), and incrementing the loading on either $(j, i)$ or $(i, l)$ and $(l, j)$ if necessary, to maintain feasibility. Call the new solution $(z_1, f_1)$.

Form a third solution $(z_2, f_2)$ by modifying $(z_1, f_1)$ in the same manner. Note that if arc $(j, i)$ capacity is tight in $(z_0, f_0)$, then arc $(i, l)$ capacity is tight in $(z_1, f_1)$, and vice versa, and that $(z_0, f_0)$, $(z_1, f_1)$, and $(z_2, f_2)$ are all in $L$. Assume without loss of generality that arc $(j, i)$ is tight in $(z_0, f_0)$. Then,

$$\left(\beta z_1 + \gamma f_1\right) - \left(\beta z_0 + \gamma f_0\right) = \beta_{ji} + \gamma_{ji} + \gamma_{il} + \gamma_{lj} = 0. \quad (\gamma_{lj} \text{ is irrelevant if } l = j).$$

Similarly,

$$\left(\beta z_2 + \gamma f_2\right) - \left(\beta z_1 + \gamma f_1\right) = \beta_{il} + \gamma_{ii} + \gamma_{il} + \gamma_{lj} = 0 \Rightarrow \beta_{ji} = \beta_{il}.$$

Since we chose nodes $j$ and $l$ arbitrarily among adjacent nodes, $\beta_a = \beta$ for all arcs $a$ incident to node $i$.

(c) $\gamma^k_a = 0, \forall a$ not incident to node $i$.
Consider again node $j$, and let $T = (N, A')$ be a directed spanning tree formed by directed paths from this node to each other node. Such paths exist by the strong connectivity assumption. That they can form a directed spanning tree follows from induction on the number of nodes. Furthermore, assume that node $i$ is a leaf of $T$ connected to the tree by arc $(j, i)$. The clique assumption guarantees that we can reroute any paths through node $i$ via adjacent nodes. Since we can modify (**) by adding to it flow balance constraints so that $\gamma^k_a = 0, \forall a \in A', \forall k \in K$. We could also prove this claim using induction on the number of nodes.

To show that $\gamma^k_a = 0$ for $a \notin A'$ not incident to node $i$, first modify the initial solution $(z_0, f_0)$ by adding 1 unit of flow through $T$ for some commodity $k$ to each node $p \neq j, i$ and along each arc of a $p - j$ directed path that does not include node $i$. Strong connectivity and the clique assumption guarantee that such directed paths exist. Load $\left\lceil \frac{\sum_k f^k_a}{2} \right\rceil + 1$ facilities on all arcs $a$ not incident to $i$ and $\left\lceil \frac{\sum_k f^k_a}{2} \right\rceil$ facilities on all arcs incident to $i$. Call the resulting solution $(z_3, f_3)$, and note that $(z_3, f_3) \in L$ since the flows on arcs incident to node $i$ are the same as those of $(z_0, f_0)$. Now consider some arc $(r, q) \notin A'$ that is not incident to node $i$. Incrementing the flow of $k$ on $(r, q)$ and the $j - r$ path of $T$, and decrementing the flow on the $j - q$ path of $T$, generates a new solution in $L$. Since $\gamma^k_a = 0, \forall a \in A', \gamma^k_a = 0$. Since we chose commodity $k$ and arc $(r, q)$ arbitrarily, the result follows.

(d) $\gamma^k_{i, r} = -\beta, \forall k \in K$.  

53
Increment the flow of $f_0$ around the cycle $(l, j) \to (j, i) \to (i, l)$ (if $l = j$, ignore the first arc, which does not exist) for some commodity $k$. One additional loading will be required on either $(j, i)$ or $(i, l)$, and, perhaps, on $(l, j)$. The resulting solution $(z_4, f_4) \in L$. Comparing $(z_0, f_0)$ with $(z_4, f_4)$ yields $\gamma_{j, i}^k + \gamma_{i, l}^k + \beta = 0$
\[ \Rightarrow \gamma_{j, i}^k = -\beta, \text{ since } (j, i) \text{ is in the tree } T. \]
Now suppose the network contains some other adjacent node $l'$, and modify $(z_4, f_4)$ by sending 2 additional units of $k$ around the cycle $(l, j) \to (j, i) \to (i, l') \to (l', l)$ (again, ignore $(l, j)$ if $l = j$, and if $l' = j$, the argument still holds). Add 1 loading to each of $(l, j), (j, i), (i, l')$, and $(l', l)$ to create $(z_5, f_5) \in L$. Comparing $(z_5, f_5)$ with $(z_4, f_4)$ yields $2\gamma_{j, i}^k + 2\gamma_{i, l}^k + 2\beta = 0$
\[ \Rightarrow \gamma_{i, l'}^k = -\beta. \]
\[ \Rightarrow \gamma_{i, a}^k = -\beta, \forall a \text{ with } i \text{ as tail node, for all commodities.} \]
(e) Show that $\gamma_{j, i}^k = 0, \forall j', \forall k \in K$.
Consider some node $j' \neq j$ adjacent to node $i$. Modify $(z_0, f_0)$ by adding 2 units of flow of commodity $k$ around the cycle $(l, j') \to (j', i) \to (i, l)$ (if $l = j'$, ignore $(l, j')$, analogous to before), and by adding 1 loading to each of $(l, j'), (j', i)$ and $(i, l)$, to create $(z_6, f_6) \in L$. Comparing $(z_6, f_6)$ and $(z_0, f_0)$ yields $2\gamma_{j', i}^k + 2\gamma_{i, l}^k + 2\beta = 0$
\[ \Rightarrow \gamma_{j', i}^k = 0. \]
So, $\gamma_{j', i}^k = 0$ for all arcs with head node $i$, for all commodities $k$.

(f) Summary
$\beta_a = 0$, for all arcs $a$ not incident to node $i$,
$\beta_a = \beta$, for all arcs $a$ incident to node $i$, for some constant $\beta$,
$\gamma_{i, j}^k = 0, \forall j \in N, \neq i, \forall k \in K$,
$\gamma_{i, l}^k = -\beta, \forall l \in N, \neq i, \forall k \in K$,
$\gamma_{i, q}^k = 0, \forall (r, q) \text{ not incident to } i, \forall k \in K$.
Using the flow balance constraints on node $i$, we can convert $\gamma$ so that
$\gamma_{i, i}^k = -\frac{1}{2} \beta$,
$\gamma_{i, l}^k = -\frac{1}{2} \beta$,
$\gamma_{i, q}^k = 0$. \(\square\)

The clique assumption is not necessary. The proof still holds if we assume only a directed path not including node $i$ between any nodes $j$ and $l$ adjacent to node $i$.

The SCPM has only one commodity, and the odd flow inequality is written as
\[ \sum_{a \in A_i} z_a - \frac{1}{2} \sum_{a \in A_i} f_a \geq \frac{1}{2}. \]
Essentially the same proof as that of Theorem 7 shows that this inequality is valid and facet defining.

**Theorem 8** If $G = (N, A)$ is strongly connected, $d_i$, the total net demand of some node $i \in N$, is odd, and node $i$ and those nodes adjacent to it form a clique, then the
odd flow inequality defines a facet of the convex hull of feasible solutions to the NLP formulation of the SCPM.

2.7 Computational Results

Our solution procedure first finds a lower bound by tightening the LP relaxation of the NLP formulation by adding cutset, residual capacity, and odd flow inequalities, in that order. As mentioned earlier, we use Gomory-Hu calculations to identify interesting cutset inequalities. We exactly separate and append residual capacity inequalities until the bound improvement falls below a threshold, and we append all odd flow inequalities since they number at most the cardinality of the node set. We then obtain an upper bound and initial feasible solution by running all four heuristics and retaining the best value. Finally, we call the CPLEX branch and bound routine.

Figure 2.15 summarizes the results of our solution procedure on five city blocks problems. Problem 7i is the city blocks problem of Example 2. 7ii and 7iii are defined on the same graph. Problems 9i and 9ii are defined on a similarly sized graph of one way streets. The portion of the graph below the zero line depicts the error of the best heuristic. In all cases except 9ii, at least one heuristic found the optimal solution, and, in that case, the best value was less than 2% from optimal. The portion of the graph above the zero line summarizes the lower bound improvement from sequential application of the cutting plane families. The length of each composite box is proportional to the LP relaxation error, and each inner box indicates the bound improvement from the corresponding family of inequalities. In Problem 9ii for example, the LP relaxation error was 12.8%, the cutset inequalities reduced the error
to 8.0%, the residual capacity inequalities reduced the error about another 1%, and the odd flow inequalities increased the lower bound to the optimal solution value.

![Graph showing error reduction by problem type](image)

Figure 2.15: Results of the branch and bound procedure on city blocks problems. The portion of the graph above the zero line depicts lower bound improvement from sequential application of the three cutting plane families, and the portion below the zero line indicates the error of the best heuristic.

We also applied the branch and bound solution procedure to 30 problems randomly generated from realistic data. The results seem good but not as dramatic as those exhibited for the city blocks problems. Given a node set in (latitude, longitude) format based on a real logistics network, we defined problems by selecting a subset of nodes, calculating arc lengths as Euclidean distances, and randomly selecting origin-destination pairs. Since we used the distance metric, all the underlying graphs were complete. About half the problems had a single origin.

We limited the branch and bound tree to 220M of memory and 2 hours of CPU time. Using all three cut families, we were able to solve 67% of the problems to optimality with an average gap reduction of 18.8% to 6.4%. (Since the procedure did not solve all the problems, the gap reflects the difference between a lower bound and the tightest upper bound.) Without the odd flow cuts, we were able to solve 30% of
the problems and reduced the gap to 7.8% on average. With no cuts, we solved only 17% of the problems. Among the solved problems, the average heuristic error was 1.3%.
Chapter 3

Compartmentalized Network Loading

This chapter describes a generalization of the Network Loading Problem that we refer to as the Compartmentalized Network Loading Problem (CNLP). Section 3.1 motivates the problem, and Section 3.2 formulates three variations. Section 3.3 briefly considers the linear programming relaxation of the CNLP and also converts a compartmentalized variation of the multicommodity flow problem to the familiar multicommodity flow problem. Section 3.4 describes a cutset inequality for each formulation. Section 3.5 considers two single arc problems, the CNLP itself on a single arc and a single arc problem arising as a Lagrangian relaxation of the CNLP. Section 3.6 solves the CNLP for the special case of a single origin-destination pair, and Section 3.7 presents results for problems on graphs with only three nodes. Section 3.8 summarizes our branch and bound solution approach, and Section 3.9 presents computational results. This chapter a broad variety of results, so we conclude with a
summary in Section 3.10.

3.1 Introduction

Capacitated network flow and network design models typically assume homogeneous capacity. Every traveler who has struggled to accommodate a fragile gift despite ample space in his suitcase, though, knows that not all capacity is equally useful. Multicommodity flow problems generalize standard network flow problems to allow for inhomogeneous flow on each arc. In this chapter, we generalize network loading to allow for certain types of inhomogeneous capacity through three models that we refer to collectively as Compartamentalized Network Loading (CNLP). Specifically, the facilities available for loading are composed of compartamentalized capacity with each compartment providing a known capacity that is reserved for a specified subset of commodities. We might view each compartamentalized facility as a partitioned bin, with each subdivision compatible with only a given subset of commodities.

Compartamentalized situations might arise in several network design applications. Most commercial passenger jets consist of two or more cabins, and a passenger cannot be seated in cabin classes less exclusive than that corresponding to his ticket. We can model as a CNLP the problem of finding the cheapest set of flights that feasibly accommodates a given stratified passenger demand over an airline network. Similarly, some multicompartment tractor trailers can channel heat in different quantities to different compartments, and we can model as a CNLP the problem of finding the cheapest allocation of such trucks over a logistics network that transports chemicals
with varying temperature requirements between specified locations.

A CNLP can also model certain content constraints of homogeneous capacity. For example, we might formulate capacitated network loading as a crude model of an integrated services digital network (ISDN). An ISDN combines telecommunications streams from various sources such as voice, video, and data. Due to statistical differences among the streams, a voice source with average demand of $x$ and a data source with average demand of $z$ might both fit in a facility unable to accommodate two different voice sources of $x$ units each. We might model network design under such constraints as a CNLP with facilities of three compartments, one for voice and data, one for voice only, and one for data only.

### 3.2 Notation and Formulation

This section formulates three variations of the Compartmentalized Network Design Problem (CNLP), a general model and the special cases of segregated compartments and nested compartments. In segregated settings only one compartment of each facility can accommodate each commodity. In nested settings each commodity belongs to one of an ordered set of classes, and a commodity in a higher class can occupy any compartment that a commodity of a lower class can occupy. The general model allows for an arbitrary relationship between compartments and commodities. All three models assume an underlying undirected graph and only one facility type. In terms of the airline fleet loading application mentioned earlier, this latter assumption corresponds to a fleet of only one aircraft type.
To formulate the general CNLP as an MILP, we use the following data:

- $G = (N, A)$: an undirected graph,
- $c_{ij}$: the cost of loading one facility on edge $(i, j) \in A$,
- $C_l$: the capacity of compartment $l$,
- $K$: the set of compartments comprising each facility,
- $Q$: the set of commodities,
- $Q_l \subseteq Q$: the set of commodities that can be placed in compartment $l$,
- $K^k \subseteq K$: the subset of compartments commodity $k$ can occupy,
- $O^k, D^k$: origin and destination node, respectively, for commodity $k$,
- $u^k$: demand for commodity $k$,

and the following variables:

- $f_{ij}^{k,l}$: the (continuous) flow of commodity $k$ from node $i$ to node $j$ in compartment $l$,
- $z_{ij}$: the (integral) number of facilities loaded on edge $(i, j)$.

Using these data and variables, we formulate the model as follows.
Compartmentalized Network Loading Problem (CNLP)

\[
\min \sum_{(i,j) \in A} c_{ij}z_{ij}
\]

subject to:

\[
\sum_{k \in K^k} (f_{ij}^k - f_{ji}^k) = \begin{cases} 
  w^k & \text{if } i = O^k, \\
  -w^k & \text{if } i = D^k, \\
  0 & \text{otherwise}
\end{cases}, \forall i, \forall k \tag{3.1}
\]

\[
\sum_{k \in Q^l} (f_{ij}^k + f_{ji}^k) \leq C^l_{ij}, \forall \{i, j\} \in A, \forall l \in K \tag{3.2}
\]

\[
f_{ij}^k \geq 0, \forall (i, j) \in A, \forall k \in Q, \forall l \in K \tag{3.3}
\]

\[
z_{ij} \geq 0, \text{integer, } \forall (i, j) \in A \tag{3.4}
\]

Constraints (3.1) balance the total flow of each commodity about each node. Constraints (3.2) require sufficient capacity of each compartment type on each edge. As in the NLP, the objective is to minimize the cost of the loaded facilities.

Each commodity of a segregated problem can occupy only one compartment type. That is, for each commodity \(k\), \(K^k\) consists of only one compartment type. Consequently, the compartment index on the flow variable is redundant, and we eliminate it, to obtain the following formulation.

Segregated Compartments Network Loading Problem (SCNLP)

\[
\min \sum_{(i,j) \in A} g_{ij}z_{ij}
\]

subject to:

\[
\sum_{j} (f_{ij}^k - f_{ji}^k) = \begin{cases} 
  w^k & \text{if } i = O^k, \\
  -w^k & \text{if } i = D^k, \\
  0 & \text{otherwise}
\end{cases}, \forall i, \forall k \tag{3.5}
\]

\[
\sum_{k \in Q^l} (f_{ij}^k + f_{ji}^k) \leq C^l_{ij}, \forall \{i, j\} \in A, \forall l \in K \tag{3.6}
\]

\[
f_{ij}^k \geq 0, \forall (i, j) \in A, \forall k \in Q \tag{3.7}
\]

\[
z_{ij} \geq 0, \text{integer, } \forall (i, j) \in A \tag{3.8}
\]

Each commodity class of a nested problem is eligible to occupy compartment types
up to some specified class, and not compartments of the next lower class. For this model, we redefine \( K^k \) to be that "lowest" compartment for which class \( k \) is eligible. Commodities \( Q^l = \{1, 2, \ldots, l\} \) can occupy compartment \( l \), permitting us to eliminate the \( Q^l \) notation. Also, we can nest the capacity constraints and so again eliminate the flow variable compartment index, giving the following formulation. This model might apply to the fleet loading problem, with each facility compartment corresponding to a cabin class.

**Nested Compartments Network Loading Problem (NCNLP)**

\[
\min \sum_{(i,j) \in A} c_{ij} z_{ij}
\]

subject to:

\[
\sum_j (f_{ij}^k - f_{ji}^k) = \begin{cases} u^k & \text{if } i = O^k, \\ -u^k & \text{if } i = D^k, \\ 0 & \text{otherwise} \end{cases}, \forall i, \forall k
\]  

(3.9)

\[
\sum_{\{k: K^k \leq l\}} (f_{ij}^k + f_{ji}^k) \leq \left( \sum_{m=1}^l C^{m}\right) z_{ij}, \forall \{i, j\} \in A, \forall l \in K
\]  

(3.10)

\[
f_{ij}^k \geq 0, \forall \{i, j\} \in A, \forall k \in Q
\]  

(3.11)

\[
z_{ij} \geq 0 \text{, integer}, \forall \{i, j\} \in A.
\]  

(3.12)

\( f_{ij}^k \) is the commodity \( k \) flow from \( i \) to \( j \) in compartments \( \{1, 2, \ldots, k\} \).

### 3.3 Linear Programming

This brief section considers two linear programs related to the CNLP. Section 3.3.1 considers the LP relaxation of the CNLP, and Section 3.3.2 casts the LP defined by preloaded compartmentalized capacity as a multicommodity flow problem.
3.3.1 Linear Programming Relaxation of the CNLP

Relaxing the design variable integrality constraints in an NLP permits the capacity allocation to exactly match flow volume on each arc, that is, permits us to restate the capacity constraints of that model as equalities. Consequently, we can substitute for the design variables in terms of the flow variables and so remove the interdependencies among the commodities. Therefore, routing each commodity on a shortest path with edge costs equal to per capacity facility costs solves the LP relaxation.

This NLP result is a consequence of the fact that each \( z_{ij} \) appears in only one constraint. In the compartmentalized models, each \( z_{ij} \) appears in multiple constraints. Consequently, as Example 5 illustrates, the shortest path solution is not necessarily optimal for even the segregated compartments CNLP. Indeed, we recognize no familiar structure in the solution to the CNLP LP relaxation.

**Example 5** We must send one unit of flow from node 1 to node 2 and one unit from node 1 to node 3 in the graph of Figure 3.1, using the specified edge costs. Each facility consists of two segregated compartments, each of capacity 2, and each commodity defines a separate class. The shortest path solution loads \( \frac{1}{2} \) facilities on each of 1-2 and 1-3 for a cost of 2. An optimal LP solution loads \( \frac{3}{2} \) facilities on each of 1-2 and 2-3 for a cost of \( \frac{3}{2} \).

3.3.2 Compartmentalized Multicommodity Flow

Consider next a multicommodity flow problem with preloaded, compartmentalized capacities instead of the familiar homogeneous capacities, that is, multicommodity flow generalized much as the CNLP generalizes the NLP. This problem is perhaps a more basic compartmentalized problem (compartmentalized standard network flow makes little sense, since it involves only one commodity) than the CNLP, but we show
Figure 3.1: One unit of flow must be sent from node 1 to node 2 in one compartment, and another unit of flow must be sent from node 1 to node 3 in another compartment. Under the specified edge costs, the shortest path solution of direct routing is suboptimal for the LP relaxation.

how to convert it to a standard, albeit potentially large, standard multicommodity flow problem. We formulate the compartmentalized multicommodity flow problem as the following LP for the case of general compartments.

Compartmentalized Multicommodity Flow

\[
\min \sum_{(i,j) \in A} a_{ij}^{k} \left( \sum_{l \in K} f_{ij}^{k,l} \right)
\]

subject to:

\[
\sum_{j} \sum_{l \in K} \left( f_{ij}^{k,l} - f_{ji}^{k,l} \right) = \begin{cases} u^{k} & \text{if } i = O^{k}, \\ -u^{k} & \text{if } i = D^{k}, \\ 0 & \text{otherwise} \end{cases}, \forall i, \forall k \tag{3.13}
\]

\[
\sum_{k \in Q^{l}} \left( f_{ij}^{k,l} + f_{ji}^{k,l} \right) \leq C^{l}z_{ij}^{0}, \forall (i, j) \in A, \forall l \in K \tag{3.14}
\]

\[
f_{ij}^{k,l} \geq 0, \forall (i, j) \in A, \forall k \in Q, \forall l \in K. \tag{3.15}
\]

The formulation is the same as that of the general compartments CNLP, except that it includes flow costs, and the number of facilities loaded on each edge \((i, j)\) is fixed at value \(z_{ij}^{0}\).

Observe that, as stated, this model is not a multicommodity flow problem since the flow capacities are defined not just on each edge, but on each edge, compartment combination. We transform this problem to the familiar multicommodity flow
problem by effectively replacing each arc in the problem with a network structure compatible with the set of commodity flows that the preloaded compartments can accommodate. The structure converts compartment capacities to edge capacities.

Figure 3.2 illustrates the transformation for the directed graph version of the multicommodity compartmentalized LP for an instance with facilities of two mixed compartments A, B and three commodities 1, 2, 3. The number next to each arc is its capacity. As examples, $u^i$ is the demand of commodity 1, $c^o$ the per facility capacity of compartment A, and $z$ facilities have been preloaded on arc $(i, j)$. Commodities 1 and 2 can occupy only compartment A, and commodity 3 can occupy both compartments A and B. The arc connectivity between nodes 1, 2, 3 and A, B corresponds to these commodity-compartment relationships. Setting the commodity 1 flow cost on arc $(i, 1)$ to the commodity 1 flow cost $a^i_{1j}$, and the commodities 2 and 3 flow costs on this arc sufficiently high, ensures that only commodity 1 traverses node 1. Given this cost induced commodity separation, only commodities compatible with, for example, compartment A traverse node A. In turn, the arc $(A, j)$ capacity enforces the compartment A capacity. Setting flow costs to 0 on commodity-compartment arcs such as $(1, A)$ and capacity arcs such as $(A, j)$ yields the desired cost for the total flow on the original arc $(i, j)$.

We can transform compartmentalized multicommodity flow on an undirected graph to a standard multicommodity flow on a directed graph by combining the structure of Figure 3.2 with the undirected to directed multicommodity flow transformation described in Ahuja, Magnanti, and Orlin [2] and illustrated in Figure 3.3. In Figure 3.3, $b_{ij}$ represents unit flow cost and $u_{ij}$ the upper bound on flow. Replacing
arc \((i', j')\) with the max flow structure of Figure 3.2 yields a transformation of the undirected compartmentalized multicommodity flow problem.

![Diagram](image)

Figure 3.2: Transformation of the compartmentalized multicommodity flow problem to the standard multicommodity flow problem by replacing each arc with a network structure.

![Diagram](image)

Figure 3.3: Transformation of an undirected multicommodity flow problem to the standard directed version, as depicted in Ahuja, Magnanti, and Orlin [2].

### 3.4 Cutset Inequalities

Cutset inequalities ensure that capacity loaded across a cut is sufficient to accommodate the flow that must cross the cut. For a single facility NLP with facility capacity \(C\), the inequality for any cutset \(S\) with demand \(D^S\) that must cross the cut \(\delta(S)\) is

\[ \sum_{e \in \delta(S)} z_e \geq \left\lceil \frac{D^S}{C} \right\rceil. \]

At least \(\left\lceil \frac{D^S}{C} \right\rceil\) facilities are required to accommodate the demand volume \(D^S\).

In Section 3.3 we employed a network structure to ensure sufficient capacity on a single edge. We now use the same structure to determine the minimum number of
facilities that can accommodate the demand that must cross a cut and so formulate a cutset inequality for each CNLP model. We state facet defining conditions for the inequalities and then apply the inequalities to the CNLP defined on a single edge.

3.4.1 Formulation of Cutset Inequalities

To determine the minimum loading on any cut $\delta(S)$, we effectively aggregate the cut into a single arc, and then expand that arc to the same network structure as before, to convert compartment capacities to arc capacities. We write the cutset inequality for the general CNLP as $\sum_{e \in \delta(S)} z_e \geq z_0$, with $z_0$ defined as the smallest integer number of facilities that might accommodate the total demand that must cross the cut. Figure 3.4 essentially replicates the structure of Figure 3.2 that we applied to the compartmentalized LP. It also corresponds to an instance with two compartments $A, B$ and three commodities $1, 2, 3$ that must cross the cut, that is, with source and destination nodes on opposite sides of the cut. Commodity demands are $u^1, u^2,$ and $u^3$, the capacities of compartments $A$ and $B$ in a single facility are $c^a$ and $c^b$, and $z$ is a candidate number of facilities to load across the cut. As in the instance of Figure 3.4, we can place commodities 1 and 2 in only compartment $A$ and commodity 3 in either compartment. The connectivity between nodes 1, 2, 3 and $A, B$ reflects these restrictions.

The minimum number of facilities $z_0$ is the smallest integer $z$ permitting a flow of $u^1 + u^2 + u^3$ from node $O$ to node $D$. The arc capacities ensure that any such flow ships, for example, exactly $u^1$ units via node 1. We can find $z_0$ using binary
Figure 3.4: Max flow feasibility problem to determine the right hand side of a CNLP cutset inequality.

search. Alternatively, we observe that the total flow constrains aggregate capacity so that $(c^a + c^b)z_0 \geq u^1 + u^2 + u^3$, and analogous conditions apply for all subsets of the commodities that must cross the cut. The max flow-min cut result permits us to combine these criteria into an analytical expression for $z_0$.

**Proposition 2** Let $Q'$ denote the set of commodities $k$ with $O_k \in S$ or $D_k \in S$ but not both. Then the cutset inequality defined by nodes $S$ is

$$
\sum_{e \in \delta(S)} z_e \geq \left[ \frac{\sum_{k \in P} u^k}{\sum_{l \in U_{k \in P} K} C^l} \right], \forall P \subseteq Q'.
$$

(3.16)

**Proof.** Max flow-min cut yields the condition $\sum_{k \in P} u^k \leq \sum_{l \in U_{k \in P} K} C^l z, \forall P \subseteq Q'$, that sufficient capacity be available for every subset of demands that must cross the cut. \(\square\)

For the Nested Compartments NLP, the analytical expression of the cutset inequality simplifies with the RHS as a max with a component for each compartment.

**Proposition 3** Let $Q'^m$ denote the subset of commodities eligible for compartments $1, 2, \ldots m$ that must cross the cut. Then for the NCNLP, the cutset inequality (3.16) simplifies to:

$$
\sum_{e \in \delta(S)} z_e \geq \left[ \frac{\sum_{k \in U_{m=1}^{K} Q'^m u^k}}{\sum_{m=1}^{K} C^m} \right], \text{ for } l = 1, 2, \ldots |K|.
$$

**Proof.** The RHS simplification implies that the only subsets $P$ of commodities crossing the cut that we need to consider are $\{Q'^1, Q'^m \cup Q'^{1-m} \cup \ldots \cup Q'^{K-m} \}$. Suppose the subset $P$ contains some, but not all, commodities in some $Q'^z$, and let
$P' = P \cup Q'^d$. Since $\sum_{k \in P} u^k > \sum_{k \in P} u^k$ and $\sum_{k \in \cup_{k \in P} K^k} C^l = \sum_{k \in \cup_{k \in P} K^k} C^l$, the inequality for $P'$ implies that for $P$. So, assume that if $P$ includes one commodity of a class, it includes all commodities of that class that must cross the cut.

Suppose next that $P$ includes $Q'^j$ for some $l < |K|$, but not some $Q'^j$ for $l < j \leq |K|$ (i.e., there is a ‘gap’ in the classes represented in $P$). Then the inequality for $P' = P \cup Q'^j$ dominates that for $P$ since $\sum_{k \in P} u^k \geq \sum_{k \in P} u^k$ (the inequality need not be strict because $Q'^j$ might be empty) and $\sum_{l \in \cup_{k \in P} K^k} C^l = \sum_{l \in \cup_{k \in P} K^k} C^l$ because a commodity in class $l$ can occupy an compartment a commodity in class $j$ can occupy.

For the Segregated Compartments model, each commodity can be placed in only one compartment, and the loading across a cut must be at least the minimum requirement of each commodity class.

**Proposition 4** If $D^{sl}$ denotes the compartment $l$ demand that must cross $\delta(S)$, then for the SCNLP, the RHS of cutset inequality 3.16 simplifies to $\sum_{e \in \delta(S)} z_e \geq \max_l \left[ \frac{D^{sl}}{C^l} \right].$

We might also obtain Proposition 4 by simplifying expression 3.16 in a manner similar to that described in the proof of Proposition 3.

Magnanti, Mirchandani, and Vachani [35] developed cutset inequalities for the two facility Network Loading Problem on undirected graphs. They showed that the inequality defined by nodes $S$ is facet defining if the demand that must cross the cut $\delta(S)$ is strictly positive and not a multiple of the higher facility capacity, and the subgraphs defined by $S$ and $N \setminus S$ are connected. A slight modification of their proof applies to each variation of the compartmentalized NLP.

**Theorem 9** The cutset inequality for the CNLP, $\sum_{e \in \delta(S)} z_e \geq [z^*]$, defines a facet of the convex hull of feasible CNLP solutions if $z^* \notin Z^+$ and the subgraphs defined by $S$ and $N \setminus S$ are connected. For the general CNLP, $z^* = \max_{Q' \subseteq Q} \left( \sum_{k \in \cup_{l \leq |K|} Q'^l \cdot u^k \right) / \sum_{l \leq |K|} C^l$; for the NCNLP, $z^* = \max_{1 \leq l \leq |K|} \left[ \frac{\sum_{k \in \cup_{l \leq |K|} Q'^l \cdot u^k}{\sum_{l \leq |K|} C^l} \right]$; and for the SCNLP, $z^* = \max_{1 \leq l \leq |K|} \left( \frac{D^{sl}}{C^l} \right)$.

70
3.4.2 Single Edge CNLP

In this final subsection, we note that cutset inequalities yield almost full understanding of the CNLP on a single edge. In this special case, the sole discrete variable is \( z \), the number of facilities loaded on the one edge. The formulation becomes:

\[
\begin{align*}
\text{min } c z \\
\text{subject to:}
\sum_{l \in K^k} f_{l}^{k,l} &= u^k, \forall k \in Q \\
\sum_{k \in Q^l} f_{k}^{l} &\leq C^l z, \forall l \in K \\
f_{k}^{l} &\geq 0, \forall k \in Q, \forall l \in K \\
z &\geq 0, \text{ integer.}
\end{align*}
\]

Assume the scalar \( c > 0 \). The problem is unbounded if \( c < 0 \), and, if \( c = 0 \), any sufficiently large value of \( z \) is feasible with cost 0.

This problem involves a cut of only one edge. In turn, a solution with \( z \) facilities is feasible if and only if \( z \geq \left\lceil \frac{\sum_{k \in P} u^k}{\sum_{l \in P \cap K^k} C^l} \right\rceil, \forall P \subseteq Q \). The smallest feasible \( z \) is optimal, and binary search on a max flow problem can solve this single edge problem.

Furthermore, the cutset inequalities define the convex hull projected to \( z \), and their addition to the full formulation guarantees a solution with an integral value of \( z \).

**Theorem 10** The convex hull of feasible loadings \( z \) for the single arc NLP is

\[
\left\{ z \mid z \geq \left\lceil \frac{\sum_{k \in P} u^k}{\sum_{l \in P \cap K^k} C^l} \right\rceil, \forall P \subseteq Q \right\}.
\]

**Proof.** The unrounded cut conditions are necessary and sufficient to guarantee existence of a feasible flow. Rounding up the cut conditions eliminates no feasible \( z \) value and defines an integral polyhedron (ray). \( \Box \).

**Corollary 8** A solution with an integral \( z \) value is optimal to the following system.
\[ \text{min } cz \]
\[ \text{subject to:} \]
\[ \sum_{k \in K} f^{k,l} = u^k, \forall k \in Q \]  \hspace{1cm} (3.21)
\[ \sum_{k \in Q} f^{k,l} \leq C^l, \forall l \in K \]  \hspace{1cm} (3.22)
\[ z \geq \left[ \frac{\sum_{k \in P} u^k}{\sum_{l \in K} C^l} \right], \forall P \subseteq Q \]  \hspace{1cm} (3.23)
\[ f^{k,l} \geq 0, \forall k \in Q, \forall l \in K \]  \hspace{1cm} (3.24)
\[ z \geq 0. \]  \hspace{1cm} (3.25)

**Proof.** The problem is unbounded if \( c < 0 \). If \( c \geq 0 \), the smallest feasible \( z \) is optimal. A fractional value of this smallest \( z \) contradicts sufficiency of the cutset conditions for feasible flow. \( \Box \)

The system of Corollary 8 does not define the convex hull of the CNLP on a single edge. Example 6 illustrates that if the flow variables have general objective coefficients, the smallest \( z \) value might be suboptimal.

**Example 6** 5 units of a single commodity must cross an arc. A facility consists of two compartments, each with 2 units of capacity, and the commodity is permitted in either compartment. If the facility cost is 10, the first compartment flow cost is 0, and the other compartment cost is 10 per unit flow, the unique optimal solution to the formulation of Corollary 8 (generalized to accommodate flow costs) loads \( \frac{5}{2} \) facilities and sends all flow in the first compartment.

### 3.5 The CNLP Single Arc Design Problem and Residual Capacity Inequalities

In Chapter 2 we mentioned the residual capacity inequalities as introduced by Magnanti, Mirchandani, and Vachani [34], and described their structure for the special case of Pup Matching. In this section we extend to the CNLP some of the polyhedral results for residual capacity inequalities for the NLP. We first review these results.
Magnanti, Mirchandani, and Vachani relax the commodity flow balance constraints of the NLP to generate the following Lagrangian subproblem that they refer to as the Single Arc Design Problem (SADP).

\[
\text{SADP} \\
\min cz - \sum_{k \in Q} a^k f^k \\
\text{subject to:}
\]

\[
f^k \leq u^k, \forall k \in Q \quad (3.26)
\]

\[
\sum_{k \in Q} f^k \leq Cz \quad (3.27)
\]

\[
f^k \geq 0, \forall k \in Q \quad (3.28)
\]

\[
z \geq 0, \text{integer.} \quad (3.29)
\]

The objective rewards commodity flow across the single arc, but penalizes loaded capacity. The upper bounds on flow tighten the Lagrangian bound. We can assume \( c \geq 0 \) and \( a^k \geq 0 \). The problem is unbounded for \( c < 0 \), and \( f^k = 0 \) in any optimal solution for a commodity with \( a^k < 0 \).

Magnanti, Mirchandani, and Vachani described as residual capacity inequalities a family of cuts with a member for each commodity subset. The residual capacity inequality for \( P \subseteq Q \) is

\[
\sum_{k \in P} f^k - r^P z \leq (\mu^P - 1)(C - r^P). \quad (3.30)
\]

In this expression, \( \mu^P = \sum_{k \in P} u^k \) and, \( r^P = \begin{cases} C, & \text{if } \mu^P \mod C = 0, \\ \mu^P \mod C, & \text{otherwise.} \end{cases} \)

They proved that the addition of this exponential cut family to the SADP formulation yields the convex hull of the MILP.

Let \( z^* = \frac{1}{C} \sum_{k \in \mathbb{N}} a^k \) be the (continuous) number of facilities needed to accommodate all commodities \( k \) for which \( \frac{a^k}{z^*} \geq \frac{C}{z^*} \), that is, those commodities with reward
that covers capacity cost. Atamtürk and Rajan [5] showed that the optimal solution
to the SADP loads either $\lfloor z^* \rfloor$ or $\lceil z^* \rceil$ facilities. For a fixed loading $z$, it is optimal
to place in descending unit contribution $a^k$, as many commodities $k$ as possible, and
we can solve the SADP by comparing the objective values for $\lfloor z^* \rfloor$ and $\lceil z^* \rceil$ facilities.
Sorting the $a^k$ is the bottleneck operation for this procedure, so the routine runs in
$O(|Q| \log |Q|)$ time.

Atamtürk and Rajan also solved the SADP separation problem for the residual
capacity inequalities in $O(|Q|)$ time. We note that their separation results, and, in
turn, our subsequent results, rely on the flow upper bounds $f^k \leq u^k$. The addition
of these bounds to the CNLP does not change the value of the optimal solution, but
would invalidate the cutset inequality facet proofs referred to in Section 3.4, since the
proofs assume we may cycle commodity flow back and forth on any edge to create a
feasible solution from some other feasible solution.

This section generalizes the SADP and the residual capacity inequalities for the
CNLP. We extend the SADP convex hull result of [34] to the segregated model, and
show that for each CNLP model, we can efficiently separate an extension of the
residual capacity inequalities.

3.5.1 Segregated Compartments

We derive the SADP variation of the Segregated Compartments NLP (SSADP) by
using Lagrangian relaxation of the flow balance constraints (3.5) and adding flow
variable upper bounds, resulting in the following model.
SSADP

\[
\min cz - \sum_{k \in Q} \alpha^k f^k
\]

subject to:

\[
f^k \leq u^k, \forall k \in Q \tag{3.31}
\]

\[
\sum_{k \in Q^i} f^k \leq C^l z, \forall l \in K \tag{3.32}
\]

\[
f^k \geq 0, \forall k \in Q \tag{3.33}
\]

\[
z \geq 0, \text{integer.} \tag{3.34}
\]

The notation is the same as that for the segregated CNLP on a general graph, aside from the addition of objective coefficients for flow variables, which are due to Lagrangian penalties of flow imbalance, and the elimination of node and edge indices.

We first note that some results of [34] extend almost directly to the segregated case.

**Proposition 5** The convex hull of the SSADP has full dimension, \(|Q| + 1\).

**Proposition 6** (i) For every commodity \(k \in Q\), \(f^k \geq 0\) defines a facet of the SSADP convex hull.

(ii) For every \(k \in Q\), \(f^k \leq u^k\) defines a facet.

(iii) If \(\sum_{k \in Q} u^k > C^l\), then \(\sum_{k \in Q^i} f^k \leq C^l z\) defines a facet.

**Proposition 7** For a given nonempty set of commodities \(P \subseteq Q^i\), the corresponding residual capacity inequality defines a facet of the SSADP convex hull if and only if \(P\) satisfies the following conditions:

(i) If \(\left\lfloor \frac{\sum_{k \in P} u^k}{C^l} \right\rfloor = 1\), then \(|P| = 1\).

(ii) If \(r^P = C^l\), then \(P = Q^i\).

Propositions 5, 6, and 7 follow from arguments very similar to those used in [34].

We next use a duality argument to show that adding a set of residual capacity inequalities for each facility compartment to this formulation defines its convex hull. The result directly generalizes a result described in [34] for the SADP of the network loading base case. Lemma 5 proves the result for two segregated compartments. A
generalization of Lemma 5 provides an induction step extending the result to an arbitrary number of segregated compartments. For ease of notation, we use $f$ for the flow variables of the commodities $Q^1$ that can occupy the first of the two compartments and $g$ for the flow variables of the commodities $Q^2$ that can occupy the other compartment. $u$ and $v$ represent their respective demands. $l^P$ and $l^R$ are residual capacity inequality right hand sides for $P \subseteq Q^1$ and $R \subseteq Q^2$ of the form (3.30).

**Lemma 5** The following system defines the convex hull of the Segregated Compartments Single Arc Design Problem for two segregated compartments:

\[
\begin{align*}
    f^k & \leq u^k, \forall k \in Q^1 \\
    g^k & \leq v^k, \forall k \in Q^2 \\
    \sum_{k \in Q^1} f^k & \leq C^1 z \\
    \sum_{k \in Q^2} g^k & \leq C^2 z \\
    \sum_{i \in P} f^k - r^P z & \leq l^P, \forall P \subseteq Q^1 \\
    \sum_{j \in R} g^k - r^R z & \leq l^R, \forall R \subseteq Q^2 \\
    f^k & \geq 0, \forall k \in Q^1 \\
    g^k & \geq 0, \forall k \in Q^2 \\
    z & \geq 0.
\end{align*}
\]

**Proof.** Consider the following three primal-dual pairs:

(P1) \[
\begin{align*}
    \min & - \sum_{k \in Q^1} a^{k,1} f^k + b^1 z \\
    \text{subject to:}
    \end{align*}
\]

\[
\begin{align*}
    f^k & \leq u^k, \forall k \in Q^1 \\
    \sum_{k \in Q^1} f^k & \leq C^1 z \\
    \sum_{k \in P} f^k - r^P z & \leq l^P, \forall P \subseteq Q^1 \\
    f^k & \geq 0, \forall k \in Q^2 \\
    z & \geq 0.
\end{align*}
\]
\[ \text{(D1)} \]
\[
\text{max } \sum_{k \in Q^1} u^k p^k + 0 s + \sum_{P \subseteq Q^1} l^P x^P
\]
subject to:
\[
p^k + s + \sum_{P | k \in P} x_P \leq -a^{k,1}, \forall k \in Q^1
\]
\[
-C^1 s - \sum_{P \subseteq Q^1} r^P x^P \leq b^1
\]
\[
p^k \geq 0, \forall k \in Q^1
\]
\[
s \geq 0
\]
\[
x^P \geq 0, \forall P \subseteq Q^1
\]
\[\text{(P2)}\]
\[
\text{min } -\sum_{k \in Q^2} a^{k,2} g^k + b^2 z
\]
subject to:
\[
g^k \leq v^k, \forall k \in Q^2
\]
\[
\sum_{k \in Q^2} g^k \leq C^2 z
\]
\[
\sum_{k \in R} g^k - r^R z \leq l^R, \forall R \subseteq Q^2
\]
\[
g^k \geq 0, \forall k \in Q^2
\]
\[
z \geq 0
\]
\[\text{(D2)}\]
\[
\text{max } \sum_{k \in Q^2} v^k q^k + 0 t + \sum_{R \subseteq Q^2} l^R y^R
\]
subject to:
\[
q^k + t + \sum_{R | k \in R} y^R \leq -a^{k,2}, \forall k \in Q^2
\]
\[
-C^2 t - \sum_{R \subseteq Q^2} r^R y^R \leq b^2
\]
\[
q^k \geq 0, \forall k \in Q^2
\]
\[
t \geq 0
\]
\[
y^R \geq 0, \forall R \subseteq Q^2
\]
\[\text{(P3)}\]
\[
\text{min } -\sum_{k \in Q^1} a^{k,1} f^k - \sum_{k \in Q^2} a^{k,2} g^k + b z
\]
subject to:
\[
f^k \leq u^k, \forall k \in Q^1
\]
\[\text{(3.64)}\]
\[
\begin{align*}
g^k & \leq v^k, \forall k \in Q^2 \\
\sum_{k \in Q^1} f^k & \leq C^1z \\
\sum_{k \in Q^2} g^k & \leq C^2z \\
\sum_{k \in P} f^k - r^Pz & \leq l^P, \forall P \subseteq Q^1 \\
\sum_{k \in R} g^k - r^Rz & \leq l^R, \forall R \subseteq Q^2 \\
f^k & \geq 0, \forall k \in Q^1 \\
g^k & \geq 0, \forall k \in Q^2 \\
z & \geq 0,
\end{align*}
\]

(D3)
\[
\begin{align*}
\max \sum_{k \in Q^1} u^k p^k + \sum_{k \in Q^2} v^k q^k + 0s + 0t + \sum_{P \subseteq Q^1} l^P x^P + \sum_{R \subseteq Q^2} l^R y^R \\
\text{subject to:}
\end{align*}
\]
\[
\begin{align*}
p^k + s + \sum_{P \cap k \in P} x^P & \leq -a^{k,1}, \forall k \in Q^1 \\
q^k + t + \sum_{R \cap k \in Q} y^R & \leq -a^{k,2}, \forall k \in Q^2 \\
-C^1s - C^2t & - \sum_{P \subseteq Q^1} r^P x^P - \sum_{R \subseteq Q^2} r^R y^R \leq b \\
p^k & \geq 0, \forall k \in Q^1 \\
q^k & \geq 0, \forall k \in Q^2 \\
s, t & \geq 0 \\
x^P & \geq 0, \forall P \subseteq Q^1 \\
y^R & \geq 0, \forall R \subseteq Q^2.
\end{align*}
\]

(P3) is the SADP LP relaxation for two segregated compartments, extended with residual capacity inequalities. Similarly, (P1) and (P2) are LP relaxations extended with residual capacity inequalities for SADPs of a single compartment – the standard NLP case considered in [34]. As shown in [34], the constraint sets of (P1) and (P2) define the convex hulls of their corresponding SADPs.

The constraint set of (P3) defines the two compartment SADP convex hull if, for any objective coefficients \((a^1, a^2, b)\) of (P3), the optimal solution value is either unbounded or the problem has an optimal solution with an integer value for \(z\). If \(b < 0\), then (P3) is unbounded, so assume \(b \geq 0\). (P3) is then a bounded problem since \(f\) and \(g\) are bounded. If for some values of \(b^1, b^2 \geq 0\) with \(b^1 + b^2 = b\), (P1) and (P2) share the same optimal integer value for \(z\), then the (P3,D3) solution given by concatenating the (P1,D1) and (P2,D2) solutions, including the integral variable \(z\), is feasible to (P3,D3) and optimal by weak duality.
Consider the optimal \( z \) values for \((P1)\) as a function of \( b^1 \) for \( b^1 \in [0, b] \). \((P1)\) is bounded for \( b^1 \) in this range since \( f \) is bounded. For \( b^1 = 0 \), the problem has arbitrarily large optimal \( z \) values, and since every extreme point of the feasible region has an integral \( z \) value, the optimal \( z \) values will be a decreasing step function of \( b^1 \). A symmetric argument holds for \((P2)\) and \( b^2 \).

Figure 3.5: Optimal \( z \) values to \((P1)\) and \((P2)\) as functions of \( b^1 \) and \( b^2 = b - b^1 \), respectively.

Figure 3.5 illustrates the monotonicity property graphically. We have drawn a possible curve of the optimal \( z \) values to \((P1)\) as a function of \( b^1 \) on the same axes as a possible curve of optimal \( z \) values to \((P2)\) as a function of \( b^2 = b - b^1 \). We can initiate the two functions ((\(P1)\) on the left, \((P2)\) on the right) at the same \( z \) level because all sufficiently large \( z \) are \((P1)\) optimal for \( b^1 = 0 \) and \((P2)\) optimal for \( b^2 = 0 \). Since both curves are nonincreasing, they must intersect at some \((b^1, b^2, z)\) combination for which \( b^1 + b^2 = b \) and \( z \) is integral. \( \square \)

**Theorem 11** The following polyhedron \( \mathcal{P} \) defines the convex hull of the Single Arc Design Problem for a facility with \( n \) segregated compartments.

**polyhedron \( \mathcal{P} \)**

\[
\begin{align*}
\bar{f}^{1} & \leq \bar{u}^{1} \\
\bar{f}^{2} & \leq \bar{u}^{2} \\
& \vdots \\
\sum_{k \in Q^{1}} \bar{f}_{k,1}^{n} & \leq \bar{u}^{n} \\
\sum_{k \in Q^{2}} \bar{f}_{k,2}^{n} & \leq C^{2}z \\
& \vdots \\
\sum_{k \in Q^{n}} \bar{f}_{k,n}^{n} & \leq C^{n}z \\
\sum_{k \in P^{1}} \bar{f}_{k,1}^{n} - \gamma P_{1}z & \leq \bar{v}_{P_{1},1}, \forall P_{1} \subseteq Q^{1}
\end{align*}
\]
\[
\sum_{k \in P^{2}} f^{k,2} - r^{P^{2},2} z \leq l^{P^{2},2}, \forall P^{2} \subseteq Q^{2} \\
\vdots \\
\sum_{k \in P^{n}} f^{k,n} - r^{P^{n},n} z \leq l^{P^{n},n}, \forall P^{n} \subseteq Q^{n}
\]
\[
f^{1}, f^{2}, \ldots, f^{n} \geq 0 \\
z \geq 0.
\]

**Proof.** In Lemma 5, replace problem (P1) with the constraints of \(\mathcal{P}\) and the objective \(cz - \sum_{l=1}^{n} \sum_{k \in Q^{l}} a^{k,l} f^{k,l}\), (P3) with the analogous problem for \(n+1\), and (D1) and (D3) with the dual problems of (P1) and (P3). Then the proof of the lemma shows that if \(\mathcal{P}\) defines the convex hull of the SADP for \(n\) segregated compartments, the constraint set \(\mathcal{P}\) defines the convex hull for an SADP with facilities of \(n+1\) compartments. The result follows by induction. \(\square\)

We can apply the \(O(|Q|)\) residual capacity inequality separation procedure of Atamtürk and Rajan once for each compartment of a segregated compartments SADP to solve its separation problem in polynomial time.

**Proposition 8** We can solve the SSADP separation problem in \(O(\sum_{1 \leq l \leq K} |Q^{l}|)\) time.

In light of the polynomial transformation of the SSADP to the SADP illustrated in Figure 3.6, it seems reasonable that the polyhedral structure of the SSADP is no more complicated than that of the SADP. The transformation permits optimization of the \(n\) compartment segregated problem with the optimization algorithm of Atamtürk and Rajan.

Figure 3.6 shows the disaggregation schematically for a two compartment instance whose first class consists of commodities \(a, b, c\) and whose second class consists of commodities \(i, ii, iii, iv\). The length of each block is proportional to the number of facilities needed to accommodate the demand of the corresponding commodity, and the commodities in each class are sorted from greatest unit contribution to least.
Figure 3.6: Commodity disaggregation underlying the SSADP to SADP transformation. The SSADP instance has two classes with commodities a, b, c, and i, ii, iii, iv, respectively. The single SADP class has commodities 1, 2, 3, 4, 5, 6.

Appendix B gives the details of the transformation. It does not permit transformation of a segregated compartments problem on a general graph to an NLP because we do not know in advance which commodities will flow on each edge.

3.5.2 Nested and Mixed Compartments

This subsection considers residual capacity inequalities for the nested and general CNLP models. We can write a set of residual capacity inequalities for each subset of compartments. The separation procedure of Atamtürk and Rajan [5] can efficiently separate the inequalities for one such fixed compartment subset, but does not directly imply efficient solution of the separation problem as a whole. We solve the nested compartments separation problem by applying their procedure a polynomial number of times, and we cast the general separation problem as an unconstrained minimization of a submodular function. We again assume the total flow of a commodity on an edge is bound by the commodity demand. We have not been able to extend the
segregated compartments SADP convex hull result, Theorem 5, to the nested or general model. We hypothesize that residual capacity inequalities yield the convex hulls of the corresponding variations of the SADP.

Appendix B describes an efficient algorithm to solve the SADP for facilities of two nested compartments. Extension of the analysis for facilities of even 3 nested compartments is noticeably more complicated and seems hardly worthwhile. Binary search is applicable since linear programming sensitivity analysis results imply convexity of the optimal objective value as a function of the capacity loading.

**Separation for Nested Compartments**

For a subset \( J \) of nested compartments, we can write a valid residual capacity inequality for each subset of the commodities that can occupy no compartment outside \( J \). Here we restrict the inequalities that might be violated to a linear number of compartment subsets, specifically those of a nested form. Consequently, we can solve the nested compartments separation problem in polynomial time by applying the procedure of Atamtürk and Rajan to each such subset.

The residual capacity inequality for a subset \( J \) of compartments assume the following form.

\[
\sum_{k \in P} \bar{f}_k - r^{P,J} \leq (\mu^{P,J} - 1)(\sum_{l \in J} C_l - r^{P,J}),
\]

where

\[
\mu^{P,J} = \left[ \frac{\sum_{l \in J} u^k}{\sum_{l \in J} C_l} \right], \text{ and } r^{P,J} = \begin{cases} 
\sum_{l \in J} C_l, & \text{if } (\sum_{k \in P} u^k) \mod (\sum_{l \in J} C_l) = 0, \\
(\sum_{k \in P} u^k) \mod (\sum_{l \in J} C_l), & \text{otherwise.}
\end{cases}
\]

82
The form is as before, except the capacity is now a sum. Recall that a flow variable in
the nested CNLP does not have a compartment index and represents commodity flow
in all of its eligible compartments. In the previous expressions, \( P \) denotes a subset
of the commodities that can occupy no compartment outside \( J \). In particular, \( P \) is a
subset of commodity classes \( \{m, m+1, \ldots | K \} \) (class 1 commodities must be placed
in all compartments, class 2 commodities may be placed in all compartments except
1, \ldots), with \( m \) chosen as the smallest integer for which \( \{m, m+1, \ldots | K \} \subseteq J \).
We can derive this cut by applying to the capacity inequality \( \sum_{k \in P} f_k \leq (\sum_{l \in J} C^l) z \),
the rounding procedure that Magnanti, Mirchandani, and Vachani applied to the
NLP. In general, other commodities might occupy compartments \( J \). Since we need
not place them in compartments \( J \), though, a residual capacity inequality involving
the corresponding flows is not necessarily valid since it would bound compartment
capacity by flows the compartments need not accommodate.

We show that replacing \( \sum_{l \in J} C^l \) with some \( C' \geq \sum_{l \in J} C^l \), and redefining \( r \) and \( \mu \nolinebreak\)
accordingly, yields a residual capacity inequality dominated by one defined by \( \sum_{l \in J} C^l \).
The result implies that we need to check residual capacity inequalities only for nested
compartment subsets \( \{m, m+1, \ldots | K \} \) for \( m = \{1, 2, \ldots | K \} \). The composite
capacity of a compartment subset \( J \) with a sequence break is effectively too high
because any commodity that must occupy capacity in \( J \) must occupy capacity in a
strict subset of \( J \). For simplicity in Lemma 6, we drop superscripts \( P \) and \( J \) from \( r \nolinebreak\)
and \( \mu \).

**Lemma 6** For \((\bar{f}, \bar{z})\) feasible to the nested compartments LP relaxation, the inequality
\[
\sum_{k \in P} f^k - r \bar{z} \leq (\mu - 1)(C - r)
\]
implies the inequality
\[ \sum_{k \in P} f^k - r'z \leq (\mu' - 1)(C' - r'), \]
for \( C < C' \), and \((r, \mu)\) and \((r', \mu')\) defined relative to \( C \) and \( C' \) respectively.

**Proof.** Case 1. \( \mu = \mu' \).
\( \mu = \mu' \) implies \( r > r' \). Let \( u^k \) be the commodity \( k \) demand. The result follows by rewriting the \( C' \) inequality as \( \sum_{k \in P} f^k + r' (\mu' - z) \leq \sum_{k \in P} u^k \). A residual capacity inequality can be violated only if \( \mu > z \) (see the translation of the proofs of Atamtürk and Rajan to our notation in Appendix B.).

Case 2a. \( \mu' < \mu \) and \( C \geq r' \).
Rewriting the inequality as \( \sum_{k \in P} f^k + r' (\mu' - z) \leq \sum_{k \in P} u^k \) shows that the condition \( \mu' > z \) implies that a solution violates the residual capacity inequality only if \( \sum_{k \in P} f^k + C(\mu' - z) > \sum_{k \in P} u^k \). However, \( \sum_{k \in P} f^k + C(\mu' - z) \leq \sum_{k \in P} u^k \) since \( C \mu' \leq \sum_{k \in P} u^k \) (\( \mu' \leq \mu - 1 \) and \( C(\mu - 1) < \sum_{k \in P} u^k \)) and \( \sum_{k \in P} f^k \leq Cz \) by the feasibility of \((\bar{f}, \bar{z})\) to the LP relaxation.

Case 2b. \( \mu' < \mu \) and \( C < r' \).
\[
\begin{align*}
\sum_{k \in P} f^k + r' (\mu' - z) - \sum_{k \in P} u^k \\
= \sum_{k \in P} f^k + r' (\mu' - z) - C'(\mu' - 1) - r' \\
= (\sum_{k \in P} f^k - Cz) + z(C - r') + (C' - r')(1 - \mu') \\
\leq 0.
\end{align*}
\]
All terms in the penultimate line are nonpositive \( (\mu' < \mu \rightarrow \mu \geq 1 \rightarrow \sum_{k \in P} u^k > 0 \rightarrow \mu' \geq 1) \). \( \square \)

**Theorem 12** The nested compartments residual capacity inequality separation problem can be solved by applying the procedure of Atamtürk and Rajan once for each facility compartment.

**Separation for General Compartments**

We next cast the problem of separating residual capacity inequalities for the general problem as a problem of minimizing an unconstrained submodular function. Atamtürk and Rajan effectively transform the NLP separation problem to a single function evaluation. We show that this value, viewed as a function over compartment subsets, is submodular.
As before, let \((\bar{f}, \bar{z})\) represent the flow and loading, respectively, on a given edge that are feasible to a linear programming relaxation of the CNLP. For a set of compartments \(J\) and a set of commodities \(P\), we can write the residual capacity inequality in the following familiar form.

\[
\sum_{i \in J} \sum_{k \in P} \bar{f}^{k,i} - \bar{r} \bar{z} \leq (\mu - 1)(\sum_{i \in J} C^i - r).
\]

In this expression, \(\mu = \lceil \frac{\sum_{k \in P} u^k}{\sum_{i \in J} C^i} \rceil\), and \(r = \begin{cases} \sum_{i \in J} C^i, & \text{if } \sum_{k \in P} \text{mod } \sum_{i \in J} C^i = 0, \\ \sum_{k \in P} \text{mod } \sum_{i \in J} C^i, & \text{otherwise.} \end{cases}\)

Let \(d^k\) be the commodity \(k\) demand, \(C = \sum_{i \in S} C^i\), and the commodity subset \(Q^S = \{k \mid \sum_{i \in S} \bar{f}^{k,i} - (\bar{z} - [\bar{z}])d^k > 0\}\). In Appendix B we translate the arguments of Atamtürk and Rajan into our notation to show that a residual capacity inequality for fixed compartment set \(S\) is violated if and only if \(C[\bar{z}] < \sum_{k \in Q^S} d^k < \sum_{i \in S} C^i[\bar{z}]\), and \(\sum_{i \in S} \sum_{k \in Q^S} \bar{f}^{k,i} - (\bar{z} - [\bar{z}]) \left( \sum_{k \in Q^S} d^k \right) - C[\bar{z}](\lceil \bar{z} \rceil - \bar{z}) > 0\). A result in the appendix implies that if \(\sum_{k \in Q^S} d^k \geq C[\bar{z}]\), then

\[
\sum_{i \in S} \sum_{k \in Q^S} \bar{f}^{k,i} - (\bar{z} - [\bar{z}]) \left( \sum_{k \in Q^S} d^k \right) - C[\bar{z}](\lceil \bar{z} \rceil - \bar{z}) \leq 0.
\]

A similar result holds if \(\sum_{k \in Q^S} d^k \leq C[\bar{z}]\).

**Lemma 7** If \(\sum_{k \in Q^S} d^k \leq C[\bar{z}]\), then

\[
\sum_{i \in S} \sum_{k \in Q^S} \bar{f}^{k,i} - (\bar{z} - [\bar{z}]) \left( \sum_{k \in Q^S} d^k \right) - C[\bar{z}](\lceil \bar{z} \rceil - \bar{z}) \leq 0.
\]

**Proof.** \(\sum_{i \in S} \sum_{k \in Q^S} \bar{f}^{k,i} - (\bar{z} - [\bar{z}]) \left( \sum_{k \in Q^S} d^k \right) - C[\bar{z}](\lceil \bar{z} \rceil - \bar{z}) \leq \sum_{i \in S} \sum_{k \in Q^S} \bar{f}^{k,i} - (\bar{z} - [\bar{z}]) \left( \sum_{k \in Q^S} d^k \right) - (\sum_{k \in Q^S} d^k)(\lceil \bar{z} \rceil - \bar{z}) = \sum_{i \in S} \sum_{k \in Q^S} \bar{f}^{k,i} - \sum_{k \in Q^S} d^k \leq 0\)

Consequently, a solution violates an inequality corresponding to compartments \(S\) if and only if \(g(S) = \lceil \sum_{k \in Q^S} \sum_{i \in S} \bar{f}^{k,i} - \sum_{k \in Q^S} (\bar{z} - [\bar{z}])d^k \rceil - (\sum_{i \in S} C^i) \lceil [\bar{z}] \rceil (\lceil \bar{z} \rceil - \bar{z}) > 0\).

We show next that \(g\) is supermodular.

**Lemma 8** \(g\) is supermodular, that is \(g(S) + g(T) \leq g(S \cup T) + g(S \cap T), \forall S, T \subseteq L\).
Figure 3.7: Commodity subsets defined by compartment subsets $S$ and $T$.

**Proof.** Since $\bar{f} \geq 0$, $Q^{S\cup T} \supseteq Q^S \cup Q^T$ and $Q^{S\cap T} \subseteq Q^S \cap Q^T$. Figure 3.7 depicts the commodity subsets. Let $Q^1 = Q^{S\cup T \setminus (Q^S \cup Q^T)}$, and $Q^2 = (Q^S \cap Q^T) \setminus Q^{S\cap T}$.

\[
g(S \cup T) + g(S \cap T) - g(S) - g(T) + \left[ (\sum_{k \in Q^1} \sum_{l \in S} f_{k,l}^T - \sum_{k \in Q^1} (\bar{z} - [\bar{z}])d^k) \right] + \left[ (\sum_{k \in Q^2 \setminus Q^T} \sum_{l \in T \setminus S} f_{k,l}^T + \sum_{k \in Q^T \setminus Q^S} \sum_{l \in S \setminus T} f_{k,l}^T) \right] + \left[ (\sum_{k \in Q^1} (\bar{z} - [\bar{z}])d^k - \sum_{k \in Q^2} \sum_{l \in S \cap T} f_{k,l}^T) \right] \]

The second term of $g()$ sums over a compartment subset, and, in the previous expression, these terms of $g()$ cancel. The RHS of the difference collects the surviving terms of $g()$ according to commodity subsets defined in Figure 3.7. The first term of the RHS is nonnegative since $\sum_{l \in S \cup T} f_{k,l}^T - (\bar{z} - [\bar{z}])d^k \geq 0 \forall k \in Q^{S\cup T}$. The second term is nonnegative since $\bar{f} \geq 0$. Finally, the third term is nonnegative since commodities $Q^2$ are not in $Q^{S\cap T}$. $\square$

Since we can efficiently evaluate $g$ and minimize submodular functions in polynomial time (see [43], [28]), we can separate residual capacity inequalities for the CNLP in polynomial time.

**Theorem 13** Residual capacity inequality separation for the general CNLP is polynomial.

### 3.6 Problems with a Single Origin-Destination Pair

Magnanti and Mirchandani [33] showed how to solve a network loading problem on a general graph and with only one origin-destination pair as a shortest path problem.
In this brief section, we extend their analysis to the CNLP. We first show that a shortest path solution remains optimal for the single O-D pair (but with an arbitrary number of commodities) CNLP by arguing that a nonbifurcated flow is optimal.

**Theorem 14** For a bounded CNLP with a single O-D pair, it is optimal to send all flow on a shortest O-D path, with lengths given by facility loading costs, and to load the minimum number of facilities needed to accommodate the flow.

**Proof:** Assume nonnegative arc costs; otherwise, the problem is unbounded. We argue that given two paths sharing start and end nodes but having no common internal nodes, and a set of commodity demands that must flow from the start to the end node using only those two paths, it is optimal to send all flow on the shorter path. Repeated application of this result to a proposed solution to the single O-D pair CNLP proves the desired result.

Specifically, suppose $P_1$ and $P_2$ are the two paths that share no internal nodes with respective lengths $L_1$ and $L_2$. Assume without loss of generality that $L_1 \leq L_2$. Suppose that $P_1$ carries commodity flows $f^1, f^2, \ldots, f^K$ and uses $l_1$ facilities on each arc, and that $P_2$ carries commodity flows $g^1, g^2, \ldots, g^K$ and uses $l_2$ facilities on each arc. It is both feasible and no more costly to flow $f^1 + g^1, f^2 + g^2, \ldots, f^K + g^K$ and use $l_1 + l_2$ facilities on each arc of $P_1$. \( \Box \)

Magnanti and Mirchandani use a dual interpretation of Dijkstra's Algorithm to prove that the single O-D pair NLP is solved by loading the shortest path (i.e., the result of Theorem 14) and that nonnegativity constraints and cutset inequalities define the convex hull of feasible loadings (that is, the feasible region projected to the design variables). Their arguments account for negative objective coefficients on the flow variables since their formulation includes design variable upper bounds (such a negative coefficient makes our problem unbounded). Also, they extend the argument to the cases of two and three facility types with "common break-even point" loading costs across the edges. In contrast, multiple compartments in a single facility complicate our analysis.

The max flow-min cut argument of Lemma 9 below shows that nonnegativity constraints and cutset inequalities also determine the set of feasible loadings for the
single O-D pair CNLP. Intuitively, the max flow-min cut result guarantees existence of a set of O-D paths that together carry at least the minimum number of facilities needed to accommodate demand across a single edge, and we can generate a feasible routing by dividing each commodity among the paths in proportion to the facility loadings on the paths.

**Lemma 9** The set of feasible design variables $z$ for a single O-D pair CNLP is:

\[
\sum_{e \in S} z_e \geq z_0, \forall O-D cutsets S \\
z_e \geq 0, \text{ integer},
\]

where $z_0$ is the minimum loading of each O-D cut, as described in Section 3.4.

**Proof.** We first show the result for segregated compartments. Since the problem involves only one O-D pair, we can apply the max flow-min cut result to each compartment separately, and it implies that the compartment $l$ commodities can flow to satisfy the total class $l$ demand $D_l$ if and only if $\sum_{(i,j) \in \delta(S)} z_{ij} \geq \frac{D_l}{C_l}$ for all cuts $S$. The result follows since $z_0 = \max \left\{ \frac{D_l}{C_l} \right\}$ in this case.

Consider next the general model and suppose that $z_e \geq z_0$ for some single edge $e$ (even though the cutset inequalities do not guarantee existence of such an edge). The max flow analysis of cutset inequalities in Section 3.4 shows that all demand could cross such an edge $e$. (Recall that we calculate the minimum loading across a cut by assuming all flow can cross the cut on a single edge.) Assigning commodities to compartments according to this solution across the hypothesized edge $e$ defines a segregated compartments problem. Application of the previous paragraph to this segregated compartments problem yields the result. \(\square\)

Lemma 9 extends the max flow-min cut result to the CNLP by showing that we can route a set of demands between a specified pair of nodes if and only if every cut separating the nodes satisfies the cutset condition. Finally, we note that the cutset inequalities yield the convex hull of feasible loadings $z$.

**Corollary 9** If $z_0$ is the minimum loading of each O-D cut described in Section 3.4, then the following inequalities define the convex hull of feasible design variables $z$ for a single O-D pair CNLP:

\[
\sum_{(i,j) \in \delta(S)} z_{ij} \geq z_0, \forall O-D cutsets S \\
z_{ij} \geq 0.
\]

**Proof.** Assume nonnegative coefficients of the objective function $\sum c_{ij} z_{ij}$; otherwise, the problem is unbounded. The cutset conditions $\sum_{(i,j) \in \delta(S)} z_{ij} \geq z_0$ require $z_0$ loadings of facilities between nodes O and D. With nonnegative edge costs, it is cheapest.
to load all such capacity on a shortest path. Since $z_0$ is integral, these constraints always yield an integral optimal solution. □

Lemma 9 and Corollary 9 provide an alternate optimality proof of loading capacity on a shortest path, and the results effectively convert the problem from one of commodity flow to one of facility flow. The cutset conditions of Corollary 9 require flowing $z_0$ facilities from the origin to the destination.

3.7 Three Node Network Loading

This section considers Compartmentalized Network Loading on a complete graph with only three nodes and extends results of Magnanti, Mirchandani, and Vachani [34]. They note that the prior results of Papernov [39] and Seymour [44] imply that the formulation of a Network Loading Problem of only three nodes, projected onto the space of design variables $z$ is the following model.

$$\min b_{12}z_{12} + b_{13}z_{13} + b_{23}z_{23}$$

subject to:

$$z_{12} + z_{13} \geq (u_{12} + u_{13})/C \quad (3.81)$$
$$z_{12} + z_{23} \geq (u_{12} + u_{23})/C \quad (3.82)$$
$$z_{13} + z_{23} \geq (u_{13} + u_{23})/C \quad (3.83)$$
$$z_{12}, z_{13}, z_{23} \geq 0 \text{ and integer.} \quad (3.84)$$

In this system, $u_{ij}$ is the demand between nodes $i$ and $j$, and $C$ is the common facility capacity. Using duality arguments, they prove that the following inequalities define the convex hull of feasible loadings:

$$z_{12} + z_{13} \geq \lceil (u_{12} + u_{13})/C \rceil \quad (3.85)$$
$$z_{12} + z_{23} \geq \lceil (u_{12} + u_{23})/C \rceil \quad (3.86)$$
\[ z_{13} + z_{23} \geq \left\lfloor \frac{(u_{13} + u_{23})}{C} \right\rfloor \]  \hspace{1cm} (3.87)
\[ z_{12} + z_{13} + z_{23} \geq \left\lfloor \frac{1}{2} \left( \left\lfloor \frac{u_{12} + u_{13}}{C} \right\rfloor + \left\lfloor \frac{u_{12} + u_{23}}{C} \right\rfloor + \left\lfloor \frac{u_{13} + u_{23}}{C} \right\rfloor \right) \right\rfloor \]  \hspace{1cm} (3.88)
\[ z_{12}, z_{13}, z_{23} \geq 0. \]  \hspace{1cm} (3.89)

We next give a direct proof of the projection result, and then a slight extension and shorter proof of the convex hull result. Section 3.7.2 extends the analysis to segregated compartments, and 3.7.3 generalizes the projection result for facilities of two general or two nested compartments.

### 3.7.1 Single Compartment Facilities

**Lemma 10** A loading \( z \) for a three node Network Loading Problem is feasible if and only if it satisfies the cut condition that capacity incident to each node is sufficient to accommodate commodity demands incident to that node. That is, the following inequalities define the set of feasible facility loadings \( z \):

\[ z_{12} + z_{13} \geq \frac{(u_{12} + u_{13})}{C} \]  \hspace{1cm} (3.90)
\[ z_{12} + z_{23} \geq \frac{(u_{12} + u_{23})}{C} \]  \hspace{1cm} (3.91)
\[ z_{13} + z_{23} \geq \frac{(u_{13} + u_{23})}{C} \]  \hspace{1cm} (3.92)
\[ z_{12}, z_{13}, z_{23} \geq 0 \text{ and integer.} \]  \hspace{1cm} (3.93)

**Proof.** The cut conditions are necessary because demand incident to a node is a lower bound on flow incident to that node. To prove sufficiency we construct a feasible flow. If the cut conditions are satisfied, direct flow of as much demand as possible (i.e., the flow on edge 1-2 of the commodity with endpoints 1 and 2 is \( \min \{u_{12}, C_{y_{12}}\} \), and likewise for edges 1-3 and 2-3) leaves at most one of the three demands unsatisfied. A node incident to two unsatisfied demands would not satisfy its cut condition. Assume without loss of generality that only demand 1-2 remains unsatisfied. The node 1 cut condition implies that the remaining capacity on arc 1-3 is sufficient to route the demand to node 3, and the node 2 cut condition implies sufficient capacity remaining on arc 2-3. □

Bienstock and Günlük [13] apply a very similar argument to a different variation of the NLP.

We next prove, for a set of polyhedra slightly more general than the previous cutset inequalities, integrality of the solutions to each constraint set that might define an
extreme point. The convex hull result of Magnanti, Mirchandani, and Vachani [34] cited at the beginning of this section is a special case.

**Theorem 15** If \( v_1, v_2, v_3, \) and \( v_4 \) are integers with \( v_4 \geq \left\lfloor \frac{1}{2} (v_1 + v_2 + v_3) \right\rfloor \), then a polyhedron \( P \) of the form

\[
\begin{align*}
z_{12} + z_{13} & \geq v_1 \quad (3.94) \\
z_{12} + z_{23} & \geq v_2 \quad (3.95) \\
z_{13} + z_{23} & \geq v_3 \quad (3.96) \\
z_{12} + z_{13} + z_{23} & \geq v_4 \quad (3.97) \\
z_{12}, z_{13}, z_{23} & \geq 0 \quad (3.98)
\end{align*}
\]

is integral.

**Proof.** We will prove integrality of all extreme points of a polyhedron \( P \) by showing that a fractional value \( z^* \) cannot be the unique optimal solution to a linear program with feasible region \( P \) and objective function \( \min b_{12}z_{12} + b_{13}z_{13} + b_{23}z_{23} \). Assume \( b_{ij} \geq 0 \); otherwise, the problem is unbounded.

**Case1:** Exactly one \( z_{ij}^* \) is fractional.
Since the \( v_i \) are integral and only one \( z_{ij}^* \) is fractional, the solution obtained by rounding down the fractional \( z_{ij}^* \) is feasible and of no greater objective value.

**Case2:** Exactly two \( z_{ij}^* \) are fractional.
Assume wlog that \( z_{12}^* \) and \( z_{13}^* \) are fractional and that \( b_{12} \leq b_{13} \). Let \( \delta_1 \in (0,1) \) be the fraction so that \( z_{12}^* + \delta_1 \) is integral, \( \delta_2 \in (0,1) \) be the fraction so that \( z_{13}^* - \delta_2 \) is integral, and \( \delta = \min \{ \delta_1, \delta_2 \} \). Then, \((z_{12}^* + \delta, z_{13}^* - \delta, z_{23}^*)\) is of no greater objective value and feasible since the floor (\( \lfloor \cdot \rfloor \)) of no LHS of (3.95) - (3.98) has decreased.

**Case3:** All three \( z_{ij} \) are fractional.
First note that at most two of (3.95) - (3.98) can be tight. Tightness in (3.98) and two of (3.95)-(3.97) implies an integral solution. If all three of (3.95)-(3.97) hold with equality, then the solution is: \( z_{12}^* = \frac{1}{2}(v_1 + v_2 - v_3) \), \( z_{13}^* = \frac{1}{2}(v_1 - v_2 + v_3) \), \( z_{23}^* = \frac{1}{2}(-v_1 + v_2 + v_3) \). In particular, each \( z_{ij} \) must be an odd multiple of \( \frac{1}{2} \), implying slack in 3.98. However, slack in (3.98) implies slack in at least one of (3.95)-(3.97) since \( v_4 \geq \left\lfloor \frac{1}{2} (v_1 + v_2 + v_3) \right\rfloor \).

**Case3a:** Slack in two of (3.95)-(3.97).
Assume wlog slack in (3.95) and (3.96). If \( b_{13} \geq b_{23} \), \((z_{12}^* - \delta, z_{13}^* + \delta, z_{23}^* + \delta)\) is of no greater objective value and feasible for sufficiently small \( \delta \) since the only LHS to decrease is that of (3.95). Similarly if \( b_{13} < b_{23} \), \((z_{12}^* + \delta, z_{13}^* - \delta, z_{23}^* - \delta)\) is a cheaper solution and feasible for sufficiently small \( \delta \).

**Case3b:** Slack in (3.98) and one of (3.95) - (3.97).
Assume wlog slack in (3.95) and (3.98). If \( b_{23} \geq b_{12} + b_{13} \), \((z_{12}^* + \delta, z_{13}^* - \delta, z_{23}^* + \delta)\) is a feasible solution of no greater cost. If \( b_{23} < b_{12} + b_{13} \), \((z_{12}^* - \delta, z_{13}^* + \delta, z_{23}^* - \delta)\) is a cheaper solution and feasible for sufficiently small \( \delta \). □
3.7.2 Segregated Compartments

We next extend the convex hull result of Magnanti, Mirchandani, and Vachani [34] to a three node Network Loading Problem with segregated compartments.

**Lemma 11** The following system describes the set of feasible design variables $z$ to the segregated compartments CNLP on a 3 node graph.

\[
\begin{align*}
    z_{12} + z_{13} & \geq \frac{(u_{12}^l + u_{13}^l)}{C^l}, \forall l \\
    z_{12} + z_{23} & \geq \frac{(u_{12}^l + u_{23}^l)}{C^l}, \forall l \\
    z_{13} + z_{23} & \geq \frac{(u_{13}^l + u_{23}^l)}{C^l}, \forall l \\
    z_{12}, z_{13}, z_{23} & \geq 0 \text{ and integer.}
\end{align*}
\]

$C^l$ is the compartment $l$ capacity of a facility, and $u_{ij}^l$ is the demand between nodes $i$ and $j$ that can be accommodated only in compartments $l$.

**Proof.** Apply the single compartment NLP result Lemma 10 to each compartment.

Max notation, integer roundup of RHSs, and application of a Gomory-Chvátal cut converts the system to:

\[
\begin{align*}
    z_{12} + z_{13} & \geq \left[ \max_i \frac{u_{12}^l + u_{13}^l}{C^l} \right] = v_1 \\
    z_{12} + z_{23} & \geq \left[ \max_i \frac{u_{12}^l + u_{23}^l}{C^l} \right] = v_2 \\
    z_{13} + z_{23} & \geq \left[ \max_i \frac{u_{13}^l + u_{23}^l}{C^l} \right] = v_3 \\
    z_{12} + z_{13} + z_{23} & \geq \left[ \frac{1}{2} (v_1 + v_2 + v_3) \right] \\
    z_{12}, z_{13}, z_{23} & \geq 0 \text{ and integer.}
\end{align*}
\]

Finally, application of Theorem 15 permits elimination of the integrality constraints.

**Corollary 10** The convex hull of feasible loadings for a three node segregated compartments Network Loading Problem is given by (3.103)-(3.106) and nonnegativity.
3.7.3 Two Compartments, Nested or General

We next extend the three node projection result to NLPs with two nested or two general compartments. We consider first the general case, with some commodities permitted in only the first compartment of a facility, some in only the second compartment, and the rest in both.

We formulate this two compartment problem for a graph $G = (N, A)$ as follows.

**CNLP for Two General Compartments**

$$\min \sum b_{ij} z_{ij}$$

subject to:

$$\sum_{j} f_{ij}^k - \sum_{j} f_{ji}^k = \begin{cases} u^k & \text{if } i = O^k \\ -u^k & \text{if } i = D^k \\ 0 & \text{otherwise} \end{cases} \tag{3.108}$$

$$\sum_{j} g_{ij}^l - \sum_{j} g_{ji}^l = \begin{cases} v^l & \text{if } i = O^l \\ -v^l & \text{if } i = D^l \\ 0 & \text{otherwise} \end{cases} \tag{3.109}$$

$$\sum_{j} h_{ij}^m - \sum_{j} h_{ji}^m = \begin{cases} w^m & \text{if } i = O^m \\ -w^m & \text{if } i = D^m \\ 0 & \text{otherwise} \end{cases} \tag{3.110}$$

$$\sum_k (f_{ij}^k + f_{ji}^k) \leq C'^I z_{ij}\forall \{i, j\} \in A \tag{3.111}$$

$$\sum_l (g_{ij}^l + g_{ji}^l) \leq C'^I z_{ij}\forall \{i, j\} \in A \tag{3.112}$$

$$\sum_k (f_{ij}^k + f_{ji}^k) + \sum_l (g_{ij}^l + g_{ji}^l) + \sum_m (h_{ij}^m + h_{ji}^m) \leq (C' + C'^I) z_{ij} \tag{3.113}$$

$$f_{ij}^k \geq 0, \forall (i, j) \in A, \forall k \tag{3.114}$$

$$g_{ij}^l \geq 0, \forall (i, j) \in A, \forall l \tag{3.115}$$

$$h_{ij}^m \geq 0, \forall (i, j) \in A, \forall m \tag{3.116}$$

$$z_{ij} \geq 0, \text{integer}. \tag{3.117}$$

Flows $f$, $g$, and $h$ can occupy compartment I only, compartment II only, and either compartment, respectively, and $u, v,$ and $w$ are the corresponding demands. In the
following discussion, we refer to the commodities that can occupy only compartment I as $f$ class commodities, and the others as $g$ and $h$ class commodities. $C^I$ and $C^{II}$ are the compartment capacities. The general mixed compartments model of Section 3.2 has a set of flow variables for each potential compartment of each variable. With only two compartments, it seems easier to use only two variables (forward and reverse) per edge for each commodity since the capacity conditions are easily enumerated without a flow variable compartment index.

We next describe the set of feasible loadings. The inequality system includes cutset conditions as well as inequalities accounting for the additional capacity requirement of demand that cannot flow directly (i.e., on one edge) from origin to destination and must flow on the other two edges. We construct a feasible flow to show sufficiency of the system.

**Theorem 16** The set of feasible facility loadings $z$ for a three node, two compartment Network Loading Problem is:

\[
\begin{align*}
C^I(z_{12} + z_{13}) & \geq u_{12} + u_{13} \\
C^I(z_{12} + z_{23}) & \geq u_{12} + u_{23} \\
C^I(z_{13} + z_{23}) & \geq u_{13} + u_{23} \\
C^{II}(z_{12} + z_{13}) & \geq v_{12} + v_{13} \\
C^{II}(z_{12} + z_{23}) & \geq v_{12} + v_{23} \\
C^{II}(z_{13} + z_{23}) & \geq v_{13} + v_{23} \\
(C^I + C^{II})(z_{12} + z_{13}) & \geq u_{12} + u_{13} + v_{12} + v_{13} + w_{12} + w_{13} \\
(C^I + C^{II})(z_{12} + z_{23}) & \geq u_{12} + u_{23} + v_{12} + v_{23} + w_{12} + w_{23} \\
(C^I + C^{II})(z_{13} + z_{23}) & \geq u_{13} + u_{23} + v_{13} + v_{23} + w_{13} + w_{23} \\
(C^I + C^{II})(z_{12} + z_{13}) + 2C_I z_{23} & \geq u_{12} + u_{13} + v_{12} + v_{13} + w_{12} + w_{13} + 2v_{23} \\
(C^I + C^{II})(z_{12} + z_{23}) + 2C_I z_{13} & \geq u_{12} + u_{23} + v_{12} + v_{23} + w_{12} + w_{23} + 2v_{13} \\
(C^I + C^{II})(z_{13} + z_{23}) + 2C_I z_{12} & \geq u_{13} + u_{23} + v_{13} + v_{23} + w_{13} + w_{23} + 2v_{12}
\end{align*}
\]
\[(C' + C'') (z_{12} + z_{13}) + 2C_{11} z_{23} \geq u_{12} + u_{13} + v_{12} + v_{13} + w_{12} + w_{13} + 2v_{23} \quad (3.130)\]

\[(C' + C'') (z_{12} + z_{23}) + 2C_{11} z_{13} \geq u_{12} + u_{23} + v_{12} + v_{23} + w_{12} + w_{23} + 2v_{13} \quad (3.131)\]

\[(C' + C'') (z_{13} + z_{23}) + 2C_{11} z_{12} \geq u_{13} + u_{23} + v_{13} + v_{23} + w_{13} + w_{23} + 2v_{12} \quad (3.132)\]

\[z_{12}, z_{13}, z_{23} \geq 0 \text{ and integer.} \quad (3.133)\]

**Proof.** Cut conditions, (3.118) - (3.126) are necessary because demand incident to a node is a lower bound on flow incident to that node. We next show necessity of (3.127). Cut condition (3.124) is necessary, so suppose \(u_{23} > C' z_{23}\). The flow on edges 1-2 and 1-3 of the \(f\) class commodity with O-D pair 2-3 must then be at least \(u_{23} - C' z_{23}\). In turn, the flow incident to node 1 must be at least \((u_{13} + u_{23}) + (v_{13} + v_{23}) + (w_{13} + w_{23}) + 2(u_{23} - C' z_{23})\). Necessity follows. (3.128) - (3.132) are similarly necessary.

We prove sufficiency by showing feasibility of the solution generated by first routing as much of each \(f\) class commodity as possible on its direct one edge path, routing the rest indirectly (i.e., on two edges), and then \(g\) class commodities and then \(h\) class commodities in the same manner. The composite class \(f\) flows are:

\[f_{12} = \min \{u_{12}, (C' z_{12}) \} + \max \{0, u_{13} - C' z_{13}\} + \max \{0, u_{23} - C' z_{23}\}\]

\[f_{13} = \min \{u_{13}, (C' z_{13}) \} + \max \{0, u_{12} - C' z_{12}\} + \max \{0, u_{23} - C' z_{23}\}\]

\[f_{23} = \min \{u_{23}, (C' z_{23}) \} + \max \{0, u_{12} - C' z_{12}\} + \max \{0, u_{13} - C' z_{13}\}\]

where, for example, \(\min \{u_{12}, (C' z_{12}) \}\) is the edge 1-2 flow of the class \(f\) commodity with O-D pair 1-2, \(\max \{0, u_{13} - C' z_{13}\}\) is the edge 1-2 flow of the class \(f\) commodity with O-D pair 1-3, and \(\max \{0, u_{23} - C' z_{23}\}\) is the edge 1-2 flow of the class \(f\) commodity with O-D pair 2-3. Similarly, the composite class \(g\) flows are:

\[g_{12} = \min \{v_{12}, (C_{11} z_{12}) \} + \max \{0, v_{13} - C_{11} z_{13}\} + \max \{0, v_{23} - C_{11} z_{23}\}\]

\[g_{13} = \min \{v_{13}, (C_{11} z_{13}) \} + \max \{0, v_{12} - C_{11} z_{12}\} + \max \{0, v_{23} - C_{11} z_{23}\}\]

\[g_{23} = \min \{v_{23}, (C_{11} z_{23}) \} + \max \{0, v_{12} - C_{11} z_{12}\} + \max \{0, v_{13} - C_{11} z_{13}\}\]

Finally, the composite class \(h\) flows are:

\[h_{12} = \min \{w_{12}, ((C_1 + C_{11}) z_{12}) - f_{12} - g_{12}\} + \max \{0, w_{13} - ((C_1 + C_{11}) z_{13} - f_{13} - g_{13})\} + \max \{0, w_{23} - ((C_1 + C_{11}) z_{23} - f_{23} - g_{23})\}\]

\[h_{13} = \min \{w_{13}, ((C_1 + C_{11}) z_{13}) - f_{13} - g_{13}\} + \max \{0, w_{12} - ((C_1 + C_{11}) z_{12} - f_{12} - g_{12})\} + \max \{0, w_{23} - ((C_1 + C_{11}) z_{23} - f_{23} - g_{23})\}\]

\[h_{23} = \min \{w_{23}, ((C_1 + C_{11}) z_{23}) - f_{23} - g_{23}\} + \max \{0, w_{12} - ((C_1 + C_{11}) z_{12} - f_{12} - g_{12})\} + \max \{0, w_{13} - ((C_1 + C_{11}) z_{13} - f_{13} - g_{13})\}\]

First, flow balance conditions are satisfied. For example, the edge 1-2 flow of the class \(f\) commodity with O-D pair 1-2 is \(\min \{u_{12}, C' z_{12}\}\), and the edges 1-3 and 2-3 flows are \(\max \{u_{12} - C' z_{12}, 0\}\). These quantities sum to the demand \(u_{12}\). The single compartment result Lemma 10 applied to constraints (3.118) - (3.123) implies that flows \(f_{12}, f_{13}, f_{23}\) and \(g_{12}, g_{13}, g_{23}\), respectively, do not violate the compartment capacity conditions, (3.111) and (3.112).

It remains to show satisfaction of the composite capacity conditions, (3.113). Observing, for example, that \(z_{12} + z_{23} \geq \frac{u_{12} + u_{13} + v_{12} + v_{13} + w_{12} + w_{13}}{C_1 + C_{11}}\) and \(z_{13} + z_{23} \geq 95\)
\[
\begin{align*}
\left[ u_{13} + w_{23} + v_{12} + v_{13} + w_{13} + w_{12} \right] & \implies (C_I + C_{II})(z_{12} + z_{13}) + 2(C_I + C_{II})z_{23} \geq u_{12} + u_{13} + v_{12} + v_{13} + w_{12} + w_{13} + 2(u_{23} + v_{23}), (3.124) - (3.132) \text{ can be rewritten as:} \\
(C_I + C_{II})(z_{12} + z_{13}) \geq & \quad (u_{12} + u_{13}) + (v_{12} + v_{13}) + (w_{12} + w_{13}) + \\
& 2 \max \{0, (u_{23} - C_I z_{23}), (v_{23} - C_{II} z_{23})\} \\
(C_I + C_{II})(z_{12} + z_{23}) \geq & \quad (u_{12} + u_{23}) + (v_{12} + v_{23}) + (w_{12} + w_{23}) + \\
& 2 \max \{0, (u_{13} - C_I z_{13}), (v_{13} - C_{II} z_{13})\} \\
(C_I + C_{II})(z_{13} + z_{23}) \geq & \quad (u_{13} + u_{23}) + (v_{13} + v_{23}) + (w_{13} + w_{23}) + \\
& 2 \max \{0, (u_{12} - C_I z_{12}), (v_{12} - C_{II} z_{12})\}.
\end{align*}
\]

The maximum of the first inequality reflects the three \(z_{12} + z_{13}\) inequalities, (3.124), (3.127), and (3.130), as well as the roundup observation. The RHSs of (3.134) - (3.136) equal:
\[
\begin{align*}
w_{12} + w_{13} + f_{12} + f_{13} + g_{12} + g_{13}, \\
w_{12} + w_{23} + f_{12} + f_{23} + g_{12} + g_{23}, \\
w_{13} + w_{23} + f_{13} + f_{23} + g_{13} + g_{23}.
\end{align*}
\]

Lemma 10 implies:
\[
\begin{align*}
(C_I + C_{II})z_{12} \geq & \quad h_{12} + f_{12} + g_{12}, \\
(C_I + C_{II})z_{13} \geq & \quad h_{13} + f_{13} + g_{13}, \\
(C_I + C_{II})z_{23} \geq & \quad h_{23} + f_{23} + g_{23},
\end{align*}
\]
and \(h_{12}, h_{13}, h_{23}\) can be accommodated in the capacity not occupied by \(f_{12}, f_{13}, f_{23}\) and \(g_{12}, g_{13}, g_{23}\), that is, satisfaction of (3.113). \(\square\)

A similar argument yields the projection of the nested two compartment case, with some commodities permitted in only one compartment and the rest of the commodities in both. The formulation follows, and the three node projection result is a special case of Theorem 16.

**CNLP for Two Nested Compartments**

\[
\min \sum b_{ij} z_{ij}
\]

subject to:

\[
\sum_j f^k_{ij} - \sum_j f^k_{ji} = \begin{cases} u^k & \text{if } i = O^k \\ -u^k & \text{if } i = D^k \\ 0 & \text{otherwise} \end{cases}
\]  
(3.137)
\[
\sum_j g_{ij}^l - \sum_j g_{ji}^l = \left\{ \begin{array}{cl} v^l & \text{if } i = O^l \\ -v^l & \text{if } i = D^l \\ 0 & \text{otherwise} \end{array} \right. \tag{3.138}
\]
\[
\sum_k (f_{ij}^k + f_{ji}^k) \leq C^I z_{ij} \tag{3.139}
\]
\[
\sum_k (f_{ij}^k + f_{ji}^k) + \sum_l (g_{ij}^l + g_{ji}^l) \leq (C^I + C^{II}) z_{ij} \tag{3.140}
\]
\[
f_{ij}^k \geq 0, \forall (i, j) \in A, \forall k \tag{3.141}
\]
\[
g_{ij}^l \geq 0, \forall (i, j) \in A, \forall l \tag{3.142}
\]
\[
z_{ij} \geq 0, \text{ integer.} \tag{3.143}
\]

**Corollary 11** The set of feasible facility loadings \( z \) for a three node, two nested compartment Network Loading Problem is:

\[
C^I (z_{12} + z_{13}) \geq (u_{12} + u_{13}) \tag{3.144}
\]
\[
C^I (z_{12} + z_{23}) \geq (u_{12} + u_{23}) \tag{3.145}
\]
\[
C^I (z_{13} + z_{23}) \geq (u_{13} + u_{23}) \tag{3.146}
\]
\[
(C^I + C^{II}) (z_{12} + z_{13}) \geq (u_{12} + u_{13}) + (v_{12} + v_{13}) \tag{3.147}
\]
\[
(C^I + C^{II}) (z_{12} + z_{23}) \geq (u_{12} + u_{23}) + (v_{12} + v_{23}) \tag{3.148}
\]
\[
(C^I + C^{II}) (z_{13} + z_{23}) \geq (u_{13} + u_{23}) + (v_{13} + v_{23}) \tag{3.149}
\]
\[
(C^I + C^{II}) (z_{12} + z_{13}) + 2C^I z_{23} \geq (u_{12} + u_{13}) + (v_{12} + v_{13}) + 2u_{23} \tag{3.150}
\]
\[
(C^I + C^{II}) (z_{12} + z_{23}) + 2C^I z_{13} \geq (u_{12} + u_{23}) + (v_{12} + v_{23}) + 2u_{13} \tag{3.151}
\]
\[
(C^I + C^{II}) (z_{13} + z_{23}) + 2C^I z_{12} \geq (u_{13} + u_{23}) + (v_{13} + v_{23}) + 2u_{12} \tag{3.152}
\]
\[
z_{12}, z_{13}, z_{23} \geq 0 \text{ and integer.} \tag{3.153}
\]

Application of the technique of Theorem 16, namely, considering feasibility conditions of a flow prioritizing direct shipments, to three node problems with facilities of more than two compartments seems more difficult because of increased combinatorial complexity between flows and compartments.

### 3.8 Solution Approach

To solve the CNLP, we embedded the CPLEX solver in a cutting plane and branch and bound solution approach similar to the approach we implemented for Pup Match-
ing. The procedure solves the linear programming relaxation as formulated in Section 3.2, then tightens the relaxation by adding residual capacity, cutset, and three partition inequalities, in that order. We did not choose this sequence for any particular reason. However, small tests seem to indicate that, while the sequence for adding cuts might impact the solution efficiency of a given instance, alternate sequences does not uniformly improve computational effectiveness. Each separation module iteratively searches for cuts and resolves the linear program until an iteration yields insufficient improvement of the objective value. We then heuristically determine an initial feasible solution and finally call the CPLEX branch and bound routine. We limited branch and bound to a search tree of 220M and 125,000 nodes, and conducted our study on a Pentium III with a 733 MHz clock speed and 256 MB of RAM.

We exactly separate residual capacity inequalities of segregated compartments problems by applying the procedure of Atamtürk and Rajan [5] to each compartment on each edge. Similarly, we apply their procedure to each of the nested compartment subsets described in Section 3.5.2 to exactly separate residual capacity inequalities of the nested model. For simplicity, we apply brute force to the general model by running the Atamtürk and Rajan procedure for each subset of compartments. None of our tests involves a general model with more than 4 compartments, so separation through the submodular minimization described in Section 3.5.2 likely offers little, if any, computational gain.

We separate the cutset inequalities heuristically. Bienstock [11] has shown that exact cutset separation for a variation of the single facility NLP is $NP$-complete. We first append each cutset inequality defined by a singleton, or lone, node. We
then iteratively find a Gomory-Hu tree (see [25], [2]) to identify additional important cuts, and append the inequalities defined by previously undiscovered cuts. As we mentioned in Chapter 2, Balakrishnan, Magnanti, Sokol, and Wang [6] search for cutset inequalities using a similar technique.

Our three partition inequality separation routine is relatively crude. Each search cycle considers all partitions with two of the three node subsets defined by 2, 3, 4, or 5 nodes. To hopefully prevent the addition of many ineffective inequalities, we resolve the LP without considering larger subsets if we find a violated inequality for one subset size.

To find a feasible initial solution, we apply two heuristics and retain the better solution. The first, edge rounding, simply rounds up each design variable of an LP relaxation to the ceiling of its current value to obtain a feasible integer solution. We apply this procedure to the initial LP relaxation, as well as to each extended LP relaxation obtained at the end of a separation module, and retain the best solution. The second heuristic iteratively resolves the LP relaxation obtained after adding the relevant cutting planes. At each iteration, we force a new solution by increasing the lower bound of the fractional design variable nearest its ceiling to the ceiling, and then resolve. The procedure terminates upon obtaining an integer, and so feasible, solution. Bienstock et al. [12] employed a similar procedure.
3.9 Computational Results

We analyzed the computational difficulty of Compartmentalized Network Loading in a series of four tests on randomly generated problem instances. The first three tests consider the segregated model exclusively, and the fourth considers all three models. In all four tests, we generated commodity demands uniformly. Section 3.9.1 summarizes the test results, and Section 3.9.2 discusses them.

3.9.1 Tests

Test One, Graph Density

The first set of tests considers graphs of 10 and 12 nodes at four levels of graph density. A density of 0 corresponds to a spanning tree, a density of 1 corresponds to a complete graph, and the metric is linear in the number of edges for intermediate values. We randomly generated five segregated instances for each node cardinality, density combination. We selected node locations uniformly over a rectangle. We selected edges uniformly with replacement but rejected edges to ensure no parallel edges and that the first \( n - 1 \) form a spanning a tree. Edge lengths equal internode Euclidean distances, and each instance has four compartments. We selected each commodity demand uniformly (integral) on \([1, 20]\) and compartment capacity uniformly on \([1, 12]\). Figure 3.8 plots, for both the 10 and 12 node series, graph density against branch and bound solution time averaged over the five instances.
Figure 3.8: Branch and bound solution time vs. graph density for test instances of 10 and 12 nodes.

**Test Two, Number of Compartments**

The second tests investigate the computational impact of the number of compartments on the segregated model. We generated five complete 8 node graphs and five complete 10 node graphs, and for each graph generated a segregated instance for each of 1, 2, 4, 8, 16, and 32 compartments. A single compartment problem is equivalent to a standard NLP. The problem generator maintained the expected total number of commodities, as well as the expected total capacity of each facility essentially independent of the number of compartments. We generated new demands and capacities for each instance, but the upper bounds on the number of commodities per facility and on the compartment capacity distribution are inversely proportional to the number of compartments in the instance. Figure 3.9 plots branch and bound computation time against the log of the number of compartments for the 8 and 10 node tests, respectively. Figure 3.10 plots the corresponding final (that is, just prior to calling branch and bound) integrality gaps.
Figure 3.9: Solution time vs. number of compartments for complete graphs of 8 and 10 nodes.

Figure 3.10: Final integrality gap vs. number of compartments for complete graphs of 8 and 10 nodes.
Test Three, Demand Density

The third tests investigate the computational impact of demand density on the segregated model. We define demand density as the ratio of expected commodity demand to expected compartment capacity. We generated five complete 10 node graphs and, for each graph, six sets of facility compartment values to achieve six different demand densities. Each facility has four segregated compartments. Figure 3.11 plots the branch and bound solution time against the log of demand density, and 3.12 plots the corresponding final integrality gaps.

![Graph showing solution time vs. demand density.](image)

Figure 3.11: Solution time vs. demand density.

Test Four, Facility Complexity

The fourth series of tests investigates the relative computational difficulty of the three CNLP models - segregated, nested, and general. We generated eight complete 10 node graphs, and for each graph, generated a four compartment instance of each model. For each graph, all three models have the same total number of commodities. Figure 3.13 graphs the average branch and bound solution time and the corresponding number of
required branch and bound nodes for each model. Figure 3.14 graphs the pre-branch and bound and heuristic gaps.

![Graph of final integrality gap vs. demand density.](image)

**Figure 3.12**: Final integrality gap vs. demand density.

![Bar charts showing solution times and average number of branch and bound nodes for each facility type.](image)

**Figure 3.13**: Solution times and average number of branch and bound nodes for each facility type.

### 3.9.2 Discussion of Results

Test 1 and Figure 3.8 support the intuition that instances defined on larger and denser graphs are harder to solve. The concavity of the plot for 12 node problems
Figure 3.14: Average integrality and heuristic gap for each facility type.

likely reflects the 125,000 node limit of branch and bound more than a leveling of solution times. Three (of twenty) 10 node instances hit the node limit without a provably optimal solution, and all three were on complete graphs. Four of the 12 node instances hit the node limit, two with \( \frac{2}{3} \) density and two on complete graphs.

The negative slopes of Figure 3.9 indicate that increasing the number of compartments decreases computational difficulty. On the 8 node tests, the average branch and bound time for instances with only 1 compartment is 67.7 seconds, and it was nearly 0 for instances of 16 and 32 compartments. Similarly, on the 10 node tests, the average time for 1 compartment is 2638.6 seconds, compared to near 0 for 32 compartments. The tightness of the LP relaxation seems to account partly for this computational trend. Figure 3.10 indicates that instances with smaller numbers of compartments tend to have larger integrality gaps. The LP relaxation (with no additional cuts) of a
single compartment problem allocates capacity to exactly accommodate flow on each edge. Compartmentalization effectively forces the LP relaxation to account for unused capacity, since capacity must be allocated for all compartments simultaneously. The LP relaxation integrality gaps fall almost monotonically with the number of compartments. For 10 node instances, the gaps for 16 and 32 compartments are about 1% each, while the gap for 1 compartment instances is 29.3%. Given this argument, though, we do not understand why the 2 compartment problems exhibit the highest final branch and bound integrality gaps.

Figure 3.11 exhibits relatively large solution times at intermediate demand densities. At sufficiently low demand densities, the problem reduces to a connectivity problem. Although the corresponding instances exhibit relatively high integrality gaps, connectivity seems combinatorially less difficult than the general flow and connectivity problem. At high densities capacity is effectively more continuous, and the LP relaxation should yield a good approximation. The negative slope of the integrality gap versus demand density plot in Figure 3.12 supports this interpretation. The average gap falls from 19.2% for instances with density $\frac{1}{10}$ to less than 0.6% for instances with density 4. The corresponding average gaps for the raw LP relaxations (not shown in a figure) are 68.0% and 1.1%. The increase in solution time from a density of 2 to a density of 4 is surprising, though we can attribute it to a single difficult instance.

The bar graphs of test four results in Figure 3.13 support the intuition that a more complicated facility corresponds to a more difficult combinatorial problem. For each graph, the segregated and nested models entail the same number of variables. The
corresponding general model has the same number of integer facility variables, but more continuous flow variables. However, the general models required significantly more branch and bound nodes on average, in addition to just more computational time. Interestingly, as indicated in Figure 3.14, the heuristics performed worst on the segregated model. The average error was 9.9% versus 7.6% and 9.1% for the other models. The final integrality gaps, however, increase with the facility complexity, leading us to argue that a strong lower bound is more important to provably solving these problems than a strong upper bound. Also, the high 25% average integrality gap for the general model likely indicates that we have overlooked a structurally significant class of inequalities for this problem.

In general, like the NLP (see [35]), the CNLP is more easily solved for certain problem parameter settings such as high demand, many compartments, and segregated facilities, that correspond to smaller integrality gaps. The CNLP also seems relatively easy to solve for situations with low demand, where both the NLP and CNLP reduce to connectivity problems. As expected, and like the NLP (see [14]), the CNLP seems harder to solve on denser graphs.

3.10 Summary of Results

Since the results of this chapter are relatively diverse, we conclude with a summary. We first motivated Compartmentalized Network Loading as a generalization of Network Loading, and then formulated three variations – problems with segregated, nested, and general compartments. Section 3.3 considered two linear programs, the
LP relaxation of the CNLP itself and a compartmentalized variation of the multicommodity flow problem. We noted that we recognize no familiar structure to help solve the former, and we described a transformation of the latter problem to the familiar multicommodity flow problem.

Sections 3.4 and 3.5 generalized to the CNLP two families of Network Loading inequalities, cutset inequalities and residual capacity inequalities. We used a max flow analysis to derive a cutset inequality for each CNLP variation, and we observed that, under fairly general conditions, a cutset inequality defines a facet of the relevant CNLP convex hull. In Section 3.5, we used a duality argument to show that residual capacity inequalities yield the convex hull of a segregated compartments variation of the Single Arc Design Problem (SADP). This result extends one of Magnanti, Mirchandani, and Vachani [34], who introduced the SADP. We also solved the residual capacity inequality separation problem for all three CNLP variations by applying the NLP separation result of Atamtürk and Rajan [5].

Sections 3.6 and 3.7 examined two special cases of the CNLP, respectively, the problem with a single origin-destination pair and the problem on a three node network. We showed that we can solve the single O-D pair problem by loading capacity on a shortest path and that cutset inequalities define the convex hull of the problem projected to the space of the design variables. These results generalize some of those described by Magnanti and Mirchandani [33]. In Section 3.7, we showed that a generalization of the NLP three partition inequalities leads to the convex hull of the three node segregated compartments problem, again projected to the space of the design variables. This result generalizes another result of Magnanti, Mirchandani,
and Vachani [34]. We also described the projection to design variables of the feasible region of the three node problem of two nested or general compartments.

Section 3.8 described our branch and bound solution approach. Section 3.9 outlined our test problems and summarized the computational results.
Chapter 4

Single Commodity Network

Loading

Most studies of network loading consider models that ship demand between prespecified origin-destination pairs and so require capacity sufficient for carrying multicommodity flow. An alternate model specifies demand origins, supply destinations, and magnitudes, but does not à priori pair the origins and destinations. That is, the model requires capacity sufficient for carrying standard (single commodity) network flow. This chapter considers this latter variation, which we refer to as the Single Commodity Network Loading Problem (SCNLP). The Single Commodity Pup Matching of Chapter 2 is a special case.

Section 4.1 develops three SCNLP formulations and discusses special properties of the problem derived from its network flow structure. Section 4.2 considers the limiting cases of continuous and large capacity, and Section 4.3 presents approximation results. Section 4.4 describes inequalities valid for the SCNLP, and Section 4.5 develops a
solution procedure for the special case of the SCNLP over a single cut. Finally, Section 4.6 summarizes an initial computational study.

4.1 Formulations

This section first formulates Single Commodity Network Loading in three ways. Section 4.1.1 develops a “full” formulation with both design and flow variables. Section 4.1.2 applies cutset properties to obtain a formulation with only design variables, and Section 4.1.3 develops a nonlinear formulation with only flow variables. In Section 4.1.4 we note properties that follow from the underlying network flow structure of the SCNLP and then prove a negative result that an extended formulation based upon these properties does not tighten the full formulation.

4.1.1 Full Formulation

Probably the most natural formulation of the SCNLP is the following NLP variation involving both flow and design variables.

\[
\min \sum_{(i,j) \in A} c_{ij} z_{ij}
\]

subject to:

\[
\sum_{j \in N} f_{ji} - \sum_{j \in N} f_{ij} = d_i, \forall i
\]  \hspace{1cm} (4.1)

\[
f_{ij} + f_{ji} \leq C z_{ij}, \forall (i, j) \in A,
\]  \hspace{1cm} (4.2)

\[
f_{ij} \geq 0, \forall (i, j) \in A
\]  \hspace{1cm} (4.3)

\[
z_{ij} \geq 0, \text{ integer, } \forall (i, j) \in A.
\]  \hspace{1cm} (4.4)

Variable \( f_{ij} \) is the flow from node \( i \) to \( j \) on edge \( (i, j) \) of undirected graph \( G = (N, A) \). Each edge requires only forward and reverse flow variables since the problem has only
one underlying commodity. Variable $z_{ij}$ is the number of facilities loaded on edge $(i,j)$, and each facility provides capacity $C$. Parameter $d_i$ is the net demand at node $i$. Constraints (4.1) balance flow around each node; constraints (4.2) bound the total flow on each edge by the loaded capacity; we wish to minimize the total cost of loaded capacity.

### 4.1.2 Cutset Formulation

The max flow-min cut result, as applied by Gale [20], implies that a facility loading $z$ permits feasible flow if and only if it provides capacity sufficient for the net demand of every node subset to cross the corresponding cut. This result permits us to formulate the SCNLP with no flow variables but exponentially many cut conditions.

$$\min \sum_{(i,j) \in A} c_{ij} z_{ij}$$

subject to:

$$C \sum_{(i,j) \in \delta(S)} z_{ij} \geq d_s, \forall S \subset N$$

$$z_{ij} \geq 0, \text{integer.}$$

(4.5) (4.6)

$\delta(S)$ is the edge cut defined by node subset $S$, and $d_s = |\sum_{i \in S} d_i|$ is the magnitude of net demand within nodes $S$.

### 4.1.3 Flow Formulation

Although the rest of the chapter focuses on the full and cut formulations, for completeness, we develop a nonlinear formulation that involves only flow variables and is analogous to the NLP Flow Formulation of Epstein [19]. Whenever some facility
loading cost \( c_{ij} \) is negative, the SCNLP is unbounded. Otherwise, some optimal solution loads the minimum number of facilities needed on each edge to accommodate flow on that edge. Consequently, we can replace each design variable with a step function in total flow on its edge and obtain the following formulation.

\[
\min \sum_{(i,j) \in A} c_{ij} \left[ \frac{f_{ij} + f_{ji}}{C} \right]
\]

subject to:

\[
\sum_{j \in N} f_{ji} - \sum_{j \in N} f_{ij} = d_i, \forall i \tag{4.7}
\]

\[
f_{ij} \geq 0. \tag{4.8}
\]

Constraints (4.7) again balance flow around each node.

### 4.1.4 Special Properties

We next state several properties of the SCNLP based upon the observation that under fixed capacity the problem reduces to network flow feasibility.

**Proposition 9** The Single Commodity Network Loading Problem has an optimal solution with:

1. unidirectional flow, if any, on each edge,
2. integral flow values, and
3. flow on all edges with unused capacity restricted to a tree.

It seems attractive to exploit observation (3) by extending the full formulation as follows to include variables defining those edges that contain unused capacity.

**Tree Formulation**

\[
\min \sum_{(i,j) \in A} c_{ij} z_{ij}
\]

subject to:

\[
\sum_{j \in N} f^k_{ji} - \sum_{j \in N} f^k_{ij} = d_i, \forall i \tag{4.9}
\]
\[
\begin{align*}
    f_{ij} + f_{ji} + s_{ij} &= Cz_{ij}, \forall\{i, j\} \in A, \quad (4.10) \\
    s_{ij} &\leq (C - 1)t_{ij}, \forall\{i, j\} \in A, \quad (4.11) \\
    \sum_{\{i,j\} \in A} t_{ij} &= n - 1, \quad (4.12) \\
    \sum_{\{i,j\} \in \gamma(S)} t_{ij} &\leq |S| - 1, \forall S \subset N \quad (4.13) \\
    A(f, z) &\leq b \quad (4.14) \\
    f_{ij} &\geq 0, \forall (i, j) \in A \quad (4.15) \\
    s_{ij} &\geq 0, \forall (i, j) \in A \quad (4.16) \\
    z_{ij} &\geq 0, \text{integer, } \forall (i, j) \in A \quad (4.17) \\
    t_{ij} &\text{ binary, } \forall (i, j) \in A. \quad (4.18)
\end{align*}
\]

Variable \(t_{ij}\) indicates whether edge \((i, j)\) can hold unused capacity, and \(s_{ij}\) is the corresponding (continuous) quantity of unused capacity. \(\gamma(S)\) is the set of edges incident to two nodes in \(S\), so (4.12) and (4.13) append constraints of the subtour elimination formulation of the Minimum Spanning Tree Problem (see, for example, [10]). The matrix \(A\) defines additional constraints involving only original variables \((f, z)\) that we might add to tighten the LP relaxation of the full formulation as stated in Section 4.1.1. Section 4.4 describes several families of such inequalities. We show next that the Tree Formulation is no tighter than the corresponding full formulation (i.e., with constraints defined by the matrix \(A\)).

**Theorem 17** The optimal value of the Tree Formulation equals that of the full formulation of Section 4.1 extended by the inequalities defined by the matrix \(A\).

**Proof.** The value of the Tree Formulation is no lower than that of the original formulation, since the \((f, z)\) components of any feasible solution to the tree formulation are also feasible to the full formulation.

We argue next that the value is no higher. Consider the following Lagrangian relaxations of the original and tree formulations defined by dualizing the capacity constraints.

**Initial Formulation**

\[
L_0(\lambda) = \min \sum_{\{i,j\} \in A} c_{ij}z_{ij} - \sum_{\{i,j\} \in A} \lambda_{ij}(Cz_{ij} - f_{ij} - f_{ji})
\]

114
subject to:

\[
\sum_{j \in N} f_{ji}^k - \sum_{j \in N} f_{ij}^k = d_i, \forall i \tag{4.19}
\]

\[
A(f,z) \leq b \tag{4.20}
\]

\[
f_{ij} \geq 0, \forall (i,j) \in A \tag{4.21}
\]

\[
z_{ij} \geq 0, \text{integer, } \forall (i,j) \in A \tag{4.22}
\]

**Tree Formulation**

\[L_t(\lambda) = \min \sum_{(i,j) \in A} c_{ij} z_{ij} - \sum_{(i,j) \in A} \lambda_{ij} (C z_{ij} - f_{ij} - f_{ji} - s_{ij})\]

subject to:

\[
\sum_{j \in N} f_{ji} - \sum_{j \in N} f_{ij} = d_i, \forall i \tag{4.24}
\]

\[
s_{ij} \leq (C - 1) t_{ij}, \forall \{i, j\} \in A \tag{4.25}
\]

\[
\sum_{\{i,j\} \in A} t_{ij} = n - 1, \tag{4.26}
\]

\[
\sum_{\{i,j\} \in \delta(S)} t_{ij} \leq |S| - 1, \forall S \subseteq N \tag{4.27}
\]

\[
A(f,z) \leq b \tag{4.28}
\]

\[
f_{ij} \geq 0, \forall (i,j) \in A \tag{4.29}
\]

\[
s_{ij} \geq 0, \forall (i,j) \in A \tag{4.30}
\]

\[
z_{ij} \geq 0, \text{integer, } \forall (i,j) \in A \tag{4.31}
\]

\[
t_{ij} \text{, binary, } \forall (i,j) \in A \tag{4.32}
\]

For any \(\lambda\) and any feasible solution to the original formulation, we can find a solution to the tree formulation with the same value by setting \(s_{ij} = 0\) for all edges, using any binary values \(t\) feasible to (4.25) - (4.27), and \((f, z)\) as the feasible solution to the initial formulation.

Let \(L_0^*\) be the optimal value to the original Lagrangian relaxation and \(L_t^*\) be the optimal value to the tree Lagrangian relaxation. Also, let \(L_0(\lambda)\) and \(L_t(\lambda)\) be the optimal value for a specific \(\lambda\) to the initial and tree formulations.

\[
L_t^* = \max_\lambda L_t(\lambda) \\
\leq \max_\lambda L_0(\lambda) \\
= L_0^*
\]

The inequality follows from the preceding paragraph. Since linear programming is strongly Lagrangian (i.e., has no Lagrangian duality gap), the optimal value of the tree formulation equals that of the original formulation. \(\square\)

Theorem 17 also holds if we instead add constraints of the spanning tree cut.
formulation.

4.2 Extreme Cases

In this section, we first note that with continuous capacities the SCNLP reduces to a minimum cost network flow problem, and then we conceive of the SCNLP with a sufficiently large facility capacity as a variation of the Steiner Network Tree Problem that we refer to as the Balanced Demand Steiner Forest problem (BDSF). We develop two formulations for the BDSF. The latter casts the problem within the constrained forest problem class considered by Goemans and Williamson [23], implying a 2-approximation.

**Proposition 10** The linear programming relaxation of the Single Commodity NLP is a min cost network flow problem, with unit flow costs \( c_{ij} \), defined by the loading cost \( c_{ij} \) on edge \((i, j)\) and the capacity per facility \( C \).

**Proposition 11** If \( C \) is sufficiently large (\( C \geq \sum_{d_i \geq 0} d_i \) will suffice), the Single Commodity NLP is equivalent to finding the minimum cost forest with the property that each tree of the forest has 0 net demand, and with the cost of each edge as the loading cost \( c_{ij} \).

We next define the problem implied by Proposition 11 as the BDSF variation of the Steiner Network Tree Problem.

**Balanced Demand Steiner Forest Problem (BDSF) Instance:** An undirected graph \( G = (N, A) \), a net demand \( d_i \) for \( i \in N \), and a nonnegative cost function \( c : A \to \mathbb{Q}^+ \).

**Problem:** Find the minimum cost forest \( T \) of \( G \) such that \( \sum_{i \in T_j} d_i = 0 \), for each tree \( T_j \) of the forest.

**Lemma 12** The BDSF is \( NP \)-hard.

**Proof.** If there is only one demand or supply node, all demand and supply nodes must be connected, and the BDSF becomes the Steiner Network Tree Problem. \( \square \)
Corollary 12 The SCNLPS is NP-hard.

Given our motivation of the BDSF as a limiting case of the SCNLPE, we might most naturally formulate it as in Section 4.1.1, with the understanding that the capacity C is sufficiently large. We refer to this formulation as BDSF1. Alternatively, we cast a formulation from cut conditions.

BDSF2
\[
\min \sum_{e \in A} c_e z_e
\]
such that:
\[
\sum_{e \in \delta(S)} z_e \geq g(S), \forall S \subset N, S \neq \emptyset \tag{4.33}
\]
\[
z_e \in \{0, 1\}, \forall e \in A, \tag{4.34}
\]
with \(g(S) = \begin{cases} 1, & \text{if } \sum_{i \in S} d_i \neq 0 \\ 0, & \text{if } \sum_{i \in S} d_i = 0. \end{cases}\)

The function \(g\) indicates the loading requirement on a cut. We must select at least one edge in a cut corresponding to a node subset of unbalanced demand.

We argue next that BDSF1 and BDSF2 are equivalent integer programming formulations, and that the LP relaxation of BDSF2 is tighter than that of BDSF1.

Lemma 13 BDSF2 is the projection onto the space of the design variable \(z\) of the BDSF1 feasible region.

Proof. Suppose \((z^*, f^*)\) is a feasible solution to BDSF1. Flow must cross cut \(\delta(S)\) if \(\sum_{i \in S} d_i \neq 0\), implying that \(\sum_{e \in \delta(S)} z_e^* \geq 1\), and that \(z^*\) is feasible to BDSF2. Conversely, suppose \(z^+\) is feasible to BDSF2. Since \(\sum_{e \in \delta(S)} z_e^+ \geq 1\) whenever \(\sum_{i \in S} d_i \neq 0\) and since \(C\) is sufficiently large, we can assume \(C\) sufficiently large, max flow-min cut implies existence of some flow \(f^+\) so that \((z^+, f^+)\) is feasible to BDSF1. \(\square\)

Lemma 14 If \(P_1\) is the set of feasible \(z\) values to the LP relaxation of BDSF1 obtained by replacing each binary restriction with \(0 \leq z_{ij} \leq 1\), and \(P_2\) is the set of feasible solution to the LP relaxation of BDSF2, then \(P_2 \subseteq P_1\), and, in some instances, \(P_2 \subset P_1\).

Proof. The argument in the second half of the proof of Lemma 13 shows that if \(z \in P_2\), then \(z \in P_1\). The containment can be strict because a feasible flow might not
require a full facility on each cut corresponding to unbalanced demand. Figure 4.1 depicts such an instance. Nodes 1 and 4 each have a supply of 1, nodes 2 and 3 each have a demand of 1, and $C = 2$. Loading $\frac{1}{4}$ facilities on each edge permits the feasible flow of $\frac{1}{2}$ units on each edge, but violates each cut condition of BDSF2 corresponding to a singleton node. □

![Figure 4.1: The BDSF2 LP relaxation of this instance strictly contains the BDSF1 LP relaxation.](image)

Furthermore, BDSF2 casts the Balanced Demand problem within the class of constrained forest problems considered by Goemans and Williamson [23]. Specifically, given a graph $G = (N, A)$, a proper function $h : 2^N \to \{0,1\}$, and a nonnegative cost function $c : A \to \mathbb{Q}_+$, they consider problems of the following form.

$$\min \sum_{e \in A} c_e x_e$$

subject to:

$$\sum_{e \in \delta(S)} x_e \geq h(S), \forall S \subseteq N, S \neq \emptyset \quad (4.35)$$

$$x_e \in \{0, 1\}, \forall e \in A. \quad (4.36)$$

The function $h$ is proper if

1. (symmetry) $h(S) = h(N \setminus S)$, $\forall S \subseteq N$, and
2. (disjointness) \( h(S) = h(T) = 0 \) implies \( h(S \cup T) = 0 \) for disjoint \( S \) and \( T \).

Goemans and Williamson describe a procedure that achieves a \( (2 - \frac{2}{|M|}) \)-approximation algorithm for any proper constrained forest problem, when \( M = \{ v \in N \mid h(\{v\}) = 1 \} \).

We can verify that the function \( g \) of formulation BDSF2 satisfies the above properness criteria.

**Corollary 13** If \( k \) is the number of demand nodes plus the number of supply nodes, the general approximation technique for constrained forest problems of Goemans and Williamson yields a \( (2 - \frac{2}{k}) \) -approximation algorithm for the BDSF.

The BDSF is a generalization of the point-to-point connection problems described by Li, McCormick, and Simchi-Levi [32] and mentioned by Goemans and Williamson as a problem fitting their approximation framework. In our terms, the point-to-point connection problem is a BDSF whose demands and supplies all have value 1.

### 4.3 Approximation of the SCNLP

Mansour and Peleg [36] describe an \( O(\log n) \) approximation algorithm for the Network Loading Problem based on light-weight, low stretch spanner subgraphs. They also show that the approximation is as tight as possible, unless \( P = NP \). In this brief section, we extend some of their analysis to the SCNLP.

A spanner subgraph combines properties of shortest path trees and minimum spanning trees. The total weight of a spanner is relatively low, and shortest path distances in a spanner are not much greater than they are on the original graph. Specifically, let \( G = (N,A) \) be an undirected graph with edge weights \( c_e \), and let \( \text{dist}_G(i,j) \) denote the length of a shortest path in \( G \) between nodes \( i \) and \( j \). A subgraph
$G' = (N, A')$ is light-weight if $\sum_{e \in A'} c_e$ is low relative to that of a minimum spanning tree of $G$, and the stretch of a spanning subgraph $G'$ is the ratio $\max_{i,j \in N} \left\{ \frac{\text{dist}_{G'}(i,j)}{\text{dist}_G(i,j)} \right\}$.

Subgraph $G'$ is a $\kappa$-spanner for $G$ if its stretch is no more than $\kappa$. Mansour and Peleg employ a result of [3] and [16] that an efficient greedy algorithm constructs a spanner $G' = (N, A')$ for $G$ satisfying the properties that $\text{stretch}(G) \leq \log(|N|)$, $|A'| = \mathcal{O}(|N|)$, and $\sum_{e \in A'} c_e = \mathcal{O}(\log(|N|) \sum_{e \in T} c_e)$. In the last expression, $T$ denotes a minimum spanning tree of $G$.

Lemma 15, which considers restricting an arbitrary network flow problem to a spanner, and a mild adaptation of the arguments of Mansour and Peleg imply that the following procedure yields an $\mathcal{O}(\log n)$ approximation for SCNLP instances requiring a solution with full connectivity of the underlying graph $G = (N, A)$.

**Single Commodity NLP Approximation Algorithm**

1. Apply the greedy spanner algorithm of [3] to obtain a $\log n$-spanner of the Single Commodity NLP graph.

2. Solve the LP relaxation of the Single Commodity problem (a min cost network flow problem) over the spanner and route the flows according to this solution. Let $\tilde{f}$ denote the optimal flow.

3. Load each edge of the spanner with the minimum sufficient capacity $\left( \left\lceil \frac{\tilde{f}_{ij} + \tilde{f}_{ji}}{c_{ij}} \right\rceil \right)$ facilities.

**Lemma 15** *Restriction of a min cost network flow problem to a $\log n$ spanner of its underlying graph increases the optimal cost by no more than a factor of $\log n$.*

**Proof.** Rerouting each O-D flow of an optimal solution to the original problem along the shortest O-D path of the spanner increases cost by no more than a factor of $\log n$ and defines a feasible solution to the restricted problem. □

**Corollary 14** *The SCNLP Approximation Algorithm yields an $\mathcal{O}(\log n)$ approximation for SCNLP instances requiring a solution with full connectivity of the underlying graph $G = (N, A)$.*

**Proof.** The origin-destination pairs of step (2) define an NLP, and step (3) effectively applies the procedure of Mansour and Peleg to that NLP. Mansour and Peleg view
the total cost as the sum of two components, a routing cost of utilized capacity and a cost of capacity unused due to integrality requirements. Lemma 15 implies that our routing cost is $O(\log n)$ of the lowest possible routing cost. The spanning tree property of spanners and the connectivity assumption imply that the cost of the unused capacity is $O(\log n)$ of the minimum total cost. Consequently, the total cost of the SCNLP Approximation Algorithm is $O(\log n)$ of optimal. □

**Corollary 15** Unless $\mathcal{P} = \mathcal{NP}$, the tightest possible SCNLP approximation is $O(\log n)$.

**Proof.** Mansour and Peleg show by transformation of the set cover problem to a single origin NLP that unless $\mathcal{P} = \mathcal{NP}$, the tightest possible NLP approximation is $O(\log n)$. The result follows since the NLP and SCNLP are equivalent in the single origin case. □

We have not been able to extend the results of Mansour and Peleg beyond this full connectivity case. A constant gap between the optimal objective values of the SCNLP and the NLP defined in step (2) would do so, and Corollary 15 would then imply that the SCNLP Approximation Algorithm is within a constant factor of the tightest possible approximation, assuming $\mathcal{P} \neq \mathcal{NP}$.

Finally, we show that rounding up the SCNLP LP relaxation provides a $C$-approximation.

**Lemma 16** If $C$ denotes the facility capacity, then the procedure of rounding up the design variables of the SCNLP LP relaxation yields a $C$-approximation.

**Proof.** Solving the LP relaxation as a minimum cost network flow problem yields an optimal solution with integral flows, so the heuristic solution value is at worst $C$ times the optimal objective value.

On the other hand, the graph of Figure 4.2 shows that $C$ is tight. In Figure 4.2, one unit of flow must be sent from node 0 to each of nodes $1, 2, \ldots, C$. The number beside each edge indicates facility loading cost, and the capacity of each facility is $C$. The LP relaxation sends each unit of flow on its shortest path of length : so the round up heuristic yields value $C$. The optimal solution, though, sends all flow via node 1 for a cost of $1 + (C - 1)\epsilon$. □

As noted by Epstein [19], this same result applies to the NLP.
Figure 4.2: The LP round up heuristic provides a solution of cost $C$. The optimal solution sends all flow via node 1 for a cost of $1 + (C - 1)\varepsilon$.

4.4 Valid Inequalities

4.4.1 Cutset Inequalities

Cutset inequalities assume the form $\sum_{e \in \delta(S)} z_e \geq \left\lfloor \frac{D_S}{C} \right\rfloor$, which is defined by the facility capacity $C$ and the magnitude of net demand within nodes $S$, $D_S$. A slight adaptation of the proof of Magnanti, Mirchandani, and Vachani [35] shows that the inequality defines a facet of the SCNLP full formulation convex hull if $D_S$ is not a multiple of $C$ and the subgraphs defined by $S$ and $N \setminus S$ are connected.

**Theorem 18** The cutset inequality $\sum_{e \in \delta(S)} z_e \geq \left\lfloor \frac{D_S}{C} \right\rfloor$ is facet defining for the full formulation of the Single Commodity NLP if $D_S$ is not a multiple of $C$ and the subgraphs defined by $S$ and $N \setminus S$ are connected.

Nonnegativity, arc capacity constraints, and flow balances imply the cutset inequality corresponding to a demand $D_S$ that is a multiple of $C$.

A similar argument shows that the cutset inequality defines a facet of the SCNLP cutset formulation convex hull if $D_S > 0$ and the subgraphs defined by $S$ are connected.

**Theorem 19** The cutset inequality $\sum_{e \in \delta(S)} z_e \geq \left\lfloor \frac{D_S}{C} \right\rfloor$ defines a facet of the cutset formulation of the Single Commodity NLP convex hull if $D_S > 0$ and the subgraphs defined by $S$ and $N \setminus S$ are connected.
4.4.2 Residual Capacity Inequalities

Since Single Commodity Network Loading involves only one commodity, each edge of the underlying graph has only one residual capacity inequality:

\[ f_f + f_r - rz \leq (\mu - 1)(C - r). \]

The variables \( f_f \) and \( f_r \) are the forward and reverse flows, respectively, on the edge, and \( \mu \) is an upper bound on the number of facilities required on the relevant edge. Parameter \( r \) is the residual, the quantity of flow that would occupy the \( \mu \)th facility if the edge achieved its maximum conceivable flow. In general, the total demand can bound flow volume, so we can set \( \sum_{i=1}^n \max\{d_i, 0\} = C\mu + r \), with \( \mu \) integral and \( 0 < r \leq C \). In specific problem instances we might find a tighter bound on flow volume and then redefine \( \mu \) and \( r \) by substituting the bound for the above left hand side.

Lagrangian relaxation of the flow balance constraints (4.1) of the full formulation permits separation by edges and, in turn, the following special case of the Single Arc Design Problem (SADP) introduced by Magnanti, Mirchandani, and Vachani [34].

**Single Commodity SADP**

\[
\min cz + bf \\
\text{subject to:}
\]

\[
\begin{align*}
    f &\leq d &\quad (4.37) \\
    f &\leq Cz &\quad (4.38) \\
    f &\geq 0 &\quad (4.39) \\
    z &\geq 0, \text{ integer.} &\quad (4.40)
\end{align*}
\]

The result in [34] implies that addition of the residual capacity inequality defines the convex hull of this problem.
Corollary 16 The addition of the residual capacity inequality to the previous formulation defines the convex hull of the Single Commodity SADP.

4.4.3 Three Partition Inequalities

We can write three partition inequalities in the familiar form

$$z_{12} + z_{13} + z_{23} \geq \left\lfloor \frac{1}{2} (v_1 + v_2 + v_3) \right\rfloor.$$

In this expression, \(v_i = \frac{w_i}{C}\); \(u_i\) is the magnitude of the net demand of node set \(i\), and \(C\) is the facility capacity. These inequalities seem weaker than their multicommodity NLP counterparts since they must effectively allow for the lowest possible flow among partitions, whereas the NLP problem statement predetermines flow volume across a partition. In particular, we can always load only two of the three edges of an SCNLP. Nevertheless, Example 7 shows that single commodity three partition inequalities are not implied by cutset inequalities and optimality conditions.

Example 7 The supplies, demands, and arc costs are shown in Figure 4.3, and the facility capacity \(C\) is 2. The cutset conditions are:

\[
\begin{align*}
&z_{12} + z_{13} \geq 1, \\
&z_{12} + z_{23} \geq 1, \\
&z_{13} + z_{23} \geq 1,
\end{align*}
\]

and the three partition inequality is:

\[
\begin{align*}
z_{12} + z_{13} + z_{23} \geq 2.
\end{align*}
\]

\((z_{12}, z_{13}, z_{23}) = (1, 0, 1)\) and \((0, 1, 1)\) are optimal integer solutions, with cost 9. The solution \(\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)\) has a cost of 7, satisfies the cutset conditions, but violates the three partition inequality.

Finally, Theorem 15 of Chapter 3 shows that the cutset conditions and three partition inequality define the convex hull of the three node SCNLP cutset formulation. The result also follows as a special case of the result of Magnanti, Mirchandani, and
Vachani [34] for the NLP on a graph of three nodes, again, since the NLP and SCNLP are equivalent for single origin problems.

**Corollary 17** The following system defines the convex hull of the cutset formulation of the three node SCNLP.

\[
\begin{aligned}
z_{12} + z_{13} &\geq \frac{s}{C} = v_1 \\
z_{12} + z_{23} &\geq \frac{d_2}{C} = v_2 \\
z_{13} + z_{23} &\geq \frac{d_3}{C} = v_3 \\
z_{12} + z_{13} + z_{23} &\geq \frac{v_1 + v_2 + v_3}{2} \\
z_{12}, z_{13}, z_{23} &\geq 0.
\end{aligned}
\]

We assume without loss of generality that node 1 has supply \( s \), node 2 demand \( d_2 \), and node 3 demand \( d_3 \).

### 4.4.4 Odd Flow Inequalities

Generalizations of the odd flow inequalities that we developed for Pup Matching are valid for Single Commodity Network Loading in the sense that they are satisfied by some optimal solution (though might cut off some feasible solutions). Recall that Pup Matching is defined on a directed graph, and the SCNLP on an undirected graph.
**Theorem 20** Let $A_i$ be the set of edges incident to node $i$, and $r$ the magnitude of node $i$ demand modulus the facility capacity $C$. Then, the inequality
\[ (C \sum_{(i,j) \in A_i} z_{(i,j)}) - (\sum_{(i,j) \in A_i} f_{(i,j)} + f_{(j,i)}) \geq \min\{r, C - r\}, \]
is satisfied by at least one optimal solution.

**Proof.** The inequality places a lower bound on the unused capacity incident to node $i$. Since we can assume unidirectional flow (Proposition 9, part 1), and consolidating several flows to a single edge can only decrease the unused capacity required to accommodate the flows, we need to consider only the case of all inflow arriving on one edge and all outflow departing on some other edge. In this case, the incident unused capacity is minimized if either the inflow or the outflow is a multiple of facility capacity since, otherwise, increasing flow would decrease unused capacity on both edges. The expressions in the $\min\{}$ equal the unused capacity in the limiting cases of multiple flow. $\square$

The argument does not require integral flows (Proposition 9, part 2). The odd flow inequality for node $i$ is a sum of the edge capacity constraints if the net demand is a multiple of $C$.

The following example illustrates that odd flow inequalities are not valid for the NLP itself on an undirected graph even under a constraint requiring nonbifurcated flows, because parallel flow might result from the multicommodity character of the problem.

**Example 8** Suppose a multicommodity, nonbifurcated flow Network Loading Problem has facility capacity 4, capacity loading cost 1, and four commodities, $A$, $B$, $C$, and $D$, with respective demands of 3, 3, 1, and 1, with origin-destination pairs as shown in the figure. The figure also shows an optimal flow using one facility per edge. Each commodity flows on a shortest path, and the flow utilizes all loaded capacity. Since the magnitude of net demand at no node is a multiple of the capacity 4, an odd flow inequality might be written for each node. However, each inequality would require unused capacity and so force a solution of cost greater than 4.

We cannot effectively negate the countercurrent flows as we might in an SCNLP, since they here represent different commodities.
Figure 4.4: Instance illustrating that odd flow inequalities are not valid for the multi-commodity NLP on an undirected graph, even for nonbifurcated flow.

4.5 Single Cut Problem

In this brief section we describe the optimal solution for the special case of the SC-NLP across a single cut. Atamtürk [4] describes the solution and convex hull of the corresponding directed problem.

**Single Cut Problem**

\[
\min \sum_{a \in A} (c_a z_a + a_a f_a^+ + b_a f_a^-)
\]

subject to:

\[
\sum_{a \in A} (f_a^+ - f_a^-) = b \tag{4.46}
\]

\[
f_a^+ + f_a^- \leq C z_a, \forall a \in A \tag{4.47}
\]

\[
f_a^+ f_a^- \geq 0 \tag{4.48}
\]

\[
z_a \geq 0, \text{integer.} \tag{4.49}
\]

\(A\) is the set of undirected edges. Lemmas 17 - 21 characterize extreme points of the convex hull of this mixed integer program. Lemma 22 describes unboundedness conditions, and Theorem 21 describes an optimal solution. In the proofs of these results, \(\delta\) represents a sufficiently small positive number.

**Lemma 17** An extreme point of the convex hull can have at most one edge that does not flow at capacity, that is, at most one edge \(a\) for which \(f_a^+ + f_a^- < C z_a\).
**Proof.** A solution with \( f_a^+ = f_a^- = 0 \) and \( z_a > 0 \) is not an extreme point. Increasing and decreasing \( z_a \) creates other solutions feasible to the convex hull whose linear combination is the first solution.

Next, suppose \( f_a^+ + f_a^- < Cz_a, f_a^+ + f_a^- < Cz_a', \) and \( f_a^+ > 0, f_a^- > 0 \). The solutions given by changing \( f_a^+ = f_a^+ + \delta \) and \( f_a^- = f_a^- - \delta \), or \( f_a^+ = f_a^+ - \delta \) and \( f_a^- = f_a^- + \delta \), ceteris paribus, are also feasible, and a linear combination of them yields the initial solution.

Situations with \( f_a^+ > 0, f_a^- > 0 \) and \( f_a^+ > 0, f_a^- > 0 \) are analogous. □

**Lemma 18** If an extreme point has parallel flow on an edge \( a \), that is, an edge \( a \) with \( f_a^+ > 0 \) and \( f_a^- > 0 \), then \( f_a^+ + f_a^- = Cz_a \).

**Proof.** Suppose \( f_a^+ + f_a^- < Cz_a \). Then the solutions given by changing \( f_a^+ = f_a^+ + \delta \), \( f_a^- = f_a^- + \delta \), and by changing \( f_a^+ = f_a^+ - \delta \), \( f_a^- = f_a^- - \delta \), are also feasible, and a linear combination of them yields the initial solution. □

**Lemma 19** An extreme point solution can have at most one edge with parallel flow.

**Proof.** By Lemma 18, \( f_a^+ + f_a^- = Cz_a \) for any edge \( a \) with parallel flow. Suppose \( f_a^+, f_a^-, f_a', f_a^- > 0 \), and \( f_a^+ + f_a^- = Cz_a \) and \( f_a^+ + f_a^- = Cz_a' \). The solutions given by changing \( f_a^+ = f_a^+ + \delta \), \( f_a^- = f_a^- - \delta \), \( f_a'^+ = f_a'^+ - \delta \), \( f_a'^- = f_a'^- + \delta \), \( f_a^+ = f_a^+ - \delta \), \( f_a^- = f_a^- + \delta \), \( f_a'^+ = f_a'^+ + \delta \), \( f_a'^- = f_a'^- - \delta \), are also feasible, and a linear combination of them yields the initial solution. □

**Lemma 20** An extreme point with no parallel flow has at most one C-fractional flow, that is, an edge \( a \) such that \( f_a^+ + f_a^- \) is not a multiple of the capacity \( C \).

**Proof.** As in the other results, we can generate two additional feasible points by transferring flow from one fractional flow to another. Since an extreme point has integer values \( z \), an edge with fractional flow has slack capacity. □

**Lemma 21** An extreme point with parallel flow has no edge with C-fractional flow.

**Proof.** Suppose arc \( a \) has parallel flow. By Lemma 18, \( f_a^+ + f_a^- = Cz_a \), and by Lemma 19, no other arc has parallel flow. Suppose \( a' \) has fractional flow with \( f_{a'}^+ > 0 \). Then the solutions given by changing \( f_a^+ = f_a^+ + \delta \), \( f_a^- = f_a^- - \delta \), \( f_{a'}^+ = f_{a'}^+ - 2\delta \), and \( f_{a'}^- = f_{a'}^- + \delta \), \( f_{a'}^- = f_{a'}^- + 2\delta \), are both feasible. The case of \( f_{a'}^- > 0 \) is analogous. □

**Lemma 22** The Single Cut Problem is unbounded if and only if either \( c_a + \frac{C}{2}(a_a + b_a) < 0 \) for some edge \( a \), \( c_a + Ca_a + c_{a'} + Cb_{a'} < 0 \) for some pair of edges \( a, a' \), or \( c_a < 0 \) for some edge \( a \).

**Proof.** By cycling flow on an edge \( a \) with costs \( c_a + \frac{C}{2}(a_a + b_a) < 0 \) or around a pair of edges \( a, a' \) with costs \( c_a + Ca_a + c_{a'} + Cb_{a'} < 0 \), we can make the objective arbitrarily small. Similarly, if \( c_a < 0 \) for some \( a \), by loading an increasing number of facilities on edge \( a \), we can make the objective arbitrarily small.

128
To show the converse, assume that $c_a \geq 0$ for all edges $a$. Note that we can then transform the LP relaxation of the single cut problem to a directed shortest path problem since the LP then has an optimal solution with $f^+_a + f^-_a = Cz_a$ on all edges $a$. Figure 4.5 illustrates the transformation for a Single Cut Problem of three edges.

Now, if the Single Cut Problem is unbounded, its LP relaxation is unbounded. The result follows because a shortest path problem is unbounded only if it contains a negative cost cycle, and the shortest path version of the single cut LP relaxation has a negative cost cycle only if the original Single Cut Problem has an edge $a$ for which $c_a + \frac{C}{2}(a_a + b_a) < 0$ or a pair of edges $a, a'$ with $c_a + Ca_a + c_{a'} + Cb_{a'} < 0$. □

**Theorem 21** If $r = C$ and the Single Cut Problem is bounded, it has an optimal solution of cost $\left(\frac{b}{C}\right)\min_{a \in A}(c_a + Ca_a)$ achieved by sending $b$ units of flow forward on any edge corresponding to the argument of the min.

If $r < C$ and the problem is bounded, the problem has an optimal solution with cost equal to the cheapest of

1. $\left(\frac{b}{C}\right)\min_{a \in A}(c_a + Ca_a) + \min_{a \in A}(c_a + ra_a)$,
2. $\left(\frac{b}{C}\right)\min_{a \in A}(c_a + Ca_a) + \min_{a \in A}(c_a + (C-r)b_a)$,
3. $\left(\frac{b}{C}\right)\min_{a \in A}(c_a + Ca_a) + \min_{a \in A}(c_a + \frac{C+r}{2}a_a + \frac{C-r}{2}b_a)$,
4. $\left(\frac{b}{C}\right)\min_{a \in A}(c_a + Ca_a) + \min_{a \in A}(c_a + \frac{r}{2}a_a + \frac{2C-r}{2}b_a)$.

In these expressions $r = b \mod C$, unless $b$ is a multiple of $C$, in which case $r = C$.

**Proof.** Consider first the case $r = C$. Since the problem is bounded, it has an optimal extreme point. Since $r = C$, a $C$-fractional flow $(f^+_a + f^-_a \neq mC$ for some integer $m$) on one edge would require fractional flow on another edge or parallel flow on another edge, contradicting either Lemma 20 or Lemma 21. So in this case an extreme point
has no fractional flow. Also, an extreme point can have at most one edge, say \( e \), with parallel flow, and \( f^+_e - f^-_e = mC \) for some integer \( m \) since, otherwise, flow balance would require either an edge with fractional flow or another edge with parallel flow. Since the problem is bounded, we can reduce the flow around the forward-reverse cycle of this edge until its flow is unidirectional without increasing the cost. So, the problem has an optimal solution with exclusively unidirectional flows in multiples of the capacity \( C \). Since the problem is bounded, removal of \( C \) units of flow from both a forward flow and a reverse flow does not increase the objective value. Therefore, some optimal solution has only forward flows. The result for \( r = C \) follows.

Consider next the case \( r < C \). Again, since the problem is bounded, it has an optimal extreme point. Since \( r < C \), a feasible solution requires either \( C \)-fractional or parallel flow, and, by the prior lemmas, an extreme point solution will have either exactly one edge with fractional flow or exactly one edge with parallel flow. Flow on all other edges will be unidirectional and in multiples of the capacity \( C \). Furthermore, we can assume that no edge will carry reverse flow of \( C \) or more units. Such a flow would require forward flow of \( C \) or more on some edge since only one edge has with \( C \)-fractional or parallel flow, and boundedness implies that we can remove \( C \) units of flow and the corresponding capacity from such a cycle without increasing cost.

So, we can restrict edges with fractional flow to have either forward flow of \( mC + r \) units for some integer \( m \) or reverse flow of \( C - r \) units. Consequently, no extreme point solution with \( C \)-fractional flow has cost cheaper than the minimum of the costs of solutions (1) and (2).

Similarly, we can restrict edges with parallel flow to have either forward flow of \( mC + \frac{C - r}{2} \) and reverse flow of \( \frac{C - r}{2} \) (i.e., net forward flow of \( r \)) or forward flow of \( mC + \frac{1}{2} \) and reverse flow of \( \frac{2C - r}{2} \), (i.e., net reverse flow of \( C - r \)). Therefore, no extreme point solution with parallel flow has cost cheaper than the minimum of costs (3) and (4).

4.6 Solution Approach and Computational Study

Section 4.6.1 describes our branch and bound solution procedure for the full formulation and a cutting plane procedure to obtain lower and upper bounds from the cut formulation. We have not implemented a solution procedure for the nonlinear flow formulation. Section 4.6.2 summarizes the results of our initial computational study of the SCNLP.
4.6.1 Solution Approach

Full Formulation

To solve the full formulation of the SCNLP, we again constructed around the CPLEX solver a cutting plane and branch and bound solution approach that first solves the LP relaxation and then tightens the lower bound by adding valid inequalities. We add cutset, residual capacity, three partition, and odd flow inequalities, in that order.

We again employ the heuristic procedure of searching cut set inequalities that appends all singleton inequalities and then iteratively determines a Gomory-Hu tree to identify additional cuts. Since each edge has only one residual capacity inequality, we add them all. We search three partition inequalities with the enumeration approach used for the CNLP computational study as described in Section 3.8. Finally, the number of odd flow inequalities is at most the number of nodes, so we add them all. The cutset and three partition modules iteratively search for cuts and resolve the LP until the lower bound improvement falls below a threshold, currently set to 0.01%.

We also tested 0.1% and 0.001%. The results seem fairly insensitive to the threshold value, at least in this range, though 0.01% seems slightly better.

To obtain a feasible solution and upper bound, we again use the two rounding heuristics described in Section 3.8 for the CNLP. The first rounds up each design variable of the initial LP relaxation to obtain a solution, then repeats for each extended LP relaxation defined at the end of a stage of cutting plane addition, and retains the best. The second iteratively increases a design variable lower bound and resolves the final extended LP relaxation. We run both heuristics and retain the
better solution.

**Cut Formulation**

We obtain upper and lower bounds from the cut formulation. To obtain the lower bound, we first add singleton cutset inequalities to the unconstrained problem, and then search for violated cutset inequalities. We iteratively solve the separation exactly as the max flow problem equivalent to checking feasibility of a capacitated network flow problem. We then round to its ceiling the RHS of any identified violation, and add the cut, until the lower bound fails to improve by 0.1%. Note that separation via the max flow problem does not apply to the full formulation. Since this formulation constructs a feasible (fractional) flow, it violates no unrounded cutset inequality.

To achieve an upper bound with the cut formulation, we use a two stage variation of the iterated edge rounding heuristic of Section 3.8. The first stage iteratively solves the separation problem until the lower bound improvement fails to achieve the threshold and then rounds up the lower bound of the loading value nearest its ceiling to that ceiling, until achieving an integer solution. The second stage repeats this process with a threshold of 0, that is, every solution output by the separation module is feasible to the LP relaxation of the problem. The integer loading obtained at the end of the second stage is feasible to the IP and so provides a valid upper bound. (The solution output by the first stage might not satisfy all cut inequalities. In one of our test instances the value output at the end of the first stage was lower than the optimal value.) The second stage can generate very large problems since it separates and adds cuts until the solution satisfies all the cut conditions. With an improvement threshold of 0.1%, all our test instances terminate within seconds, it seems by chance.
At 0.01% and 1.0%, one instance seems to become stuck at the second stage.

With these bounding procedures, we can, in theory, solve the cut formulation to optimality with a problem specific branch and bound procedure. However, the task of coding such a procedure seemed daunting. Bienstock et al. [12] tailored branch and cut to a cut formulation of the multicommodity NLP based on the rounded metric inequalities of Onaga and Kakusho [38]. They obtained lower bounds through heuristic separation and upper bounds through a procedure very similar to our iterative rounding heuristic.

4.6.2 Computational Study

To study the computational difficulty of the SCNLP, we randomly generated test instances of 8, 10, 12, 15, 20, and 25 nodes. Each graph is complete. We selected node locations uniformly over a square, and edge lengths calculated with as Euclidean distances. Each of the first \( n-1 \) nodes was equally likely to be a supply or demand node, with demand magnitude chosen uniformly on \([0, 10]\). The final demand balances the others. For comparison, we defined an NLP instance from each SCNLP instance of 8, 10, 12, and 15 nodes, on the same graphs, with the same origin and destination nodes, and with the same total demand.

Tables 4.1, 4.2, and 4.3 summarize the results. We have scaled the solution values so that the optimal solution (or best solution found for the starred (*) instances that we did not provably solve) has value 1.00. Table 4.1 summarizes the results for the full formulation. Column 2 indicates the value of the raw, uncut LP relaxation,
and column 3 indicates the value of the final extended formulation, that is, the LP relaxation after addition of valid inequalities. Column 4 shows the value of the heuristically determined initial branch and bound solution, and column 5 shows the time spent both generating the cutting planes and solving the heuristics. Columns 6 and 7 show, respectively, the branch and bound solution time and number of nodes.

Table 4.2 summarizes results for the cut formulation. Columns 2 and 3 indicate the lower and upper bounds, and column 4 shows elapsed time. Again, the computations with the full formulation did not provably solve the starred (*) problems. In fact, the cut formulation heuristic procedure found a better solution on test 25b than our full length branch and bound search, even though the full formulation violates no unrounded cutset inequalities.

Table 4.3 summarizes results of the NLP tests. The column labels have the same meanings as those of Table 4.1, and starred tests were not solved to optimality.

Columns 6 and 7 of Table 4.1 indicate that we have some difficulty provably solving (within 125,000 branch and bound nodes) SCNLP instances of 25 nodes. We tried 30 node instances as well, and were able to solve only one of five instances. Columns 5 and 6 of Table 4.3 indicate that the NLP is computationally burdensome relative to the SCNLP. Three of the instances hit the 125,000 node limit, and the average branch and bound time and node count for the 15 node instances are 4662 seconds and 68,434, respectively. The corresponding numbers for the SCNLP instances are 19.1 seconds and 2698.4 nodes.

The tests seem to indicate that the SCNLP is computationally easier than the NLP. The heuristics for the full formulation of the SCNLP and the NLP exhibit
<table>
<thead>
<tr>
<th>test</th>
<th>raw LP</th>
<th>LP bnd</th>
<th>full heur</th>
<th>pre time</th>
<th>BB time</th>
<th>nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>8a</td>
<td>0.65</td>
<td>0.86</td>
<td>1.07</td>
<td>0.1</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>8b</td>
<td>0.65</td>
<td>0.86</td>
<td>1.16</td>
<td>0.1</td>
<td>0.2</td>
<td>29</td>
</tr>
<tr>
<td>8c</td>
<td>0.77</td>
<td>1.00</td>
<td>1.20</td>
<td>0.1</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>8d</td>
<td>0.89</td>
<td>0.96</td>
<td>1.00</td>
<td>0.1</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>8e</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>avg</td>
<td>0.79</td>
<td>0.94</td>
<td>1.09</td>
<td>0.1</td>
<td>0.1</td>
<td>5.8</td>
</tr>
<tr>
<td>10a</td>
<td>0.10</td>
<td>0.99</td>
<td>1.01</td>
<td>0.3</td>
<td>0.1</td>
<td>5</td>
</tr>
<tr>
<td>10b</td>
<td>0.66</td>
<td>1.00</td>
<td>1.00</td>
<td>0.3</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>10c</td>
<td>0.84</td>
<td>0.93</td>
<td>1.02</td>
<td>0.3</td>
<td>0.6</td>
<td>132</td>
</tr>
<tr>
<td>10d</td>
<td>0.55</td>
<td>0.82</td>
<td>1.14</td>
<td>0.3</td>
<td>1.7</td>
<td>518</td>
</tr>
<tr>
<td>10e</td>
<td>0.74</td>
<td>0.91</td>
<td>1.11</td>
<td>0.4</td>
<td>1.1</td>
<td>87</td>
</tr>
<tr>
<td>avg</td>
<td>0.75</td>
<td>0.93</td>
<td>1.06</td>
<td>0.3</td>
<td>0.7</td>
<td>148.4</td>
</tr>
<tr>
<td>12a</td>
<td>0.72</td>
<td>0.90</td>
<td>1.04</td>
<td>1.1</td>
<td>2.3</td>
<td>193</td>
</tr>
<tr>
<td>12b</td>
<td>0.72</td>
<td>0.85</td>
<td>1.05</td>
<td>1.1</td>
<td>10.1</td>
<td>3158</td>
</tr>
<tr>
<td>12c</td>
<td>0.52</td>
<td>0.79</td>
<td>1.04</td>
<td>2.1</td>
<td>6.7</td>
<td>1437</td>
</tr>
<tr>
<td>12d</td>
<td>0.72</td>
<td>0.86</td>
<td>1.01</td>
<td>1.1</td>
<td>3.7</td>
<td>470</td>
</tr>
<tr>
<td>12e</td>
<td>0.91</td>
<td>0.97</td>
<td>1.00</td>
<td>1.3</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>avg</td>
<td>0.72</td>
<td>0.88</td>
<td>1.03</td>
<td>1.3</td>
<td>4.6</td>
<td>1051.6</td>
</tr>
<tr>
<td>15a</td>
<td>0.55</td>
<td>0.85</td>
<td>1.02</td>
<td>6.1</td>
<td>9.6</td>
<td>1660</td>
</tr>
<tr>
<td>15b</td>
<td>0.57</td>
<td>0.86</td>
<td>1.09</td>
<td>0.3</td>
<td>10.4</td>
<td>971</td>
</tr>
<tr>
<td>15c</td>
<td>0.84</td>
<td>0.93</td>
<td>1.05</td>
<td>0.3</td>
<td>2.9</td>
<td>134</td>
</tr>
<tr>
<td>15d</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>5.8</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>15e</td>
<td>0.59</td>
<td>0.81</td>
<td>1.08</td>
<td>0.3</td>
<td>72.7</td>
<td>10,727</td>
</tr>
<tr>
<td>avg</td>
<td>0.71</td>
<td>0.89</td>
<td>1.05</td>
<td>2.6</td>
<td>19.1</td>
<td>2698.4</td>
</tr>
<tr>
<td>20a*</td>
<td>0.58</td>
<td>0.78</td>
<td>1.03</td>
<td>68.0</td>
<td>1149.6</td>
<td>125,000</td>
</tr>
<tr>
<td>20b</td>
<td>0.71</td>
<td>0.87</td>
<td>1.13</td>
<td>68.0</td>
<td>124.4</td>
<td>6181</td>
</tr>
<tr>
<td>20c</td>
<td>0.92</td>
<td>0.96</td>
<td>1.04</td>
<td>119.9</td>
<td>48.7</td>
<td>7969</td>
</tr>
<tr>
<td>20d</td>
<td>1.0</td>
<td>1.0</td>
<td>1.00</td>
<td>58.1</td>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>20e</td>
<td>0.94</td>
<td>0.98</td>
<td>1.08</td>
<td>58.7</td>
<td>4.5</td>
<td>525</td>
</tr>
<tr>
<td>avg</td>
<td>0.83</td>
<td>0.92</td>
<td>1.06</td>
<td>73.5</td>
<td>265.4</td>
<td>27,935</td>
</tr>
<tr>
<td>25a</td>
<td>0.87</td>
<td>0.94</td>
<td>1.06</td>
<td>38.6</td>
<td>162.4</td>
<td>3027</td>
</tr>
<tr>
<td>25b*</td>
<td>0.47</td>
<td>0.70</td>
<td>1.01</td>
<td>1.4</td>
<td>3711.6</td>
<td>125,000</td>
</tr>
<tr>
<td>25c</td>
<td>0.85</td>
<td>0.94</td>
<td>1.05</td>
<td>354.1</td>
<td>114.3</td>
<td>6953</td>
</tr>
<tr>
<td>25d*</td>
<td>0.79</td>
<td>0.90</td>
<td>1.04</td>
<td>388.6</td>
<td>5893.1</td>
<td>125,000</td>
</tr>
<tr>
<td>25e</td>
<td>0.68</td>
<td>0.85</td>
<td>1.16</td>
<td>1.6</td>
<td>685.4</td>
<td>12,131</td>
</tr>
<tr>
<td>avg</td>
<td>0.73</td>
<td>0.87</td>
<td>1.07</td>
<td>156.9</td>
<td>2113.4</td>
<td>54,422.2</td>
</tr>
</tbody>
</table>

135
Table 4.2: Computational results, cut formulation of SCNLP

<table>
<thead>
<tr>
<th>test</th>
<th>cut LR</th>
<th>cut heur</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>8a</td>
<td>0.88</td>
<td>1.00</td>
<td>0.02</td>
</tr>
<tr>
<td>8b</td>
<td>0.90</td>
<td>1.01</td>
<td>0.03</td>
</tr>
<tr>
<td>8c</td>
<td>1.00</td>
<td>1.20</td>
<td>0.01</td>
</tr>
<tr>
<td>8d</td>
<td>0.96</td>
<td>1.00</td>
<td>0.02</td>
</tr>
<tr>
<td>8e</td>
<td>1.00</td>
<td>1.00</td>
<td>0.01</td>
</tr>
<tr>
<td>avg</td>
<td>0.95</td>
<td>1.04</td>
<td>0.02</td>
</tr>
<tr>
<td>10a</td>
<td>0.99</td>
<td>1.00</td>
<td>0.03</td>
</tr>
<tr>
<td>10b</td>
<td>1.00</td>
<td>1.00</td>
<td>0.02</td>
</tr>
<tr>
<td>10c</td>
<td>0.98</td>
<td>1.13</td>
<td>0.03</td>
</tr>
<tr>
<td>10d</td>
<td>0.88</td>
<td>1.02</td>
<td>0.05</td>
</tr>
<tr>
<td>10e</td>
<td>0.92</td>
<td>1.05</td>
<td>0.05</td>
</tr>
<tr>
<td>avg</td>
<td>0.95</td>
<td>1.04</td>
<td>0.04</td>
</tr>
<tr>
<td>12a</td>
<td>0.82</td>
<td>1.16</td>
<td>0.08</td>
</tr>
<tr>
<td>12b</td>
<td>0.90</td>
<td>1.07</td>
<td>0.09</td>
</tr>
<tr>
<td>12c</td>
<td>0.82</td>
<td>1.05</td>
<td>0.05</td>
</tr>
<tr>
<td>12d</td>
<td>0.61</td>
<td>1.05</td>
<td>0.07</td>
</tr>
<tr>
<td>12e</td>
<td>0.98</td>
<td>1.16</td>
<td>0.05</td>
</tr>
<tr>
<td>avg</td>
<td>0.82</td>
<td>1.10</td>
<td>0.07</td>
</tr>
<tr>
<td>15a</td>
<td>0.81</td>
<td>1.03</td>
<td>0.12</td>
</tr>
<tr>
<td>15b</td>
<td>0.84</td>
<td>1.07</td>
<td>0.10</td>
</tr>
<tr>
<td>15c</td>
<td>0.97</td>
<td>1.07</td>
<td>0.10</td>
</tr>
<tr>
<td>15d</td>
<td>1.00</td>
<td>1.00</td>
<td>0.04</td>
</tr>
<tr>
<td>15e</td>
<td>0.76</td>
<td>1.14</td>
<td>0.17</td>
</tr>
<tr>
<td>avg</td>
<td>0.88</td>
<td>1.06</td>
<td>0.11</td>
</tr>
<tr>
<td>20a*</td>
<td>0.77</td>
<td>1.13</td>
<td>0.33</td>
</tr>
<tr>
<td>20b</td>
<td>0.90</td>
<td>1.15</td>
<td>0.38</td>
</tr>
<tr>
<td>20c</td>
<td>0.95</td>
<td>1.04</td>
<td>0.28</td>
</tr>
<tr>
<td>20d</td>
<td>1.00</td>
<td>1.00</td>
<td>0.13</td>
</tr>
<tr>
<td>20e</td>
<td>0.97</td>
<td>1.02</td>
<td>0.33</td>
</tr>
<tr>
<td>avg</td>
<td>0.92</td>
<td>1.07</td>
<td>0.29</td>
</tr>
<tr>
<td>25a</td>
<td>0.88</td>
<td>1.06</td>
<td>0.92</td>
</tr>
<tr>
<td>25b*</td>
<td>0.64</td>
<td>0.99</td>
<td>0.55</td>
</tr>
<tr>
<td>25c</td>
<td>0.93</td>
<td>1.06</td>
<td>0.89</td>
</tr>
<tr>
<td>25d*</td>
<td>0.88</td>
<td>1.10</td>
<td>0.68</td>
</tr>
<tr>
<td>25e</td>
<td>0.85</td>
<td>1.05</td>
<td>0.80</td>
</tr>
<tr>
<td>avg</td>
<td>0.83</td>
<td>1.05</td>
<td>0.77</td>
</tr>
<tr>
<td>test</td>
<td>raw LP</td>
<td>LP bnd</td>
<td>heur</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>8a</td>
<td>0.78</td>
<td>0.98</td>
<td>1.14</td>
</tr>
<tr>
<td>8b</td>
<td>0.71</td>
<td>0.99</td>
<td>1.00</td>
</tr>
<tr>
<td>8c</td>
<td>0.73</td>
<td>0.93</td>
<td>1.09</td>
</tr>
<tr>
<td>8d</td>
<td>0.96</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>8e</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>avg</td>
<td>0.84</td>
<td>0.98</td>
<td>1.05</td>
</tr>
<tr>
<td>10a</td>
<td>0.95</td>
<td>0.98</td>
<td>1.06</td>
</tr>
<tr>
<td>10b</td>
<td>0.70</td>
<td>0.96</td>
<td>1.00</td>
</tr>
<tr>
<td>10c</td>
<td>0.84</td>
<td>0.93</td>
<td>1.10</td>
</tr>
<tr>
<td>10d</td>
<td>0.69</td>
<td>0.95</td>
<td>1.07</td>
</tr>
<tr>
<td>10e</td>
<td>0.79</td>
<td>0.98</td>
<td>1.01</td>
</tr>
<tr>
<td>avg</td>
<td>0.80</td>
<td>0.96</td>
<td>1.05</td>
</tr>
<tr>
<td>12a</td>
<td>0.77</td>
<td>0.92</td>
<td>1.09</td>
</tr>
<tr>
<td>12b</td>
<td>0.77</td>
<td>0.93</td>
<td>1.00</td>
</tr>
<tr>
<td>12c*</td>
<td>0.74</td>
<td>0.84</td>
<td>1.05</td>
</tr>
<tr>
<td>12d</td>
<td>0.74</td>
<td>0.92</td>
<td>1.17</td>
</tr>
<tr>
<td>12e</td>
<td>0.91</td>
<td>0.95</td>
<td>1.03</td>
</tr>
<tr>
<td>avg</td>
<td>0.79</td>
<td>0.91</td>
<td>1.07</td>
</tr>
<tr>
<td>15a*</td>
<td>0.71</td>
<td>0.84</td>
<td>1.09</td>
</tr>
<tr>
<td>15b*</td>
<td>0.80</td>
<td>0.91</td>
<td>1.02</td>
</tr>
<tr>
<td>15c</td>
<td>0.90</td>
<td>0.96</td>
<td>1.00</td>
</tr>
<tr>
<td>15d</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>15e</td>
<td>0.66</td>
<td>0.86</td>
<td>1.06</td>
</tr>
<tr>
<td>avg</td>
<td>0.81</td>
<td>0.91</td>
<td>1.03</td>
</tr>
</tbody>
</table>
similar errors, and neither performance seems to degrade with graph size. Both the final and raw NLP LP formulation are tighter than the SCNLP formulations. Indeed, since our SCNLP inequalities are variations of NLP inequalities, we likely do not understand the SCNLP polyhedral structure as well as that of the NLP. Nevertheless, the SCNLP required not only less solution time, but also smaller branch and bound trees.

The tests leave inconclusive the relative merit of the SCNLP cut formulation. The bounds provided by the cut formulation seem comparable to those achieved by the cutting plane and heuristic procedures of the full formulation. The solution procedure for the cut formulation seems quicker than the pre-branch and bound stages of the procedure for the full formulation, though neither seems burdensome relative to branch and bound. As indicated earlier, the two stage cut formulation upper bounding procedure involves an improvement threshold parameter, and the performance of a particular setting seems instance dependent. The tabulated solution times do not reflect tuning times. Also, the cut procedure, as coded, consumes system memory on some 30 node instances. The full formulation achieves upper and lower bounds for the 30 node instances, though struggles to provably solve them with branch and bound. Finally, of course, without a procedure that solves the cut formulation to optimality, our tests can offer only a restricted comparison of the formulations' relative computational difficulty.
Chapter 5

Contribution Summary and Research Questions

We conclude by summarizing our contributions and posing several unanswered questions from our research agenda. We have considered three variations of the Network Loading Problem (NLP), and emphasized throughout the thesis valid inequalities and their computational impact on a branch and bound approach for solving the NLP.

Chapter 2 formulates the logistics problem of pup matching as a special case of the NLP on a directed graph. We examined four heuristic methods and a cutting plane based branch and bound procedure for solving the Pup Matching Problem. Among the more realistic test problems that we solved to optimality, the heuristics performed very well, obtaining solutions with objective values within 1.3% of optimal. To what extent we are witnessing a selection bias (that is, whether the heuristics were more effective for problems we have been able to solve) remains to be seen.

Despite the apparent practical success of the heuristics, we would consider an ap-
proximation algorithm with a bound less than 2 a significant addition to this research.
The Matching Approximation provides a 2-approximation for Pup Matching, each of
the three shortest path based heuristic provides a 2-approximation for the incomplete
NLP formulation, and examples show that the ratio of 2 is tight for all four. An ap-
proximation ratio of 2 is often readily achieved and seems especially natural for this
problem since it coincides with the towing capacity of each cab. A tighter algorithm
would likely reflect new insight.

Even though the heuristic methods were able to generate good feasible solutions,
because of weak linear programming lower bounds, a default implementation of branch
and bound was not able solve problems to optimality within reasonable running times.
Consequently, as in other application settings, our computational study underscores
the importance of high quality lower bounds to provably solve integer programs. To
this end, the odd flow inequalities have proved very effective. They permitted us to
solve in seconds city blocks problems that we were previously unable to solve with
days of computation time. Although these cuts are a special case of the generalized
cutset inequalities that Chopra, Gilboa, and Sastry [17] described for a single origin-
destination pair NLP variation, we believe our parity interpretation of validity and
facet definition result are new. The concept of odd flow inequalities generalizes for a
single facility Network Loading Problem on a directed graph with arbitrary facility
capacities $C$, instead of 2, to exploit the observation that the loading on any arc
whose total flow is not a multiple of $C$ requires spare capacity. We suspect that cuts
based on similar parity arguments would help solve other network design problems
and consider the discovery of such cuts a primary remaining initiative of the Pup

140
Matching research.

Chapter 3 generalizes Network Loading to allow for facilities of compartmentalized capacity. We consider the main contribution of this research the extension of NLP polyhedral results to the compartmentalized models. In particular, we show that residual capacity inequalities define the convex hull of a segregated compartments version of the Single Arc Design Problem (SADP) introduced by Magnanti, Mirchandani, and Vachani [34], and we employ the procedure of Atamtürk and Rajan [5] to efficiently separate residual capacity inequalities for all three compartmentalized models.

We have been most successful extending the polyhedral results to the segregated model, and would consider similar development of the nested and general models a significant contribution to the research. In particular, we have not been able to extend the SADP integrality result beyond the segregated model and are interested whether residual capacity inequalities define the convex hull of SADP variations of the latter two compartmentalized models.

Four series of computational tests seem to support the intuition that the compartmentalized problems are easiest to solve when the data make the LP relaxation inherently tight, for example, when the capacity values are small. The tests also reveal relatively large integrality gaps for the general model, suggesting the potential for developing additional cut families. The primary aim of this research has been the extension of NLP results, and the cuts that we apply to the compartmentalized models are extensions of cuts valid for the NLP. Additional problem insight and computational efficiency would likely result from inequalities specific to the compartmentalized
model that reflect, perhaps, yet to be discovered structural relationships between the commodities and the compartments.

Chapter 4 considers a variation of the NLP we refer to as Single Commodity Pup Matching and that has apparently received very little attention in the research literature. We develop three formulations of the problem and show that a fourth, extended variable formulation based on the network flow character of the problem, is no tighter than the first formulation. Tests indicate that the SCNLP is computationally less burdensome than the NLP, though relatively large integrality gaps suggest that further research might yield computationally useful additional families of valid inequalities.

We have derived one such family of new cuts from the single cut SCNLP that we solved in Section 4.5. Atamtürk [4] showed that the generalized cutset inequalities of Chopra, Gilboa, and Sastry [17] yield the convex hull of the single cut problem for directed graphs. We have shown that the similar set of inequalities,

$$f^+(S) - f^-(S) + Cy(S) + r(y(A\backslash S) - y(S)) \geq r \mu, \quad (5.1)$$

is valid for the undirected version of the problem considered in Section 4.5. In this expression, edges $A$ define the cut in question, $f^+(S)$ is the sum of flows crossing the cut in one direction on edges $S \subseteq A$, $f^-(S)$ is the sum of flows crossing the cut in the other direction on edges $S$, and $C$ and $r$ are the capacity and residual demand, respectively. We think that our optimality result for this problem can be leveraged to show that this or a related family of inequalities yields the convex hull of the single undirected cut problem.
However, these generalized cutset inequalities, like the other cuts we apply to SCNLP, were originally derived in terms of the NLP. In particular, we have not derived any cuts specific to the SCNLP that reflect its different structure. As for the compartmentalized problems, new cuts would likely entail new understanding of the problem and also improve computational efficiency.

We named the limiting case of the SCNLP with capacity sufficient to accommodate all flow the Balanced Demand Steiner Forest Problem (BDSF), and cast the problem within the framework of Goemans and Williamson [23] to achieve a 2-approximation. Our formulation of the BDSF as a constrained forest problem includes the an exponential set of inequalities,

$$\sum_{e \in \delta(S)} z_e \geq g(S), \forall S \subset N,$$

with the right-hand side function $g$ defined as

$$g(S) = \begin{cases} 
1, & \text{if } \sum_{i \in S} d_i \neq 0 \\
0, & \text{if } \sum_{i \in S} d_i = 0,
\end{cases}$$

where $d_i$ is the net demand at node $i$. An analogous cut set applies to the Steiner Tree Problem, another constrained forest problem, and involves the right-hand side function

$$f(S) = \begin{cases} 
1, & \text{if } S \cap T \neq \emptyset, T \\
0, & \text{otherwise}.
\end{cases}$$

In this expression, $T$ is the set of nodes required to be connected in the Steiner Forest Problem. Separation for this problem can be solved as a set of min cut problems. Despite the similarity between $f$ and $g$, we have not efficiently solved the separation
problem defined by \( g \). Finding a polynomial separation for the BDSF is an alluring research challenge.

Mansour and Peleg [36] generalized their spanner based NLP approximation algorithm from the special case when the demand structure requires full connectivity of the underlying graph by exploiting a constant factor approximation for the generalized Steiner Forest Problem (that of finding the cheapest forest that connects a specified set of node pairs), which is equivalent to an NLP with sufficiently large capacity. We could apply this generalization to the SCNLP if we can establish a constant gap between the SCNLP and the NLP defined by pairing nodes according to the routings of the SCNLP LP relaxation. Corollary 15 would then imply that we have achieved, within a constant factor, the tightest possible SCNLP approximation unless \( P = NP \).

Finally, we note that our computational tests on the SCNLP cutset formulation of Section 4.1.2 examine only the quality of values provided by upper and lower bounding procedures. Solving the SCNLP to optimality through a problem specific branch and bound procedure constructed around these bounding procedures would permit a more thorough investigation similar to that Bienstock et al. [12] performed on the NLP. They concluded that the computational difficulty of their cutset formulation based on the metric inequalities is comparable to that of the corresponding full flow formulation. Metric inequalities are more complex than cutset inequalities, yet they offer greater simplification of the relevant LP relaxations by permitting elimination of multicommodity flow variables. Consequently, the computational burden of the SCNLP cutset formulation relative to that of its full formulation is unclear.
Although developing an SCNLP specific branch and bound routine might entail a significant coding burden, it seems necessary to fairly assessing the potential of the cutset formulation, and, more generally, the computational difficulty of the SCNLP.
Bibliography


148


Appendix A

Supplement to Pup Matching

This appendix contains two relatively long \( \mathcal{NP} \)-completeness proofs related to Pup Matching that we refer to in Chapter 2. We first show, by transformation of Three Dimensional Matching, that a decision problem* version of Pup Matching is \( \mathcal{NP} \)-complete. Section A.2 proves, by transformation of Satisfiability, \( \mathcal{NP} \)-completeness of determining whether a solution to the Pup Matching NLP formulation is free of waiting ring infeasibilities.

A.1 \( \mathcal{NP} \)-completeness of Pup Matching

Pup Matching as a decision problem poses the Yes/No question of whether there exists a feasible loading with cost no greater than a parameter \( C \). We shall refer to this problem as PMD. Three Dimensional Matching (3DM) was proven \( \mathcal{NP} \)-complete by Karp [29] and can be stated as follows (see Garey and Johnson [21]):

**Three Dimensional Matching (3DM)**

**Instance:** A set \( M \subseteq W \times X \times Y \), defined by disjoint sets \( W, X, \) and \( Y \) each of cardinality \( q \).

**Problem:** Does \( M \) contain a matching of cardinality \( q \), that is, a subset \( M' \subseteq M \) with \( |M'| = q \) and satisfying the property that no two elements of \( M' \) share any coordinate?
First, \( \text{PMD} \in \mathcal{NP} \), because a routing of loading cost no greater than \( C \) with a dispatching sequence to prove its feasibility provides a succinct certificate to any Yes instance.

Next, we show how to transform 3DM to PMD. Given an instance of 3DM, define the following instance of PMD:

1. A node set of cardinality \( 1 + 5q + |M| \) labeled as:
   
   (a) a central node labeled \( I \),
   (b) 3 nodes for each of the \( q \) components \( i \) of \( W \), labeled \( W_i, W_{ia}, \) and \( W_{ib} \),
   (c) 1 node for each of the \( q \) components \( i \) of \( X \), labeled \( X_i \),
   (d) 1 node for each of the \( q \) components \( i \) of \( Y \), labeled \( Y_i \),
   (e) 1 node for each component \( i, j, k \) of \( M \), labeled \( M_{ijk} \).

2. An arc set of cardinality \( 4q + 3|M| \) with connectivity as follows:
   
   (a) an arc of cost 1 from \( I \) to each of \( W_{ia}, i = 1, 2 \ldots q \),
   (b) an arc of cost 1 from \( I \) to each of \( W_{ib}, i = 1, 2 \ldots q \),
   (c) an arc of cost 0 from \( W_{ia} \) to \( W_i, i = 1, 2 \ldots q \),
   (d) an arc of cost 0 from \( W_{ib} \) to \( W_i, i = 1, 2 \ldots q \),
   (e) an arc of cost 1 from \( W \) to each of the nodes corresponding to components of \( M \) including \( W_i, i = 1, 2 \ldots q \),
   (f) two arcs each of cost 0 from each \( M_{ijk} \), one to \( X_j \) and one to \( Y_k, \forall i, j, k \in M \).

3. \( 4q \) pups, 2 corresponding to each component of \( W \), 1 to each component of \( X \), and 1 to each component of \( Y \), all originating at node \( I \) and with destinations:
   
   (a) 1 pup to \( W_{ia}, i = 1, 2 \ldots q \),
   (b) 1 pup to \( W_{ib}, i = 1, 2 \ldots q \),
   (c) 1 pup to \( X_i, i = 1, 2 \ldots q \),
   (d) 1 pup to \( Y_i, i = 1, 2 \ldots q \).

4. Cost parameter \( C = 3q \).

Figure A.1 shows the graph corresponding to the following 3D array \( M \) for \( q = 3 \):
Figure A.1: Pup Matching Graph to solve the Three Dimensional Matching instance illustrating the 3DM-PMD transformation. Nonzero arc costs are shown.

$3q$ is a lower bound on the pup matching loading cost because at least one cab must travel from node $I$ to each of $W_{ia}, i = 1, 2 \ldots q$ and to each of $W_{ib}, i = 1, 2 \ldots q$ to deliver the first $2q$ pups ((a) and (b) in the pup section of the instance definition),
and $2q$ pups must travel from some node $W_i$ to some node $M_{ijk}$, requiring at least $q$ cabs.

Suppose some matching on $M$ has components given by $(w_i, x_i, y_i)$ for $i = 1, 2 \ldots q$. Then there exists a routing of cost $3q$. 1 pup must be routed to each of nodes $W_{ia}$ and $W_{ib}$, $i = 1, 2 \ldots q$, for a combined cost of $2q$. The pup with destination node $X_{x_i}$ can be matched with the pup routed to node $W_{w_{ia}}$ and then sent singly to node $W_{w_i}$. Similarly, the pup with destination node $Y_{y_i}$ can be matched with the pup routed to node $W_{w_{ib}}$ and then sent singly to node $W_{w_i}$. These two pups can then be sent matched from node $W_{w_i}$ to node $M_{w_{ix_iy_i}}$ for a cost of 1, and finally sent singly to $X_{x_i}$ and $Y_{y_i}$, respectively.

Next, suppose that some routing achieves a cost of $3q$. Since at least $2q$ cabs must leave node $I$ and at least $q$ must leave the nodes $W_i$, a solution of cost $3q$ requires that each cab leaving $I$ carry two pups, one of which terminates after one arc at node $W_{ia}$ or $W_{ib}$. Consequently, 2 pups must arrive at each node $W_i$ and then travel matched to some node $M_{ijk}$. Since the routing is feasible, these pairings, in turn, imply the existence of a matching of cardinality $q$ in $M$.

So, pup matching has a routing of cost $3q$ if and only if 3DM has a matching of cardinality $q$ in $M$, and we can solve an instance of 3DM by solving the corresponding instance of PMD with cost parameter $C = 3q$. Since our transformation from 3DM to PMD is polynomial, Pup Matching as a decision problem is $\mathcal{NP}$-complete.
A.2 \( \mathcal{NP} \)-completeness of the Waiting Ring Problem

We showed in Section 2.3 that a waiting ring might render a feasible solution to the Network Loading formulation of Pup Matching infeasible to Pup Matching itself, and noted the \( \mathcal{NP} \)-completeness of the decision problem of determining whether a Network Loading solution is free of waiting rings. We refer to that decision problem as the Waiting Ring Problem (WR) and state it in instance-problem format in Section 2.3. In this appendix, we prove the \( \mathcal{NP} \)-completeness result by transformation of Satisfiability.

Satisfiability (see Garey and Johnson [21]) involves a set \( U = \{ u_1, u_2, \ldots, u_m \} \) of Boolean variables and a set \( C = \{ C_1, C_2, \ldots, C_n \} \) of clauses. Each clause is a disjunction over a subset of the variables \( U \) or their negations, and a truth assignment for \( U \) sets each component variable true or false. SAT was proven \( \mathcal{NP} \)-complete by Cook [18], and can be stated as follows:

**Satisfiability (SAT)**
- **Instance:** A set \( U \) of variables and a collection \( C \) of clauses over \( U \).
- **Problem:** Is there a satisfying truth assignment for \( C \)?

First, WR \( \in \mathcal{NP} \) because a loading utilization sequence and identification of the tokens advanced by each loaded unit provide a succinct certificate for any Yes instance.

We next describe three gimmicks used in the transformation of SAT to WR. The first, depicted in Figure A.2, is used to control the flow of four tokens corresponding to a single variable.
Figure A.2: A gimmick controlling token flow to effectively assign a variable either true or false.

Token $T$, corresponds to a variable value of true, enters the gimmick at node 1, and exits at node 2. Similarly, $F$ corresponds to false, and must also travel from node 1 to node 2. Token $P$ is assigned path 1-2-3-4. $R$ must travel from node 3 to 4, then cross a portion of the graph represented by the kinked arc that can be traversed only if all clauses have effectively been made true, and finally exit at node 2. Capacity loadings are indicated on the arcs. Since only one unit is loaded on arc 3-4, tokens $R$ and $P$ must travel together on this arc. In turn, $P$ must travel with either $T$ or $F$ in the first loading utilized on arc 1-2 to avoid a waiting ring involving $P$ and $R$. Either $T$ or $F$ is left to eventually pair with $R$ over this 1-2, and $R$ will not arrive at node 1 until all clauses have been made true. Consequently, the gimmick controls token advancement to effectively assign a variable either true or false.

The truth assignment of a variable makes true a subset of clauses. This simulta-
neous activation of clauses is controlled by the second gimmick, shown in Figure A.3 for the case of an assignment activating clauses $C_1, C_2,$ and $C_3$.

Figure A.3: A gimmick effectively transmitting a variable assignment to those clauses the assignment makes true.

Token $C_1$ corresponds to clause 1. It must advance from node 1 to node 2, and its arrival at node 2 indicates that the clause is true or activated. Similarly, $C_2$ must travel from node 2 to 3, and $C_3$ must travel from node 3 to 4. Token $T$ corresponds to a variable value of true. It enters the gimmick at node 1 from another portion of the graph, as indicated by the dashed line, and exits at node 4. Each of arcs 1-2, 2-3 and 3-4 has one loading. Consequently, $C_1, C_2,$ and $C_3$, must each wait for $T$, and, upon its arrival at node 1, all three can advance.

A disjunctive clause can be activated or made true by any of its constituent variables, and the clause cannot be activated without appropriate assignment of one such variable. Also, the graph structure must not hinder advancement of tokens corresponding to the constituent variables, in particular, tokens attempting to activate a clause that has already been made true. The third gimmick provides such a structure and is shown in Figure A.4 for the case of the clause $C = L_1 \lor L_2' \lor L_3$.

For each potentially activating variable assignment, one token enters the gimmick at node 1 from another portion of the graph, as indicated by the dashed arrow, and exits
Figure A.4: A gimmick to control the potential activation of clause \( C = L_1 \lor L_2 \lor L_3 \) by any one of its three constituent variables.

at node 2. For example, \( T1 \) corresponds to a value of true for the first variable in the clause. \( C \), the token corresponding to the clause itself, enters at node 1 and exits at node 4B. Additional tokens enter the gimmick at each of nodes 3A and 4A, and both exit at node 2. Three units, the number of disjunctions in \( C \), are loaded on arc 1-2, and one on each of arcs 3A-3B and 4A-4B. Since each of these latter loadings must advance two tokens, neither token \( E1 \) nor \( E2 \) can advance without \( C \). Similarly, \( C \) cannot advance to node 2 prior to the arrival of \( T1, F2, \) or \( T3 \) since each loading on 1-2 must carry two tokens. Upon arrival of one of \( T1, F2, \) or \( T3 \), however, \( C \) can exit the gimmick, and both \( E1 \) and \( E2 \) can advance to node 1, and then wait there to accompany the remaining variable tokens, \( T1, F2, \) or \( T3 \), to node 2.

Before combining the gimmicks to transform SAT, we introduce some additional notation. Let \( c_j \) denote the number of variables (or negations) in clause \( j \in C \). Let \( t_i \) be the subset of clauses in \( C \), ordered in the same sequence as they appear in \( C \), made true by assignment of true to variable \( i \in U \). Similarly, let \( f_i \) be the subset of
clauses made true by assignment of false to variable $i$, also ordered as they appear in $C$.

Given an instance of SAT, define the following instance of WR:

1. A node set of:

(a) 4 nodes for each variable $i \in U$, labeled $U_{i,1}, U_{i,2}, U_{i,3}, \text{and } U_{i,4}$;
(b) $3+2(c_j-1)$ nodes for each clause $j \in C$, labeled $C_{j,\alpha}, C_{j,\beta}, C_{j,\omega}, E_{j,1A}, E_{j,1B}, ... E_{j,(c_j-1)A}$, and $E_{j,(c_j-1)B}$;
(c) $|U|(|C| + 1)$ nodes, labeled $UC_{1,1}, UC_{1,2}, ... UC_{1,|U|}; UC_{2,1} ... UC_{2,|U|}; ... UC_{|C|+1,1}, ... UC_{|C|+1,|U|}$.

2. An arc set with the following connectivity and loadings:

(a) 1 arc from $U_{i,1}$ to $U_{i,2}$ with 2 units, for $i \in U$;
(b) 1 arc from $U_{i,2}$ to $U_{i,3}$ with 1 unit, for $i \in U$;
(c) 1 arc from $U_{i,3}$ to $U_{i,4}$ with 1 unit, for $i \in U$;
(d) 1 arc from $U_{i,4}$ to $UC_{1,i}$ with 1 unit, for $i \in U$;
(e) 1 arc from $UC_{j,\alpha}$ to $UC_{j+1,\alpha}$ with 1 unit each, for $j \in C, i \in U$;
(f) 1 arc from $UC_{j+1,\alpha}$ to $UC_{j+1,i}$ with 1 unit each, for $j \in C, i = 1, 2, ... |U|-1$;
(g) 1 arc from $UC_{j+1,|U|}$ to $C_{j,\omega}$ with 1 unit each, for $j \in C$;
(h) 1 arc from $UC_{|C|+1,i}$ to $U_{i,1}$ with 1 unit each, for $i \in U$;
(i) 1 arc from $U_{i,2}$ to $C_{j,\alpha}$ with 2 units each, for $i \in U, j \in C$;
(j) 1 arc from $C_{j,\alpha}$ to $C_{j,\beta}$ with $c_j$ units each, for $j \in C$;
(k) 1 arc from $C_{j,\beta}$ to $E_{j,1A}$ with 1 unit each, for all $j \in C$ such that $c_j > 1$;
(l) 1 arc from $E_{j,kA}$ to $E_{j,kB}$ with 1 unit each, for all $j \in C$ such that $c_j > 1, k = 1, 2, ... c_j - 1$;
(m) 1 arc from $E_{j,kB}$ to $E_{j,(k+1)A}$ with 1 unit each, $j \in C$ such that $c_j > 2, k = 1, 2, ... c_j - 2$;
(n) 1 arc from $E_{j,kB}$ to $C_{j,\alpha}$ with 1 unit each, $j \in C$ such that $c_j > 1, k = 1, 2, ... c_j - 1$;
(o) 1 arc from $E_{j,(c_j-1)B}$ to $UC_{j,1}$ with 1 unit each, $j \in C$ such that $c_j > 1$;
(p) 1 arc from $C_{j,\beta}$ to $UC_{1,1}$ with 1 unit each, $j \in C$ such that $c_j = 1$;
(q) 1 arc from $C_{j,\beta}$ to $C_{l,\alpha}$ with 2$|U|$ units each, $j = 1, 2, ... |C| - 1, l = j + 1, ... |C|$.

3. Tokens with the following paths:

(a) 4 tokens for each variable $i \in U$: 158
i. \( T_1 : U_{i,1} \rightarrow U_{i,2} \rightarrow C_{t(1),a} \rightarrow C_{t(1),b} \rightarrow C_{t(2),a} \rightarrow C_{t(2),b} \cdots C_{t(|t|),a} \rightarrow C_{t(|t|),b} \) (terminates at \( U_{i,2} \) if \( t = \emptyset \));

ii. \( F_i : U_{i,1} \rightarrow U_{i,2} \rightarrow C_{f(1),a} \rightarrow C_{f(1),b} \rightarrow C_{f(2),a} \rightarrow C_{f(2),b} \cdots C_{f(|t|),a} \rightarrow C_{f(|t|),b} \) (terminates at \( U_{i,2} \) if \( f_i = \emptyset \));

iii. \( P_i : U_{i,1} \rightarrow U_{i,2} \rightarrow U_{i,3} \rightarrow U_{i,4} \);

iv. \( R_i : U_{i,3} \rightarrow U_{i,4} \rightarrow UC_{1,i} \rightarrow UC_{2,i} \cdots UC_{|C|+1,i} \rightarrow U_{i,1} \rightarrow U_{i,2} \);

(b) \( c_j \) tokens for each clause \( j \in C \):

i. \( C_j : C_{j,\alpha} \rightarrow C_{j,\beta} \rightarrow E_{j,1,A} \rightarrow E_{j,1,B} \rightarrow E_{j,2,A} \rightarrow E_{j,2,B} \cdots \rightarrow E_{j,2(c_j-1),A} \rightarrow E_{j,2(c_j-1),B} \rightarrow UC_{j,1} \rightarrow UC_{j,1+1} \rightarrow UC_{j,2} \rightarrow UC_{j,2+1} \rightarrow \cdots \rightarrow UC_{j,|U|} \rightarrow C_{j,\omega} \) (if \( c_j = 1 \), the path goes directly from \( C_{j,\beta} \) to \( UC_{j,1} \));

ii. \( e_{j,k} : E_{j,k,A} \rightarrow E_{j,k,B} \rightarrow C_{j,\alpha} \rightarrow C_{j,\beta} \) for \( k = 1, 2, \ldots, c_j - 1 \).

Figure A.5 illustrates the WR problem corresponding to the following three variable,

three clause problem:

\[
\begin{align*}
C_1 & : u_1 \lor u_2 \lor u_3 \\
C_2 & : u_1' \lor u_3 \\
C_3 & : u_2' \lor u_3.
\end{align*}
\]

As in the gimmick diagrams, a thick arrow points to each origin node and from each destination node. Note that \( t_1 = \{C_1\} \), \( f_1 = \{C_2\} \), \( t_2 = \emptyset \), \( f_2 = \{C_1, C_3\} \), \( t_3 = \{C_2\} \), \( f_3 = \{C_1, C_3\} \). For each variable \( i \), nodes \( U_{i,1}, U_{i,2}, U_{i,3} \), and \( U_{i,4} \) function as the nodes of a gimmick \( 1 \), and nodes \( UC_{1,i}, \ldots, UC_{j+1,i} \) assume the role of the kinked structure of gimmick \( 1 \) that can be traversed only if all clauses can be made true. Nodes \( C_{t(1),a}, C_{t(1),b}, \ldots, C_{t(|t|),a}, C_{t(|t|),b} \) function similar to the nodes of a gimmick \( 2 \), as do \( C_{f(1),a}, C_{f(1),b}, \ldots, C_{f(|t|),a}, C_{f(|t|),b} \). Finally, for each clause \( j \), nodes \( C_{j,\alpha}, C_{j,\beta}, E_{j,1,A}, E_{j,1,B}, \ldots, E_{j,(c_j-1),A}, E_{j,(c_j-1),B} \) function as the nodes of a gimmick \( 3 \).

If a truth assignment exists for the transformed SAT instance, each token can reach its destination. In particular, if \( \{u_1, u_2, \ldots, u_{|U|}\} \) is a truth assignment (with \( u_i = T_i \)
Figure A.5: An example of a SAT to WR transformation.
or $F_i$), then advancing $u_i$ and $P_i$ together from $U_{i,1}$ to $U_{i,2}$ and $u_i$ singly from $U_{i,2}$ to $C_{u_{i,1},a_i}$ for all $i$, permits tokens $C_j$ to advance to $UC_{j,1}$, and tokens $E_{j,1}, \ldots, E_{j,c_{j-1}}$ to $C_j$. Also, advancing $P_i$ singly from $U_{i,2}$ to $U_{i,3}$ and paired with $R_i$ from $U_{i,3}$ to $U_{i,4}$ for all variables $i \in U$, permits the tokens $C_j$ to pair with $R_i$ over the nodes $UC_{k,t}$ and reach $C_{j,\omega}$ and $U_{i,1}$, respectively. From $U_{i,1}$, $R_i$ can effectively free token $u_i'$ to advance to its destination.

On the other hand, a waiting ring is inevitable if no truth assignment exists. As explained in the context of the first gimmick, the first load traversing arc $(U_{i,1}, U_{i,2})$ must carry $T_i$ or $F_i$, as well as $P_i$, to avoid a waiting ring involving $P_i$ and $R_i$. However, if there is no truth assignment, no combination of such loads on $(U_{i,1}, U_{i,2})$ will permit all tokens $C_j$ to advance to $UC_{j,1}$. A waiting ring results among $R_i, C_j$, and $u_i$, where $u_i$ is a variable in the disjunction corresponding to one of the $C_j$ that cannot advance to $UC_{j,1}$.

So, a truth assignment to the SAT instance exists if and only if the WR instance contains no waiting ring. Finally, since the SAT - WR transformation is polynomial, the result follows.
Appendix B

Supplement to Compartmentalized Network Loading

This appendix contains results related to Compartmentalized Network Loading that we refer to in Chapter 3, but deem nonessential to the main text of the thesis. Section B.1 formalizes the transformation depicted in Figure 3.6 of a Single Arc Design Problem (SADP) for the segregated compartments model to the standard network loading SADP introduced by Magnanti, Mirchandani, and Vachani [34]. Section B.2 describes the optimal solution of an SADP for facilities of two nested compartments, and Section B.3 translates to our notation the residual capacity inequality separation results of Atamtürk and Rajan.
B.1 Transformation of Segregated Compartments

SADP

This section details the transformation shown schematically in Figure 3.6 of Section 3 of the Single Arc Design Problem (SADP) for the segregated CNLP model to the standard SADP introduced by Magnanti, Mirchandani, and Vachani in [34]. The segregated compartments SADP and the SADP itself are formulated in Section 3.5. Assume positive facility costs and nonnegative commodity flow rewards.

The transformation exploits the continuity of the flow variables to disaggregate each commodity into possibly several commodities with the same unit contribution and whose demands sum to the original commodity demand. If we disaggregate the commodities such that the demand corresponding to the $k$th largest contribution in each commodity class can fill the same fraction of a facility, we can effectively merge the classes. For a fixed capacity, it is optimal to flow, in each class, commodity demand in order of decreasing unit reward, until either compartment capacity or commodity demand is exhausted.

Figure 3.6 depicts the disaggregation for an instance with commodities $a, b, c$ in one class and commodities $i, ii, iii, iv$ in the other. The length of each block is proportional to the number of facilities needed to accommodate the demand of the corresponding commodity, and the commodities in each class are sorted from greatest to least unit contribution. The figure effectively assumes that the capacity occupied by all first class commodities equals that occupied by all second class commodities. We can add dummy commodities with 0 contribution to ensure that this condition holds. In the
figure, the transformation disaggregates commodities so that each class has six, and then merges the commodities into a single class to yield an SADP.

More formally, suppose that \( \sum_{k \in Q} \frac{u_{k,l}^l}{c_l} \) equals some constant independent of \( l \), and that the commodities have been sorted so that \( a_{1,l} \geq a_{2,l} \geq \ldots \geq a_{Q,l} \). Let \( u_{l,k,l}^l = \frac{u_{k,l}^l}{c_l} \) so that the \( u' \) convert demands to facility requirements, and define \( \bar{u} \) as the corresponding cumulative facility requirements:

\[
\begin{align*}
\bar{u}_{l,1}^l &= u_{l,1,1}^l \\
\bar{u}_{l,k}^l &= \bar{u}_{l,k-1}^l + u_{l,k,l}^l, \quad k = 2, 3, \ldots \\
\end{align*}
\]

\( \bar{u} \) is obtained by concatenating the \( \bar{u} \) and \( \bar{u} \) and then sorting the combination from least to greatest. \( w' \) are the increments of \( \bar{w} \):

\[
\begin{align*}
w_{l,1} &= \bar{w}_{l}^1 \\
w_{l,k} &= \bar{w}_{l}^k - \bar{w}_{l}^{k-1}, \quad k = 2, 3, \ldots,
\end{align*}
\]

and \( w^k \) is obtained from \( w_{l,k} \) by converting from facility fractions back to demand value: \( w^k = w_{l,k} (\sum_{l \in K} C^l) \). The \( w^k \) define commodity demands of the new SADP, and the objective coefficients, \( d^k \), are defined by the corresponding contributions from each class:

\[
\begin{align*}
d^k &= \sum_{m=1}^l a_{k,m}^l \\
&\text{with } k(m) = \min \{ m \mid \bar{u}_{m,l} \geq \bar{w}_{l}^k \}.
\end{align*}
\]

The resulting SADP is then:

\[
\min - \sum d^k t^k + cz
\]

subject to:

\[
\begin{align*}
t^k &\leq w^k, \forall k \quad \text{(B.1)} \\
\sum_{l \in K} t^k &\leq (\sum_{l \in K} C^l) z \quad \text{(B.2)} \\
z &\geq 0, \text{ integer} \quad \text{(B.3)} \\
t^k &\geq 0 \quad \text{(B.4)}
\end{align*}
\]

The transformation works because we can convert any solution to the transformed
problem to a feasible solution to the original segregated problem of the same objective value, and vice versa. Furthermore, the transformation is polynomial since the number of commodities in the new problem is no more than $\sum_{i \in \mathcal{K}} |Q^i|$.

B.2 Optimal Solution for SADP of 2 Nested Compartments

This section describes the optimal solution for a Single Arc Design Problem of two nested compartments. All commodities can be added to compartment 1, and only a subset of commodities, say $Q^l$, can be added to the other. Let $f$ be the flows of commodities $Q^l$ and $v$ the corresponding demands. Let $Q^{II} = Q \setminus Q^l$, $g$ be the flows of $Q^{II}$, and $u$ the corresponding demands. $C^1$ and $C^2$ are the capacities of compartments 1 and 2, respectively.

We next define some relevant functions and quantities. Lemma 23 proves their correctness. Suppose that within each class the commodity contributions have been sorted, let $a_f(t)$ be the (nonincreasing) marginal commodities $Q^l$ contribution if $t$ units of commodities $Q^l$ have been loaded in sequence, and let $a_g(t)$ be defined analogously for class $Q^{II}$. Let $z^l = \min \{ z \mid a_f(C^2 z) \leq \frac{c}{C^1 + C^2} \}$, $z^{II} = \min \{ z \mid a_g(C^1 z) \leq \frac{c}{C^1 + C^2} \}$, and $z^0 = \min \{ z^l, z^{II} \}$. If $z^0 = z^{II}$, then $z^* = z^0 + \min \{ z \mid a_f(C^2 z^0 + (C^1 + C^2) z) \leq \frac{c}{C^1 + C^2} \}$, and if $z^0 = z^l$, $z^* = \min \{ z \mid C^2 a_f(C^2 z) + C^1 a_g(C^1 z) \leq c \}$. $z^*$ is the optimal loading for the LP relaxation of the SADP for two nested compartments.

The procedure for finding $z^*$ effectively flows class $Q^{II}$ and class $Q^l$ commodities.
continuously in the ratio $\frac{C_i^1}{C_i^2}$ until one of their marginal contributions hits the facility cost per volume, $\frac{c}{C_i^1+C_i^2}$. If class $Q'^I$ hits the limit first, then commodities $Q'^I$ only are loaded in both compartments until their marginal contribution also hits $\frac{c}{C_i^1+C_i^2}$. If $Q'$ hits the limit first, then placement in the $\frac{C_i^1}{C_i^2}$ ratio continues until the composite marginal contribution $\frac{C_i^1a_i+C_i^2a_l}{C_i^1+C_i^2}$ falls to $\frac{c}{C_i^1+C_i^2}$.

The optimal solution for the discrete problem can be found by comparing the optimal objective values for $[z^*]$ and $[z^+]$ facilities. Observe that for a given $z$ it is optimal to first ship the $C^2z$ most valuable units' worth of commodities $Q'^I$ and then the $C^1z$ most valuable units' worth of the remaining (unplaced) commodities.

**Lemma 23** The optimal loading for a Single Arc Design Problem of two nested compartments is the better of $[z^*]$ and $[z^+]$ facilities.

**Proof.** Concavity of the optimal solution as a function of capacity follows from this observation (as well as from general linear programming results). It follows that if $z^*$ solves the continuous problem then either $[z^*]$ or $[z^+]$ solves the discrete problem.

It remains to show correctness of $z^*$. Since the marginal contribution of each unit of this continuous allocation exceeds its marginal cost, there cannot exist a superior $z$ value less than $z^*$. Furthermore, no capacity larger than $z^*$ can yield a greater objective. The observed fixed capacity solution to the fixed capacity problem shows that the commodities effectively loaded by the $z^*$ calculation are optimal for fixed capacity $z^*$. Additionally, optimal commodity placement corresponding to a larger capacity entails all commodities of the $z^*$ solution and some additional commodities whose marginal contributions do not exceed the marginal capacity cost. □

### B.3 Notational Translation of Residual Capacity

**Separation Proof**

This appendix translates the separation argument of Atamtürk and Rajan [5] for the NLP to our notation for the general (mixed compartments) NLP. Recall that for this
CNLP model, a set of valid residual capacity inequalities can be written for each subset of compartments, and, that if \((f, z)\) represent a proposed flow and loading, respectively, on a given edge, the residual capacity inequality can be written as follows for a set of compartments \(S\) and a set of commodities \(P\):

\[
\sum_{i \in S} \sum_{k \in P} f_{k,l} - \tau z \leq (\lambda - 1)(\sum_{i \in S} C_l - r),
\]

where \(\mu = \left[ \frac{\sum_{i \in S} u_i^k}{\sum_{i \in S} C_l} \right]\), and \(r = \begin{cases} \sum_{i \in S} C_l, & \text{if } \sum_{k \in P} \text{umod } \sum_{i \in S} C_l = 0, \\ \sum_{k \in P} \text{umod } \sum_{i \in S} C_l, & \text{otherwise}. \end{cases}\)

We formulate the separation problem (with \((f, z)\) fixed) for variable compartments as the following nonlinear program.

\[
\max \sum_{l=1}^{L} \sum_{k \in Q^l} f_{k,l} - \tau z - (\lambda - 1)(C - r)
\]

subject to:

\[
C = \sum_{i=1}^{L} C_l q^l
\]

\[
s^k \leq \sum_{l \in K^k} p_{k,l}^l\quad (B.5)
\]

\[
p_{k,l} \leq s^k
\]

\[
p_{k,l} \leq q^l
\]

\[
\sum_{k \in Q^l} d^l \leq s^k
\]

\[
0 < r < C
\]

\[
\mu \in \mathbb{Z}^+
\]

\[
s^k, p_{k,l}^l, q^l \in \{0, 1\}\quad (B.12)
\]

Variable \(s^k\) indicates whether commodity \(k\) is selected; \(q^l\) indicates whether compartment \(l\) is selected; and \(p_{k,l}^l\) indicates whether commodity \(k\) and compartment \(l\) are both selected. The variable \(\mu\) is the minimum number of facilities needed to accommodate the selected commodities, assuming that capacity can be fully utilized. \(d^l\) is the given commodity \(k\) demand, \(L\) the set of compartments, \(Q\) the set of commodities, and \(Q^l\) the set of commodities permitted in compartment \(l\). Capacity constraints imply in-
equalities with \( r = C \). Consequently, the \( 0 < r < C \) constraint does not eliminate any potentially violated inequalities. \((\overline{f}, \overline{z})\) violates a residual capacity inequality if and only if the above maximum exceeds 0.

**Lemma 24** \((\overline{f}, \overline{z})\) does not violate any residual capacity inequalities for which \( \mu \leq \overline{z} \) or \( \mu \geq \overline{z} + 1 \).

**Proof.** Select a set of compartments and a set of commodities \( P \), and let \( C \) and \( \overline{f} \) be the corresponding capacity and composite flow of commodities \( P \), respectively. Suppose \( \mu \leq \overline{z} \). Then,
\[
\begin{align*}
\overline{f} - r\overline{z} - (\mu - 1)(C - r) \\
\leq \overline{f} - r\mu - \mu C + r\mu + C - r \\
= \overline{f} - C(\mu - 1) - r \\
= \overline{f} - \sum_{k \in P} d^k \leq 0.
\end{align*}
\]

Now, suppose \( \mu \geq \overline{z} + 1 \). Then,
\[
\begin{align*}
\overline{f} - r\overline{z} - (\mu - 1)(C - r) \\
\leq \overline{f} - r\overline{z} - (\overline{z})(C - r) \\
= \overline{f} - \overline{z}C \leq 0, \text{ by capacity constraints.} \quad \square
\end{align*}
\]

Since the residual capacity inequalities are valid, none is violated if \( \overline{z} \in Z^+ \).

Consequently, we replace \( \mu \) with \( [\overline{z}] \) and \( \mu - 1 \) with \( [\overline{z}] \), and the separation problem becomes:
\[
\text{max } \sum_{l=1}^L \sum_{k \in Q} f^{k,l} p^{k,l} - r\overline{z} - [\overline{z}](C - r)
\]
subject to:
\[
\begin{align*}
C &= \sum_{l=1}^L C^l q^l \\ 
\overline{s}^k &\leq \sum_{l \in K^k} p^{k,l} \\ 
\overline{p}^{k,l} &\leq \overline{q}^l \\ 
\sum_{k \in Q} d^k s^k &= C(\overline{z}] + r \\ 
0 < r < C \\ 
s^k, p^{k,l}, q^l &\in \{0, 1\}.
\end{align*}
\]

Substitution for \( r \) yields:
\[
\text{max } \sum_{l=1}^L \sum_{k \in Q} f^{k,l} p^{k,l} - (\overline{z} - [\overline{z}]) (\sum_{k \in Q} d^k s^k) - C [\overline{z}](\overline{z} - \overline{z})
\]
subject to:
\[
\begin{align*}
C &= \sum_{l=1}^{L} C^l q^l \\
\tilde{s}^k &\leq \sum_{l \in K^k} p^{k,l} \\
p^{k,l} &\leq \frac{\tilde{s}^k}{q^l} \\
p^{k,l} &\leq q^l \\
C[\tilde{z}] &< \sum_{k \in Q^s} d^k \tilde{s}^k < C[\tilde{z}] \\
\tilde{s}^k, p^{k,l}, q^l &\in \{0, 1\}.
\end{align*}
\]

For a fixed set of compartments \(S\), let \(Q^S\) denote those commodities with a positive objective coefficient. That is, \(Q^S = \{ k \mid \sum_{l \in S} \tilde{f}^{k,l} - (\tilde{z} - [\tilde{z}])d^k > 0 \}\).

**Lemma 25** If \((\tilde{f}, \tilde{z})\) violates a residual capacity inequality for a fixed set of compartments \(S\), then \((\tilde{f}, \tilde{z})\) violates the inequality defined by some commodity subset \(P \subseteq Q^S\).

**Proof.** Suppose \((\tilde{f}, \tilde{z})\) violates the inequality defined by commodities \(Q'\).  
Case 1: \(\sum_{k \in Q' \cap Q^S} d^k > C[\tilde{z}]\). 
\(Q'\) corresponds to a violated inequality, so \(C[\tilde{z}] > \sum_{k \in Q'} d^k \geq \sum_{k \in Q' \cap Q^S} d^k\). This inequality and the case assumption imply that \(Q' \cap Q^S\) satisfies the strict inequality constraint of the separation problem. The objective corresponding to \(Q' \cap Q^S\) is at least that corresponding to \(Q'\) by definition of \(Q^S\), \(Q' \cap Q^S\) also defines a violated inequality.  
Case 2: \(\sum_{k \in Q' \cap Q^S} d^k \leq C[\tilde{z}]\). 
In this case, \(\sum_{l \in S} \sum_{k \in Q'} \tilde{f}^{k,l} - (\tilde{z} - [\tilde{z}]) \left( \sum_{k \in Q'} d^k \right) - C[\tilde{z}] \left( [\tilde{z}] - \tilde{z} \right) \leq C[\tilde{z}] \left( [\tilde{z}] - \tilde{z} \right) \)
\[= \sum_{l \in S} \sum_{k \in Q' \cap Q^S} \tilde{f}^{k,l} - (\tilde{z} - [\tilde{z}]) \left( \sum_{k \in Q' \cap Q^S} d^k \right) \] 
\[\leq 0.\]
The first term in the last expression is nonpositive by definition of \(Q^S\), and the second is nonpositive by commodity demand bounds. The result contradicts violation of the inequality corresponding to \(Q'\). \(\square\)

**Lemma 26** If \(\sum_{k \in Q^S} d^k \leq C[\tilde{z}]\) or \(\sum_{k \in Q^S} d^k \geq C[\tilde{z}]\), then \((\tilde{f}, \tilde{z})\) violates no residual capacity inequalities corresponding to \(S\).

**Proof.** Suppose \(\sum_{k \in Q^S} d^k \leq C[\tilde{z}]\). Then the result follows from Lemma 25 since there is no \(P \subseteq Q^S\) such that \(\sum_{k \in P} d^k > C[\tilde{z}]\). Now suppose \(\sum_{k \in Q^S} d^k \geq C[\tilde{z}]\) and that \((\tilde{z}, \tilde{f})\) violates the inequality corresponding to \(P\). By Lemma 25, we may assume \(P \subseteq Q^S\). The separation objective for \(P\) is 
\[\sum_{l \in S} \sum_{k \in P} \tilde{f}^{k,l} - (\tilde{z} - [\tilde{z}]) \left( \sum_{k \in P} d^k \right) - C[\tilde{z}] \left( [\tilde{z}] - \tilde{z} \right)\]

169
\[
\leq \sum_{l \in S} \sum_{k \in Q^s} f^{k,l}_{l} - (\bar{z} - [\bar{z}]) (\sum_{k \in Q^s} d^k) - C[\bar{z}](\bar{z} - \bar{z}) \\
\leq C\bar{z} - (\bar{z} - [\bar{z}]) (C[\bar{z}]) - C[\bar{z}](\bar{z} - \bar{z}) = 0,
\]
contradicting the assumed violation. \( \square \)

So, the argument of Atamtürk and Rajan shows that a residual capacity inequality for a fixed set of compartments \( S \) is violated if and only if \( (\sum_{l \in S} C^l)[\bar{z}] < \sum_{k \in Q^s} d^k < (\sum_{l \in S} C^l)[\bar{z}] \) and
\[
\sum_{l \in S} \sum_{k \in Q^s} f^{k,l}_{l} - (\bar{z} - [\bar{z}]) (\sum_{k \in Q^s} d^k) - C[\bar{z}](\bar{z} - \bar{z}) > 0.
\]
The proof of Lemma 26 shows that if \( \sum_{k \in Q^s} d^k \geq C[\bar{z}] \), then
\[
\sum_{l \in S} \sum_{k \in Q^s} f^{k,l}_{l} - (\bar{z} - [\bar{z}]) (\sum_{k \in Q^s} d^k) - C[\bar{z}](\bar{z} - \bar{z}) \leq 0.
\]