The Thesis Committee for Chinmoy Mohapatra
Certifies that this is the approved version of the following thesis:

Multicommodity network flow models with FIFO transshipment handling policies

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor: David P. Morton

Co-Supervisor: Anantaram Balakrishnan

Multicommodity network flow models with FIFO transshipment handling policies

by

Chinmoy Mohapatra, B.Tech.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

August 2011

Dedication

I dedicate this work to my family.

Acknowledgements

My utmost gratitude goes to my advisor, Dr. Anant Balakrishnan, for his valuable guidance, insights and continuous support during this research and preparation of this thesis. Working under his supervision has been a great learning experience for me, and his patience and enthusiasm has always motivated me to work harder. I am also extremely thankful to my supervisor, Dr. David P. Morton, for his constructive comments and suggestions that have greatly improved and clarified this work. Finally, I would like to thank my family and all my friends who have always believed in me and supported me.

Abstract

Multicommodity network flow models with FIFO transshipment handling policies

Chinmoy Mohapatra, M.S.E

The University of Texas at Austin, 2011

Supervisors: David P. Morton and Anantaram Balakrishnan

Integer multicommodity network flow (MCNF) models have applications in various areas like logistics, freight transportation, telecommunication and manufacturing. In this thesis we study an extension of the integer MCNF problem (MCNF-FIFO) where commodities are handled (processed) in a first-in-first-out (FIFO) order at each transshipment location and resource capacities are shared across arcs in the network. The objective of the MCNF-FIFO model is to find feasible routes for all commodities from their origins to destinations while minimizing the total transportation and holding cost or the sum of delivery times.

We formulate the MCNF-FIFO problem on a time-space network and develop three different integer-programming (IP) formulations for the FIFO constraints, and two IP formulations for the flow conservations requirements. Since these formulations have a

very large number of variables and constraints, we develop various algorithmic strategies to obtain good quality solutions quickly. The first strategy is to reduce the problem size by using properties of the optimal solution. We develop novel problem reduction and decomposition techniques that eliminate variables and constraints, and decompose the problem into smaller components. To further reduce the problem size, we classify the FIFO constraints into different categories by utilizing the relationships between different commodities, and provide specialized formulations for each of these categories so as to reduce the number of FIFO constraints significantly. The second strategy is to develop heuristic algorithms that provide near-optimal solutions to the MCNF-FIFO problem. Our first algorithm is an optimization-based heuristic that solves a relaxed MCNF-FIFO model with a limited number of FIFO constraints. Then, it removes the remaining infeasibilities in the solution of the relaxed MCNF-FIFO model using a repair heuristic to obtain a feasible solution. We develop two other heuristic algorithms that are stand-alone construction heuristics that build a feasible solution from scratch.

To assess the effectiveness of the modeling and algorithmic enhancements, we implement the methods and apply them to three real life test instances. Our tests show that the problem reduction techniques are very effective in reducing the solution times. Among the heuristic algorithms, the optimization-based heuristic performs the best to find near-optimal solutions quickly.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 MOTIVATION FOR THESIS

Multicommodity network flow problems (MCNF) deal with routing several commodities across a network from their respective origins to destinations. The commodities share the resources available in the network. The broad subdivisions of MCNF are capacitated MCNF in which arcs have capacity restrictions, and uncapacitated MCNF. If the problem requires every commodity to take a unique path from its origin to destination, then it is an integer multicommodity network flow problem (Barnhart et al. 1996). Integer MCNF problems are used in many interesting applications in transportation, manufacturing, facility location, and telecommunication. When the nodes of a network are defined by not only geographical or spatial characteristics but also time instants or periods, and the arcs define the spatial and temporal connections between the nodes, then the network is called a time-space network. Many planning problems in industries such as logistics, telecommunication, and manufacturing require modeling them as an integer MCNF problem defined on time-space networks.

This thesis studies and solves an extension of the integer MCNF problem defined on a time-space network in which we make routing decisions for the commodities by considering not only the capacity constraints on available resources, but also the various operational policies that define the order in which commodities are handled inside a transshipment location. Specifically, we study the FIFO handling policy, which requires commodities to depart in a first-in-first-out (FIFO) order from every transshipment location. This model has many real life applications to planning problems in logistics, transportation, and scheduling.

To illustrate a general application of this model, we consider the shipment-dispatching problem in logistics planning. Suppose we have a set of shipments to be dispatched from their respective origins to destinations using transport services between different geographic locations that run according to a pre-specified schedule. The shipment-dispatching problem is to assign each of these shipments to a transport service so that every shipment reaches its destination, and the total cost of transportation and holding or the total sum of delivery times is minimized. The assignment of these shipments to various transport services has to obey the capacity limitations of these services. In many cases, due to legacy infrastructure or operating policies, there are constraints on the order in which shipments are processed through a transshipment location. One ordering rule is the first-in first-out rule, in which shipments are sent to the end of the processing queue as soon as they arrive at a location, and outgoing shipments are dispatched from the top of the processing queue. We refer to the capacitated routing problem with FIFO handling as the integer multicommodity network flow problem with FIFO constraints (MCNF-FIFO). The transport services between various locations and their respective schedules define the time-space network. The shipments to be dispatched are the commodities, and the available capacity of the transport services and the processing order rule impose capacity and FIFO constraints, respectively.

To consider a general and practical framework, we also add two important characteristics of such service networks to our model. The first one is consolidation operations that are central to many service networks (Crainic 2000). Apart from the specific origin, destination and shipment related physical characteristics; each individual shipment may have other requirements such as service quality or type of vehicle. To meet these requirements, planners often group the commodities together so that they can be assigned to a particular type of service. Moreover, at every intermediate location, the

commodities need to be reclassified according to their next destination and arranged for dispatch. Since the classification process is costly and time-consuming, it is always effective to keep the group intact until it needs to be split up or be reclassified. To incorporate these considerations and manage traffic effectively, such networks are generally modeled as multi-layered (logical and physical) networks. The physical layer consists of the actual transshipment locations and transport services between these locations. The logical layer is created on top of the physical layer, and contains only the locations at which a commodity needs to be reclassified. A shipment may take multiple logical links (links in the logical layer) to travel from its origin to destination since it can be reclassified at multiple locations. Moreover, different logical links can share the same physical link (transport service). This gives rise to the next important characteristic of service networks, namely, multiple logical links may share the capacity of the same physical transport service in the network. In the simplest version of this model, each logical link corresponds to a transport service, i.e., the logical layer is equivalent to the physical layer. Then, each commodity is reclassified at every location and every logical link has its dedicated capacity without any sharing.

Besides the above example application from freight transportation, many real life applications in different areas also have similar underlying network structure and requirements. A few examples of such applications are as follows:

- *Deterministic Traffic Flow:* One application is a deterministic traffic flow model for road networks where commodities correspond to vehicles; the road network is the underlying graph and the FIFO discipline of road traffic queues are the handling policies (Smith 1993).
- *Cargo Routing*: Another example is the cargo routing problem in the shipping industry. In this problem the commodities correspond to containers carrying

cargo, the network of ports connected through ship routes represent the underlying network, and the infrastructures at the *transshipment port*s define the handling rules for switching a container from one ship to another (Agarwal & Ergun 2008).

- *Crew Training Scheduling*: In airlines and many other industries, the training schedules and required resources are determined a priori, and assignments of the crews to different training programs are done later (Qi et al. 2004). Since the crew is still paid when they are undergoing training, the organization likes to assign crews to training schedules such that the total time spent by crew members on training is minimized. An operating rule can specify that the crew member who becomes available for training first be given priority over a later available crew. This problem then can also be expressed as a MCNF-FIFO model, where the training schedules define the time-space network for a one year time horizon, the crews to be trained form the set of commodities, and the priority rules define the FIFO constraints.

In general, there can be many different handling policies defining the interaction between commodities in a transshipment location, and it is important to consider them while routing the commodities through these locations to obtain practically implementable solutions. These applications of this class of problems to different areas motivate this thesis.

1.2 GOAL OF THESIS

In this research, we study extensions of the integer multicommodity network flow problem (MCNF-FIFO) where every commodity must follow a unique path from its origin to destination, and commodity routes must satisfy additional side constraints

representing FIFO handling policies at transshipment locations. The goal of this thesis is to:

"*Develop effective modeling and algorithmic strategies to solve the integer multicommodity network flow model with FIFO handling policies* (MCNF-FIFO) *optimally or near-optimally (with performance guarantees), within acceptable computational time for practical applications.*"

We pursue the following steps to achieve this objective:

- Develop integer-programming (IP) formulations for the MCNF-FIFO model so that practical problems can be solved effectively and efficiently by exact optimization methods. The primary focus is to develop formulations for this large-scale problem that have fewer constraints, provide tight linear-programming (LP) relaxations, and have sparse constraint matrices so as to quickly obtain good quality solutions.
- Develop algorithmic strategies to obtain good quality solutions to the MCNF-FIFO problem quickly. The strategies can be divided into the following classes:
    - Problem reduction and decomposition: Develop techniques to reduce problem size by fixing variables and eliminating constraints by exploiting certain properties of the optimal solution. Propose a method to decompose the problem into smaller sub-problems that can be solved independently and quickly.
    - Upper-bounding techniques: Develop three different heuristic algorithms that solve the problem quickly and provide good quality upper bounds. The heuristic algorithms are broadly divided into two classes:
        - Optimization-based heuristic: We propose an algorithm that first solves a relaxed optimization problem to obtain a good quality but

slightly infeasible solution. Then, we apply a repair heuristic to obtain a feasible solution to the problem.

  - Stand-alone heuristic: We develop two different stand-alone heuristic algorithms that construct feasible solutions from scratch. These are effective in generating good quality upper bounds that can be used to provide a warm start to exact optimization models.

- Investigate the performance of these modeling and algorithmic approaches to solve real life test instances. The main purpose is to determine which of the modeling enhancements and algorithmic approaches are most effective in solving the problem in a computationally effective manner. Finally, we provide a comparative study on the performance of different models and methods on various test instances.

1.3 STRUCTURE OF THESIS

The remainder of this thesis is divided into six chapters. Chapter 2 formally introduces the multicommodity network flow problem with FIFO transshipment handling policies. We first describe the time-space network over which we define the problem, and then introduce the FIFO handling policies, and discuss the inputs and assumptions for the model. Finally, we provide a general mathematical formulation of the model.

Chapter 3 provides a brief overview of the mathematical models and concepts from the past literature relevant to this research. We focus on applications and extensions of multicommodity network flow models, including problems on time-space networks and formulations involving side constraints. We particularly concentrate on extensions involving FIFO constraints and those that require a unique origin-to-destination path for each commodity. The objective in studying each of these models is to understand how

researchers have previously formulated such problems, and what solution techniques have proven to be effective.

Chapter 4 deals with various alternative IP formulations for the MCNF-FIFO problem. First, we classify the FIFO constraints into different categories, provide a specialized formulation for each category, and prove that this classification reduces the number of constraints compared to the formulation presented in Chapter 2. Then, we introduce three alternative formulations for the FIFO constraints, and two alternative formulations for the flow conservation constraints. We compare these formulations in terms of the number of constraints, strength of their LP relaxations, and sparsity of the constraint matrix.

In Chapter 5, we discuss algorithmic approaches to solve the MCNF-FIFO problem. First, we present an effective problem reduction technique that can fix some variables and eliminate constraints a priori, thereby making it easier to obtain optimal solutions to the model using exact optimization methods. Then we discuss a problem decomposition method, and finally introduce three different heuristic algorithms for the problem that can quickly provide good quality upper bounds.

Chapter 6 provides the details of the computational environment and the test data that we use to investigate the performance of the IP formulations, and the algorithmic approaches to solve the MCNF-FIFO problem. Our aim in this chapter is to explore and compare the quality of solutions provided by these methods, and judge their merit in terms of solving real life problems.

Finally in Chapter 7, we conclude this thesis by providing a summary of the findings and contributions of this research. We discuss the directions for further research and methods to extend the problem. In particular, we discuss extensions to include

different handling policies and further applications of the solution strategies presented in this thesis.

Chapter 2: Multicommodity network flows with FIFO transshipment handling policies

In this chapter, we first define a time-space network over which we formulate the MCNF-FIFO problem. Then, we formally define the MCNF-FIFO problem, the inputs and assumptions for the model, and explain the FIFO constraints in detail. Finally, we discuss a general mathematical formulation for the problem.

2.1 NOTATION AND DEFINITIONS

Let L represent the set of locations and K represent the set of commodities. Each commodity $k \in K$ originates at location $o_k \in L$ and travels to its destination $d_k \in L$, through a set of transshipment locations $L(k) \subset L$. Note that $L(k)$ only represents the set of locations where a commodity k is processed (origin and all transshipment locations except destination location) and destination location, i.e., it represents the nodes in the logical network for commodity k. ta_k denotes the arrival time of commodity k at its origin o_k. Thus, a commodity enters the system at ta_k and is ready to be routed to its destination.

Let R represent the set of resources. We have different metrics to measure the capacity usage of a resource. Examples of different capacity usage metrics are length of commodities, weight of commodities and number of commodities. Let M represent the set of different capacity usage metrics. We define the capacity of each resource $r \in R$ by v_r^m for each capacity usage metric $m \in M$. Similarly, u_k^m represents the capacity required by commodity $k \in K$ for different metrics $m \in M$. Based on these notations and definitions, we now describe the time-space network for the model.

2.2 THE TIME-SPACE NETWORK

The time-space network representation has been quite popular to model problems that have a time dimension, and we discuss many applications of multicommodity network flow problems on time-space networks in our literature review in Chapter 3. We represent the time-space network as $G = (N, A)$, where N is the set of nodes in the network and A is the set of arcs. Each node in the time-space network is uniquely characterized by a location and a time instant at which either an arc enters or departs from a location. The time-space network in our model contains three types of nodes. $N_o^k \subset N$ is the set of origin nodes that represents the arrival of a commodity $k \in K$ at its origin location o_k at its origin arrival time ta_k. We denote each origin node as $<o_k, ta_k> \in N_o$ and the supply of each commodity at the origin node is 1. Figure 1 shows an example time-space network for a single commodity k, with origin o_k and destination d_k. In this example, node $<o_k, t1>$ is the origin node. The destination nodes $N_d^k \subset N$ represent the different times at which a commodity $k \in K$ can arrive at its destination location d_k. In Figure 1, commodity k can arrive at its destination d_k at times $t11$, $t12$ and $t13$. Thus, the set of destination nodes for k is denoted by $<d_k, t11>$, $<d_k, t12>$ and $<d_k, t13>$, respectively. Finally, we have the set of transshipment nodes $N_{lt}^k \subset N$, corresponding to each time point t at which a commodity $k \in K$ can arrive or depart from any transshipment location $l \in L(k)$. This set excludes the origin nodes and the destination nodes. Note that the set of transshipment nodes includes the nodes corresponding to the origin location of a commodity k and all time points at which it can depart from its origin. There is no supply or demand at the transshipment nodes. In Figure 1, $<o_k, t3>$ and $<l, t5>$ are examples of transshipment nodes.

The different times at which arcs can arrive or depart from a location l are called events, and $T(l)$ denotes the set of events at a location l. $T(l,k) \subset T(l)$, denotes the times at

which a commodity k can arrive or depart from a location l. For example in Figure 1, $\{t1, t2, t3, t4\} \subset T(o_k)$ is the set of arrival and departure events at o_k. In general we sequentially index all the event times at a particular location $l \in L$ from 1,…,$|T(l)|$, and for any $t \in T(l)$, t-1 represents the previous event time at location l and t+1 represents the next time instant.

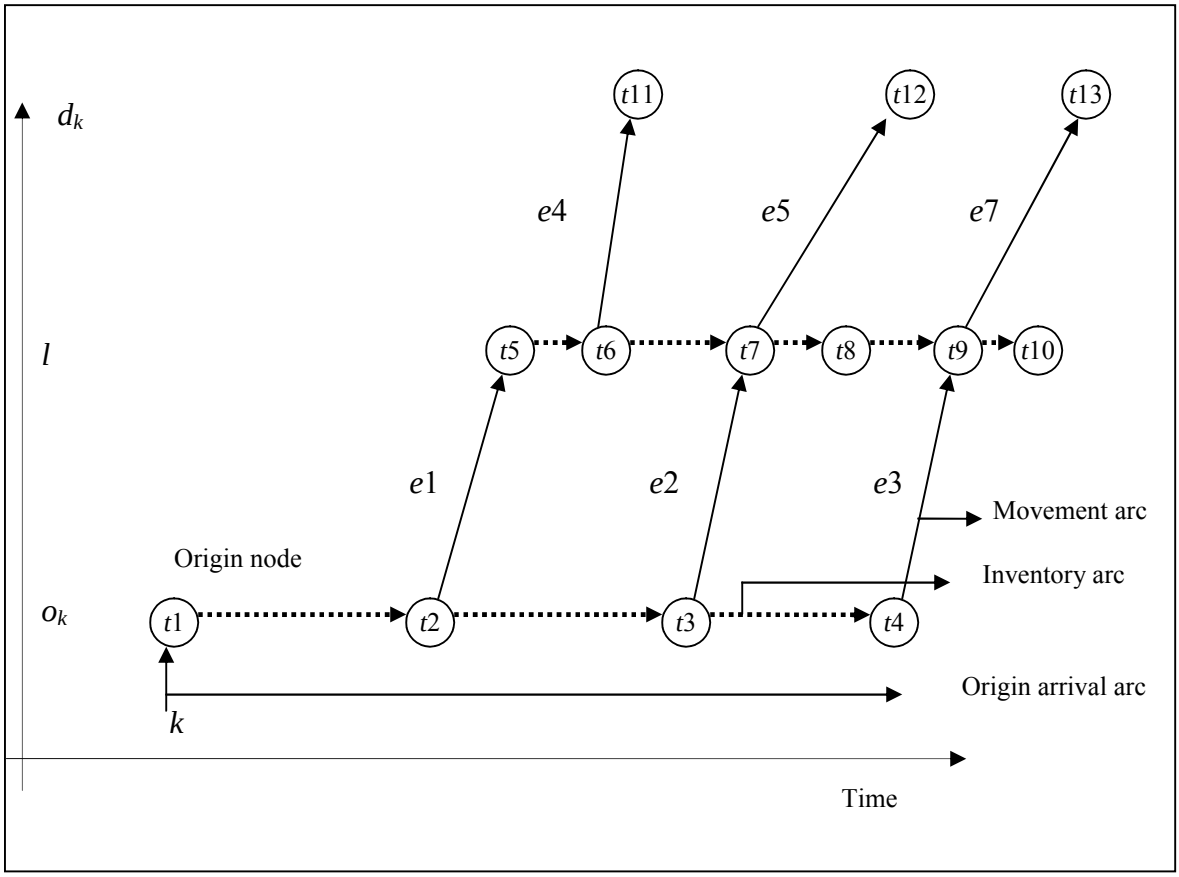

Figure 1: Time-space network example for a single commodity

The set of arcs A in the time-space network consists of three subsets: origin arrival arcs, inventory arcs, and movement arcs. The origin arrival arcs A_{OA} enter the nodes in N_o, i.e., they represent the arrival of a commodity k at its origin. The set of inventory arcs A_I connect the nodes $<l,t> \in N_{lt} \cup N_o$ and $<l,t+1> \in N_{lt}$ at each location l for every

commodity k. Thus, these arcs carry the inventory of a commodity k at a location l from every time point t to the next time point t+1. We take a commodity out of the system as soon as it reaches its destination; so there are no inventory arcs at the destination of a commodity. Note that we do not carry forward any inventory after the end of the horizon. Thus, no inventory arc enters the first node at a location or leaves the last node at a location. In Figure 1, the arcs connecting the nodes $<o_k,t1>$ and $<o_k,t2>$, and $<l,t6>$ and $<l,t7>$ are examples of inventory arcs.

The movement arcs A_M transport a commodity from one location to its next transshipment location. Thus, a movement arc starts from a node $<l,t> \in N_{lt} \cup N_o$ and ends at a node $<s,t'> \in N_{lt} \cup N_d$, where $s \neq l$ and $t' \geq t$. The movement arcs in our model correspond to transport on the logical links for each commodity. Thus, multiple movement arcs may share the capacity of a resource $r \in R$ available in the network, and $A(r)$ denotes the set of movement arcs that share resource r. We call the set of movement arcs that a commodity k can take as $A(k)$. We further divide the set $A(k)$ into entering and leaving arcs at each location; we denote the set of arcs a commodity k can take to enter a location l as $AE(k,l)$ and those that on which the commodity can depart the location as $AS(k,l)$. In Figure 1, arcs $e1$-$e7$ are the movement arcs, where $\{e1, e2, e3\} \subset AE(k, l)$ and $\{e4, e5, e6, e7\} \subset AS(k, l)$.

In our model, the costs of origin arrival arcs are set to 0. The costs of inventory arcs are h_{kt}^{l}, and correspond to the cost of holding inventory of commodity k at location l from node $<l, t\text{-}1>$ to $<l,t>$. Similarly, the costs of movement arcs are c_e^k, and correspond to the cost of transporting commodity k using movement arc $e \in A(k)$.

2.3 PROBLEM DEFINITION

Based on the above definitions and the time-space network, we define MCNF-FIFO as the problem of transporting all the commodities *k* arriving within a pre-specified decision horizon *T*, from their respective origins to the destinations, while:

- minimizing the total transportation cost in the system;
- satisfying the capacity restrictions of each resource;
- routing each commodity on a unique path from its origin to destination, i.e., a commodity is not allowed to split in its transit; and,
- following the FIFO handling policy at each location.

We make the following assumptions and simplifications in our model:

- We assume that the locations have enough capacity to handle all the commodities that flow through them.
- We assume that there is no "due date" or deadline by which each commodity must be delivered to its destination. However, we can readily incorporate commodity due dates by limiting the set of destination nodes for each commodity.

This problem is an extension of the integer min-cost multicommodity network flow problem (Barnhart et al. 1996), obtained by adding the FIFO handling policies as constraints. The handling policies are primarily a set of rules that define the order in which commodities are processed at each transshipment location. These rules are created based on many different factors, such as the existing infrastructure at a location or management decisions to give priority to a particular class of commodities over others. For example, in rail networks, the infrastructure in the classification yards (transshipment locations) defines the order in which the incoming rail cars are classified and made ready to be dispatched (Crane et al. 1955). If the solution routes do not follow these rules, then the proposed solution cannot be implemented. Similarly, in a supply chain with a

distribution center receiving several products from different plants and supplying these products to retail stores, managers may decide to give priorities to some products over others regardless of their arrival time or holding costs, thereby dictating the order in which these products are processed in the distribution center.

A common handling policy is the FIFO policy. The FIFO policy states that if there are two commodities *k*1 and *k*2 such that *k*1 arrives at location *l* before *k*2, then *k*2 must leave the location *l* with or after *k*1. Figure 2 provides an example of a FIFO violation at location *l*. Although it is quite simple to state, the FIFO policy is quite difficult to formulate mathematically. This is because the arrival times of any two commodities at a location with respect to each other are not known a priori. For example, in Figure 2, if *k*1 arrives at *l* before *k*2, then it must depart with or before *k*2, and vice versa. We can see that as the number of commodities and arrival and departure arcs at a location increase, the number of such combinations increases greatly, making it difficult to formulate a constraint for each combination.

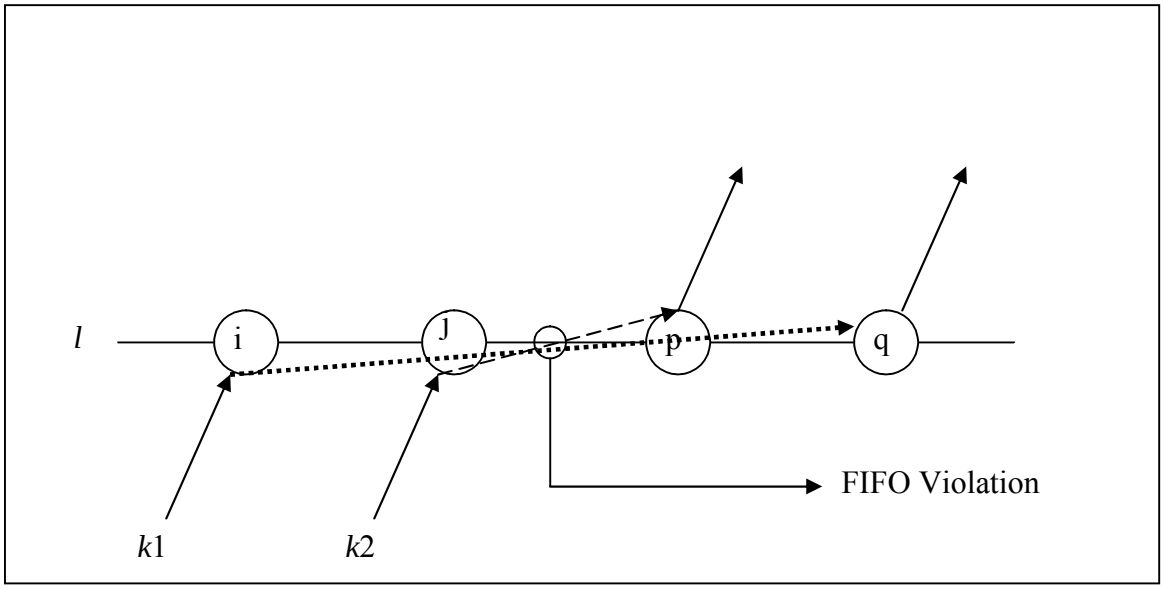

Figure 2: FIFO violation example

In the FIFO formulations in this thesis, we assume that the FIFO rule is applied for each pair of arrival and departure arcs at a location *l*. In many practical situations, this assumption may not necessarily be the requirement. If two commodities *k*1 and *k*2 are travelling to two different transshipment locations from the current location *l*, i.e., they do not share any outgoing arcs from *l*, they may be handled differently at *l* and hence do not interfere with each other in terms of the FIFO rule. This can be easily incorporated in our formulations by creating FIFO constraints just among commodities which share some outgoing arcs from a location *l*, i.e., commodity pair *k*1 and *k*2 with $AS(k1,l) \cap AS(k2,l) \neq \phi$.

2.3 MATHEMATICAL FORMULATION

The mathematical formulation that we discuss next has three different types of variables. The *movement variable* x_e^k denotes the assignment of a commodity *k* to a movement arc $e \in A(k)$. The *inventory variable* z_{kt}^l denotes the inventory of commodity *k* that is carried into event time $t \in T(k,l)$ at location *l* from the previous event time (*t*-1). The previous event can either be the arrival or departure of an arc. As we discuss above, there is no carry forward inventory for any commodity at the end of the decision horizon *T*; hence the initial and final inventory for every commodity at every location is 0. Finally, we have *indicator variables* to denote the connection between an incoming and outgoing arc that a commodity makes at a location *l*. The indicator variable y_{ef}^k is 1 if and only if $x_e^k = 1$ and $x_f^k = 1$, i.e., if the commodity *k* enters location *l* through arc $e \in AE(k,l)$ and "connects" to a departing arc using arc $f \in AS(k,l)$.

SETS

$[0, T]$	Decision horizon
G	Time-space network
N	Set of nodes in network G
A	Set of arcs in network G
L	Set of locations
K	Set of commodities
R	Set of resources
M	Set of different metrics for capacity usage of any resource; examples of different metrics are length, weight, and number of commodities.
$L(k)$	Set of locations that commodity k visits
$A(k)$	Set of movement arcs that a commodity $k \in K$ can use
$AS(k,l)$	Set of movement arcs that a commodity k can take to depart from location l, where $l \in L(k)$ and $k \in K$
$AE(k,l)$	Set of movement arcs that a commodity $k \in K$ can take to reach location $l \in L(k)$
$A(r)$	Set of arcs that use resource $r \in R$
$K(e)$	Set of commodities that can be assigned to arc $e \in$A
$T(l)$	Set of event times (arrival or departure of an arc) at location $l \in L$
$T(l, k)$	Set of event times (arrival or departure of an arc) at location $l \in L(k)$ for commodity $k \in K$

PARAMETERS

ts_e	Starting time of arc $e \in A$
te_e	Ending time of arc $e \in A$
o_k	Origin location of commodity $k \in K$
d_k	Destination location of commodity $k \in K$
ta_k	Arrival time of commodity k at its origin
u_k^m	Capacity required by commodity k for each metric $m \in M$
v_r^m	Maximum available capacity of resource $r \in R$ for metric $m \in M$
c_e^k	Cost of assigning commodity k to arc $e \in A(k)$
h_{kt}^l	Cost of holding inventory of commodity k at location $l \in L$ at t time $t \in T(l,k)$

Table 1: Sets, parameters and decision variables

DECISION VARIABLES

x_e^k	The movement variable which is 1 if commodity k is assigned to arc $e \in A(k)$, otherwise 0
z_{kt}^l	The inventory variable denoting the inventory of commodity $k \in K$ at location $l \in L(k)$ at time $t \in T(l,k)$
y_{ef}^k	The indicator variable which is 1 if the commodity k enters location l through an arc $e \in AE\ (k,l)$ and leaves through an arc $f \in AS\ (k,l)$

Table 1: Sets, parameters and decision variables (continued)

With these parameters and decision variables, we formulate MCNF-FIFO as follows:

[MCNF-FIFO] Minimize Transportation Cost:

$$\textbf{\textit{Minimize}} \quad \sum_{k \in K} \sum_{e \in A(k)} c_e^k x_e^k + \sum_{k \in K} \sum_{\substack{t \in T(l,k): \\ l \in L(k) \setminus \{d_k\}}} h_{kt}^l z_{kt}^l \qquad \ldots (1)$$

subject to:

Dispatch from Origin:

$$\sum_{e \in AS(k,o_k)} x_e^k = 1 \qquad \text{for each } k \in K \qquad \ldots(2)$$

Arrive at Destination:

$$\sum_{e \in AE(k,d_k)} x_e^k = 1 \qquad \text{for each } k \in K \qquad \ldots(3)$$

Flow Conservation:

$$\sum_{\substack{e \in AE(k,l) \\ :te_e = t}} x_e^k + z_{kt}^l = \sum_{\substack{f \in AS(k,l) \\ :ts_f = t}} x_f^k + z_{k,t+1}^l \qquad \text{for each } k \in K,\ t, t+1 \in T(l,k),$$

$$l \in L(k) \setminus \{d_k\} \qquad \ldots (4)$$

Capacity:

$$\sum_{e \in A(r)} \sum_{k \in K(e)} u_k^m x_e^k \le v_r^m \qquad \text{for each } r \in R,\ m \in M \qquad \ldots(5)$$

Connection:

$$x_e^k + x_f^k - y_{ef}^k \le 1 \qquad \text{for each } k \in K, e \in AE(k,l), f \in AS(k,l), l \in L(k) \setminus \{d_k\} \qquad \ldots 6(a)$$

$$x_e^k + x_f^k - 2y_{ef}^k \ge 0 \qquad \ldots 6(b)$$

Commodity-pair FIFO:

$$\sum_{\substack{f' \in AS(k,l) \\ : ts_{f'} \ge ts_f}} y_{ef'}^k + \sum_{\substack{q \in AS(k',l) \\ : ts_q < ts_f}} y_{pq}^{k'} \le 1 \qquad \text{for each } l \in L,\ k, k' \in K(l), e \in AE(k,l), f \in AS(k,l), p \in AE(k',l) \text{ and } te_p > te_e \ldots (7)$$

Integrality:

$$x_e^k, z_{kt}^l, y_{ef}^k \in \{0,1\} \qquad \text{for each } k \in K, l \in L(k), e, f \in A(k), t \in T(l,k) \qquad \ldots (8)$$

The objective function (1) minimizes the total transportation and holding cost of sending all commodities from their origins to destinations. The first part of the objective function captures the cost of assigning commodities to different movement arcs, and the second part captures the cost of holding inventory at various locations. A special case of this objective function with appropriately defined arc costs allows for minimizing the sum of delivery times of all the commodities at their destinations. For this special case, we assume that there is no cost of holding inventory at any of the locations and there is

no transportation cost except for the last arc entering the commodities destination. The modified objective function is,

Minimize Sum of Delivery times:

$$\text{Minimize} \quad \sum_{k \in K} \sum_{e \in AE(k,d_k)} te_e x_e^k \qquad \ldots(1a)$$

In objective function (1*a*), $AE(k,d_k)$ is the set of arcs that a commodity k can use to reach its destination, and te_e is the arrival time of the arc $e \in AE(k,d_k)$ at its destination. For the rest of the thesis, we assume that our objective is to minimize the sum of delivery times of all commodities. Note that most of the modeling enhancements and algorithmic techniques we discuss in later chapters are applicable to both the objective functions in (1) and (1*a*). We specifically point out when we develop any theory that is applicable only to the objective function (1*a*) but may not apply to the objective function (1).

Constraints (2) and (3) ensure that every commodity leaves its origin and reaches its destination. The flow conservation constraints (4) ensure that the flow is conserved at the origin and at every intermediate location for each commodity. There is a summation term for the incoming arcs and the outgoing arcs at each time point since there can be multiple arcs entering a location at the same time or leaving a location at the same time. The capacity constraints (5) aggregate all the commodities assigned to a given arc and aggregate all the arcs that can use a resource to ensure that the capacity usage of each resource does not exceed the available capacity. Constraints (6) are the connection constraints that enforce the definition of the indicator variables. Constraints (7) are the FIFO constraints which enforce the FIFO order of arrival and departure at any particular location. We can understand the constraints better through the example in Figure 3, and comparing the indices in the figure with those in constraints (7).

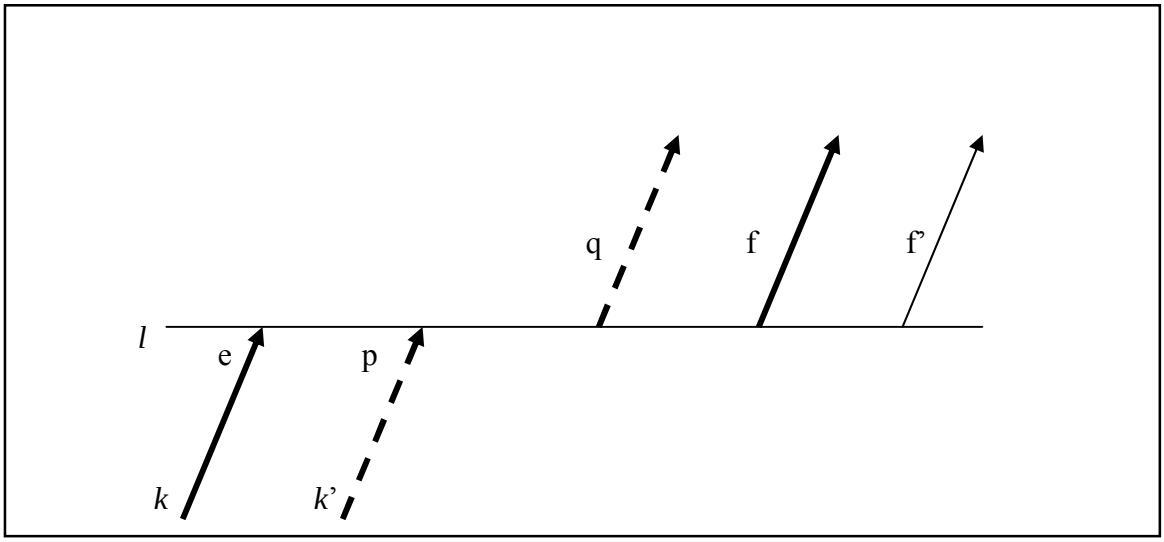

Figure 3: Example time-space network to understand FIFO constraints

In this example, suppose commodity *k* enters location *l* through arc *e* and commodity *k'* enters location *l* through arc *p*. Assume, commodity *k* leaves through arc *f*, then the corresponding connection variable y_{ef}^{k} =1. Similarly, if the commodity *k*' leaves through arc *q*, then the corresponding connection variable $y_{pq}^{k'}$ =1. These connections create a FIFO violation since commodity *k* arrives earlier than commodity *k*' but departs later than *k*'. Constraints (7) prevent this situation. If we expand the FIFO constraints for this scenario, we get:

$$y_{ef}^{k} + y_{ef'}^{k} + y_{pq}^{k'} \leq 1,$$

which prevents y_{ef}^{k} and $y_{pq}^{k'}$ to be simultaneously 1, thus enforcing a FIFO order of flow through location *l*. Since we create a constraint for each pair of commodities and every pair of arrival and departure arcs, we call this formulation as the commodity-pair FIFO formulation. Finally, constraints (8) enforce the integrality condition on all the variables.

The most difficult constraints in the above formulation are the FIFO constraints because the number of constraints can be huge even for a small instance of a real life problem. In Chapter 4, we discuss in detail the disadvantages of this formulation, and

provide various improved alternative versions of the FIFO constraints that are tighter as well as lower in number.

Chapter 3: Related literature

Multicommodity network flow models are one of the most studied problems in the operations research literature owing to their extensive application in numerous industries such as telecommunication, urban traffic, railway systems, production-distribution, logistics and scheduling systems, and military operations. It is interesting to note that even though the general multicommodity network flow model and its extensions have been studied extensively, and effective algorithms have been developed for them, not much work has been done on extending the general model to include the handling policies at transshipment locations that we study in this thesis.

In this chapter, we focus on past work related to the basic multicommodity network flow (MCNF) models and its extensions. As the existing literature on this topic is vast, we limit our discussions to applications and extensions of the MCNF models, where the structure of the problem is similar to our problem. Primarily, we focus on applications where the MCNF model has been applied to a time-space network or has been extended to include side constraints, specifically the FIFO constraints or any other handling policy. For each of the research articles we review, we first discuss the problem context briefly, and then discuss the modeling and algorithmic approaches taken by the authors and their effectiveness in finding good solutions. This review has potentially two advantages; the solution strategies proven successful for these problems can be applied to our problem since the problem structure is similar, and also the extensions of these problems can be potential applications of our model.

The basic (without any additional side constraints) multicommodity network flow model can be divided into classes of linear and non-linear models. There are numerous

papers dealing with both classes of problems and many different solution techniques have been developed for them. The solution techniques can be categorized into three classes:

- *Price directive decomposition* where the capacity constraints are relaxed using *Lagrange multipliers* and different methods are used for solving the dual sub-problem.
- *Resource directive decomposition* where the arc capacities are allocated to the K commodities and in each iteration K sub-problems are solved to finally reach an optimal allocation. One of the popular methods to find the near optimal capacity allocation is *sub-gradient optimization*.
- *Set partitioning* methods those maintain a small working basis of the constraint matrix with respect to the original problem and update the basis on the fly.

These methods provide the basic approaches upon which specialized algorithms are developed. We refer the reader to excellent papers by Assad (1978), Kennington (1978) and Awerbuch and Leighton (1993) that provide a detailed discussion of all these approaches and survey the algorithms based on these methods.

Authors	Application Area	Problem characteristics	Solution Techniques
Rao and Zionts (1968)	Transportation	Time-space network Side constraints for shipping commitments and time restrictions	Multiple formulations Column generation based algorithm
White and Bomberault (1969)	Transportation	Time-space network	Inductive heuristic algorithm
Bellmore et al. (1971)	Scheduling	Time-space network	Dantzig-Wolfe (DW) decomposition to solve LP Cutting plane and branch & bound
Haghani and Oh (1996)	Disaster relief management	Time-space network	Heuristic 1: Lagrangian relaxation based heuristic Heuristic 2: LP rounding based heuristic
Aggarwal (1995)	Production planning & scheduling	Time-space network Integer flows	Heuristic algorithm using parametric analysis
Barnhart et al. (1996)	General	Flow is not allowed to split	Column generation to solve LP and use it in branch & bound tree
Ozdaglar and Bertsekas (2003)	Optical networks	Flow is not allowed to split	Piecewise linear convex cost function
Wollmer (1971)	General	Side constraints in resources	Column generation
Chen and Chin (1992)	General	Side constraints on arcs	Column generation
Holmberg and Yuan (2003)	Telecommunication	Side constraints on paths	Column generation and constrained shortest path problem

Table 2: Summary of some past research

Authors	Application Area	Problem characteristics	Solution Techniques
Jha et al. (2008)	Railroad	Time-space network Integer flows	VLSN heuristic
Kwon (1998)	Railroad	Time-space network Side constraints to incorporate service quality and traffic variability constraints	Column generation based algorithm
Florian et al. (1972)	Railroad, scheduling	Time-space network	Benders' decomposition
Booler (1980)	Locomotive scheduling	Time-space network	Optimization based heuristic
Forbes et al. (1991)	Locomotive scheduling	Time-space network	Exact optimization
Ahuja et al. (2005)	Locomotive scheduling	Time-space network Operational side constraints	Multi-stage heuristic algorithm
Vaidyanathan et al. (2007)	Railroad crew scheduling	Time-space network FIFO operational constraints	Successive constraint generation Cost perturbation techniques

Table 2: Summary of some past research (continued)

There are many applications in the literature where the multicommodity network flow model has been studied on an underlying time-space network. Rao and Zionts (1968) study the problem of allocating transportation units to alternative trips. This model can be applied to different applications such as the allocation of railroad cars to

alternative trips and routing a fleet of vessels. In these applications, each shipment denotes a commodity, a trip from one port to another whether loaded or empty is represented by an arc, and the ports represent nodes in the network. Rao and Zionts develop two MCNF formulations, a disaggregated version and a compact version by aggregating round-trips into *cycles*. The models also contain additional constraints for satisfying supply commitments and incorporating the running time for each type of transportation unit that carries the commodities. As the number of variables in this case is huge, the authors develop a column generation based method to solve the problem but report that its computational capability is limited as only small instances of problem could be solved.

White and Bomberault (1969) study a similar problem for re-distributing empty rail cars in a railway system in anticipation of future demand. The empty rail cars are the commodities and the physical rail network, rail yards and train schedules form the time-space network. The authors assume that all the rail cars are of the same type. Based on this assumption, they propose an inductive heuristic algorithm that solves only a part of the network as a sub-problem and satisfies the demand at a subset of nodes, and then adds more nodes to this sub-problem iteratively. The authors use the acyclic property of the time-space network to label their nodes in topological order and apply this algorithm.

Bellmore et al. (1971) consider the problem of routing and scheduling a fixed fleet of non-homogeneous tankers to make a pre-specified set of shipments. The authors create a time-space network in which each node corresponds to the arrival of a shipment at a delivery port or its departure from its origin port, at one of the allowable times. The arcs either correspond to a tanker from a shipment's origin port to its destination port, or a reassignment of a tanker from a destination port to a shipment's origin port. The different types of shipments represent the commodities, and the objective is to determine

a schedule and route for tankers while minimizing operational cost. The authors formulate the problem as a mixed integer MCNF model and apply Dantzig-Wolfe decomposition to solve the LP relaxation, and then apply a branch and bound algorithm to obtain integer solutions.

Haghani and Oh (1996) study another interesting application of an MCNF model on a time-space network. The authors develop a multicommodity multi-modal network flow model for disaster relief management. The nodes in the network represent the supply and demand points for each mode of transport over time, and the arcs represent the connecting routes between these points. In this paper, the time-space network can be seen as an overlay of several individual networks corresponding to each mode, similar to our model. The overlaid networks are coupled together by transfer links, which allow commodities to be transferred from one mode to another. Haghani and Oh formulate the problem as an MCNF problem in which they have constraints for commodities as well as for vehicles belonging to different modes. They propose two heuristic procedures for the problem; a Lagrangian relaxation based heuristic where the linking constraints between the commodities and vehicles are dualized, and an LP rounding heuristic where they solve the LP relaxation first and round some integer variables, and continue solving until all variables are integer.

Aggarwal (1995) proposes a heuristic procedure to solve the multicommodity network flow problem with integer flows, and applies it to the equipment replacement problem that determines the time and quantity of the old equipment to be replaced while minimizing operational cost. The heuristic primarily tries to find the best allocation of the available capacity among each commodity. The algorithm starts by finding an initial feasible solution by using problem characteristics. Then it uses dual variable information

and parametric analysis to reallocate capacity among commodities, one arc at a time, with an aim to improve the objective value.

Florian et al. (1972) introduce the locomotive scheduling problem in railway networks and model it as a multicommodity network flow problem in a time-space network. The primary objective of the problem is to provide each train with sufficient engines to meet its motive power requirement while minimizing the total capital investment and the operating cost. In the time-space network, stations and arrival and departure times of different trains characterize the nodes, and the trains form the arcs. There are different types of engines in the model which represent the commodities and the flow of commodity on an arc denotes the number of engines allocated to that train. The authors use the data from Canadian Railways and apply Benders' decomposition technique to solve the problem. The authors report good solution times for medium size problem but unsatisfactory solution times for large instances. Booler (1980) addresses the same problem but considers each train as a *job* and finds the set of locomotive workings to cover all jobs in minimum cost. Booler provides an LP relaxation based heuristic algorithm to solve the problem and reports results on smaller instances. Forbes et al. (1991) provide an exact optimization model for the problem and solve the problems with branch and bound using intelligent branching priority rules. Ahuja et al. (2005) extend the model to include many other operating constraints and solve a weekly repeating locomotive scheduling problem as a multicommodity network flow problem with side constraints. They develop a multi-stage heuristic algorithm where at each stage they solve the model for only a single type of decision variables. They provide computational results for real life instances of the problem provided by CSX railways.

Kim et al. (1999) study another interesting application of multicommodity network flow model on a time-space network. The authors study the problem of routing

packages from their origin to destination given a limited number of transport services, limited capacity at intermediate locations and very tight service windows. They develop valid inequalities to improve the lower bound and apply a node consolidation problem reduction technique to reduce the problem size significantly. Finally, the authors apply heuristic algorithms to generate integer feasible solutions to this very large scale optimization problem.

As evident, there are many applications where the multicommodity network flow model and its extensions have been applied on a time-space network and different solution techniques including column generation, Lagrangian relaxation, cutting planes, problem reduction, branching priority rules and optimization based as well as stand-alone heuristic algorithms have been proposed. The solution techniques to be applied in a particular case mainly depend on the problem size and properties of the formulation that make the sub-problems easier to solve. An important point to note here is that many techniques such as column generation and Lagrangian relaxation depend heavily upon the ease of solving the sub-problems. The addition of FIFO constraints to the model breaks the special structure of the sub-problems and renders the problem difficult to solve. Next we discuss some extensions of the basic multicommodity network flow problem that are relevant to our model. Our problem requires that a commodity cannot split at any point, i.e., a commodity must travel in a unique path from its origin to destination. This extension of general MCNF is called the integer multicommodity network flow problem. Barnhart et al. (1996) solve the integer MCNF problem by considering a path-based formulation. The authors solve the LP relaxation of the model using column generation techniques and develop a method to use this LP solution in a branch and bound tree, called branch and price, to generate lower bounds. The authors report that the technique performed well in finding good integer feasible solutions when compared to the normal

branch and bound procedure used by commercial software such as CPLEX, but found difficulty in proving optimality.

Ozdaglar and Bertsekas (2003) propose another method to obtain optimal solutions to the integer multicommodity network flow problem in the context of routing and wavelength assignment problem in optical networks. Ozdaglar and Bertsekas use piecewise linear cost function with integer break-points for each arc and solve the LP relaxation directly to obtain integer solutions in many instances. This problem can be solved easily using commercial software packages, as only the LP model needs to be solved. Though, this result does not hold in general. The authors point out that for a majority of network structures, solving only the relaxed problem with this cost structure leads to integer optimal solutions.

Another important characteristic of our model is the sharing of resources across arcs. Wollmer (1971) introduces the multicommodity network flow problem with resource constraints where the arcs share common resources. The general multicommodity network flow problems that we discuss above have individual arcs subject to capacity constraints, while the problem Wollmer describes is more general because multiple arcs can share the same resource, and each resource is subject to capacity constraints, similar to our problem. There are many applications of this general model, for example, generalized traffic flow, problems with node capacities, and aircraft allocation problems. Wollmer presents two different formulations for the problem and uses column generation and DW decomposition to obtain solutions.

Many real world applications require additional extensions to the general multicommodity network flow model apart from unique flow restrictions and shared resources. These extensions stem generally from the requirements of real world operations and form additional side constraints to the general (integer) multicommodity

network flow model. One important paper closely related to our research is by Vaidyanathan et al. (2007) who develop an optimization model for crew scheduling for railroads in North America. Apart from the various crew union rules, an important operational constraint in this problem is that the crew belonging to same crew pool should follow FIFO order of departure from a station. They model this problem as a multicommodity network flow problem with FIFO side constraints on a time-space network. They formulate the FIFO constraint among each pair of incoming and outgoing train and aggregate it for each crew in a crew pool using a big M factor. This is a weaker version of the commodity-pair FIFO formulation we discuss in Chapter 2. As we discuss later and investigate in our computational results, this formulation may not be very efficient to obtain practical solutions due to large problem size and a weak LP relaxation. The authors discuss two different strategies to solve this problem. The first one is an exact optimization strategy in which they solve the problem without FIFO constraints first, and then iteratively apply a subset of FIFO constraints by examining the relaxed solution. The procedure terminates when integrality and FIFO violations are satisfied. The authors report that this method of removing FIFO violations iteratively may not be very efficient in solving real-life problems. The other technique is a heuristic method in which they perturb the waiting cost for a crew at any location and thus make it costlier to wait longer. This forces the crew to depart from a station as early as possible. The authors prove that this results in an optimal solution when there is no priority in assigning crews to trains. In case of priorities this method provides a FIFO infeasible solution. This paper presents another important application of the MCNF-FIFO model in real–world operations.

Chen and Chin (1992) study the multicommodity flow problem with another additional parameter associated with the arcs along with cost and capacity. This is the

safety parameter, which defines the relative degree of difficulty to pass through an arc. In this model, a commodity has a threshold of difficulty it can withstand in its transit from its origin to destination and these safety considerations form the side constraints. The authors transform this problem to the resource-capacitated problem introduced by Wollmer. Chen and Chin consider the set of arcs that define a path, and the maximum difficulty level from this set defines the safety parameter of the path. The authors apply similar solution techniques as Wollmer to solve the problem.

A similar problem with side constraints on paths is introduced by Holmberg and Yuan (2003), in the context of communication networks. In this model a commodity represents a communication pair with a certain demand of telecommunication traffic. The authors extend the basic MCNF model to include real world issues such as delay and reliability while choosing the paths to travel from origin to destination of commodities. Holmberg and Yuan include these as side constraints on the paths. When the capacity constraints are relaxed in this model, it decomposes into a constrained shortest path problem for each commodity. The authors propose a column generation based method to solve the problem efficiently, where the column generation sub-problem is a constrained shortest path problem.

Railway operations present many other challenging problems that have similar underlying network structures as our problem. Jha et al. (2008) solve the train make-up problem that deals with finding the optimal assignments of a group of shipments (blocks) to trains so as to minimize the total operational cost. The authors provide arc-based and path-based formulations for the problem and discuss an effective heuristic method called very large-scale neighborhood search (VLSN) to solve the problem quickly and effectively. Another problem that Kwon (1998) discusses deals with traffic routing in rail networks. The objective is to route the shipments from origin to destination and a pre-

defined train schedule constitutes the time-space network. Kwon deals with the problem from a service quality point of view, and adds service quality requirements as side constraints to the multicommodity network flow model. The author provides a column generation based approach to solve the problem. One of the major difficulties in routing traffic in rail networks is its dependence on the yard operations. Every yard has an established infrastructure and there is a sequence in which the incoming blocks are dismantled and the cars are switched. Crane et al. (1955) provide a lucid explanation of the classification operations in a railroad yard and explain the sequence in which switching operations are carried out in a yard. The policies, which define these operations, are the FIFO handling policies we discuss in this thesis. None of the models described above handle this kind of constraints.

The review of the literature shows that multicommodity network flows on a time-space networks are quite popular to model real-world applications. These are challenging problems to solve, and numerous methods have been developed to obtain an effective and practical solution to such problems. That said, not much research has been done in the area of FIFO or any handling policies at the intermediate or transshipment locations. One of the primary reasons is the difficulty of modeling such constraints mathematically and obtaining an optimal solution in real time. Moreover, many of the successful strategies to solve MCNF problems such as column generation and Lagrangian relaxation are not particularly effective in presence of FIFO constraints because of the complicated structure of the sub-problems.

Chapter 4: Models for multicommodity flow problems with FIFO constraints

In this chapter, we discuss alternative mathematical formulations to the formulation we present in Chapter 2 for the MCNF-FIFO problem. The primary objective is to develop formulations that have fewer constraints, provide tight LP relaxations, and have sparse constraint matrices. In general, there is a trade-off between these objectives and it is difficult to develop a formulation that excels simultaneously in all three criteria. The alternative formulations and enhancements we discuss in this chapter concentrate mainly on the FIFO and the flow conservation constraints. In the first section, we discuss techniques to classify the FIFO constraints into different "types" by exploiting the relationships between different commodities. We then classify the commodity-pair FIFO constraints presented in chapter 2 into these "types" and develop specialized constraints for each of them, leading to significant reduction in the number of constraints. In the next two sections, we discuss two alternative formulations for the FIFO constraints called the latest departure time and the arc-connection FIFO formulations. Finally, we present two alternative formulations for the flow conservation constraints, namely the consolidated and cumulative inflow-outflow flow conservation formulations. At each stage we discuss the merits and demerits of the alternative formulations by comparing them on the basis of the three criteria that we discuss above. We present the empirical results on the performance of these alternative models in Chapter 6.

4.1 SPECIALIZED VERSIONS OF FIFO CONSTRAINTS

The commodity-pair FIFO formulation creates a constraint for each pair of commodities and every pair of arrival and departure arcs. Since even a small instance of a real world problem may contain thousands of commodities and arcs, this formulation can create millions of FIFO constraints. To reduce the number of constraints, we develop specialized FIFO formulations by exploiting the information about the arrival times of two commodities with respect to each other. We next discuss the classifications of FIFO constraints.

4.1.1 FIFO constraint classifications

- FIFO Type I constraints: A FIFO constraint between any two commodities at a location *l*, in which both the commodities originate at *l* is a FIFO Type I constraint. In Figure 4(*a*), commodities *k* and *k*' originate at location *l* and hence the constraint between them is a FIFO Type I constraint. Since both commodities originate at the same location, we specifically know their arrival times (and order) at this location. Thus, to avoid a FIFO violation, we only need to control their departures with respect to each other. This information about their arrival times can be used to reduce the number of constraints as we discuss in the next section.
- FIFO Type II constraints: Any FIFO constraint at a location *l* in which one commodity is originating and the other commodity is a pass-though at *l*, and the earliest possible arrival time of the pass-through commodity is "after" the arrival time of the originating commodity is a FIFO Type II constraint.

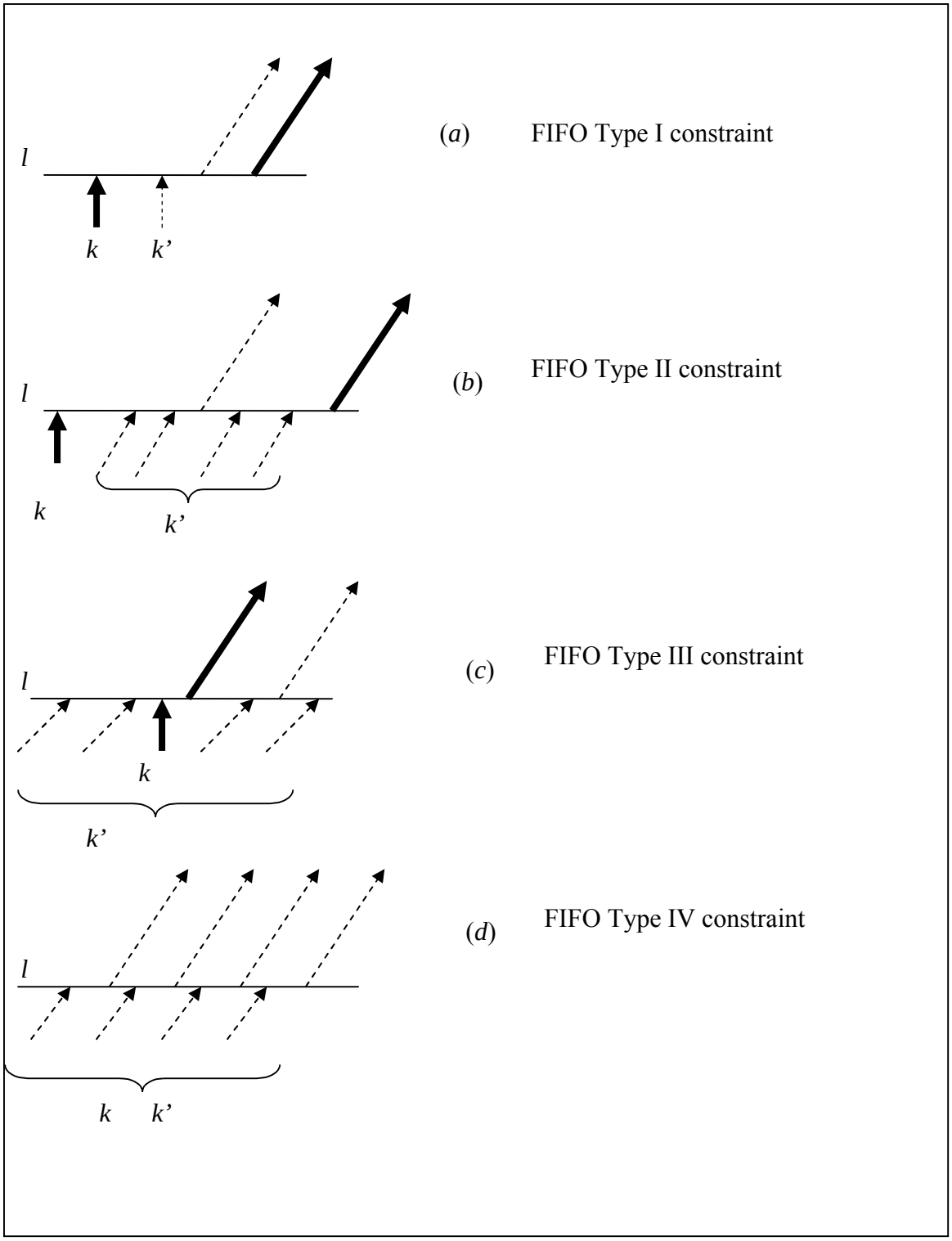

Figure 4: FIFO constraint classifications

In Figure 4(*b*), *k* is an originating commodity and *k*' is a pass-through commodity at location *l*. The commodity *k*' can arrive through any of the dashed arcs, but the earliest possible arrival time of *k*' is after the arrival time of *k*. So, in this case, even if we are not sure when *k*' actually arrives at *l*, we know that *k*' arrives after *k*, and hence it should depart with or after *k*. This is similar to FIFO Type I constraints, because we are sure about the order of departure of the two commodities.

- FIFO Type III constraints: Any FIFO constraint at a location *l* in which one commodity is originating and the other commodity is a pass-though at *l*, and the earliest possible arrival time of the pass-through commodity is "before" the arrival time of the originating commodity is a FIFO Type III constraint. In other words, we are not sure of the arrival time of the pass-through commodity with respect to the originating commodity in this case. In Figure 4(*c*), commodity *k* is the originating commodity and *k'* is the pass-through commodity, and it can arrive anytime with respect to *k*.
- FIFO Type IV constraints: This is the most general type of FIFO constraint in which both commodities are pass-through at the location at which we create the constraint. Figure 4(*d*) shows an example of this type of situation. As both *k* and *k*' are pass-through at location *l*, we are not sure of their arrival times with respect to each other and hence cannot reduce the number of FIFO constraints among these commodities.

In the next section, we classify the commodity-pair FIFO constraints according to these types and provide specialized versions of FIFO constraints for each type. Then we discuss, through an example, how this classification reduces the number of constraints.

4.1.2 Specialized versions of commodity-pair FIFO constraints

Commodity-pair FIFO Type I constraints:

These constraints are created only between commodities that originate at the same location. We define some additional notation before formulating the specialized FIFO Type I constraints. For each location l, $K^o(l)$ denotes the set of commodities originating at l and $TA(l)$ is the set of origin arrival times of commodities in $K^o(l)$. Multiple commodities may originate at the same time at a location l; let $K_a^o(l)$ denote the set of commodities originating at the same time $a \in TA(l)$.

Additional Notation	
$K^o(l)$	Set of commodities originating at location $l \in L$
$TA(l)$	Set of arrival times of originating commodities $K^o(l)$, at any location $l \in L$
$K_a^o(l)$	Set of commodities at location $l \in L$ originating at time $a \in TA(l)$

Table 3: Additional notation for commodity-pair based FIFO Type I constraints

With the additional notation in Table 3, we form the FIFO Type I constraints as

$$\sum_{\substack{e' \in AS(k,l) \\ : ts_{e'} \geq ts_e}} x_{ke'} + \sum_{\substack{f' \in AS(k',l) \\ : ts_{f'} < ts_e}} x_{k'f'} \leq 1 \quad \text{for each } l \in L,\ a, a+1 \in TA(l), k \in K_a^o(l), k' \in K_{a+1}^o(l),$$

$$e \in AS(k,l),\ f \in AS(k',l) \qquad \ldots (9)$$

Constraint (9) states that if any commodity originates at location l at time a and departs through any arc e, then any other commodity k' originating at l at time a+1, cannot depart using any arc f departing before e. Note that in this formulation, we are only creating constraints among commodity pairs originating at consecutive arrival time points. In Figure 5, there are two arrival points a and a+1, with the set of commodities $K_a^o(l)$ originating at time a and $K_{a+1}^o(l)$ originating at time a+1. Constraints (9) only

form the constraints among these pairs of commodities and each feasible departure arc for each consecutive arrival time point pair $a, a+1 \in TA(l)$.

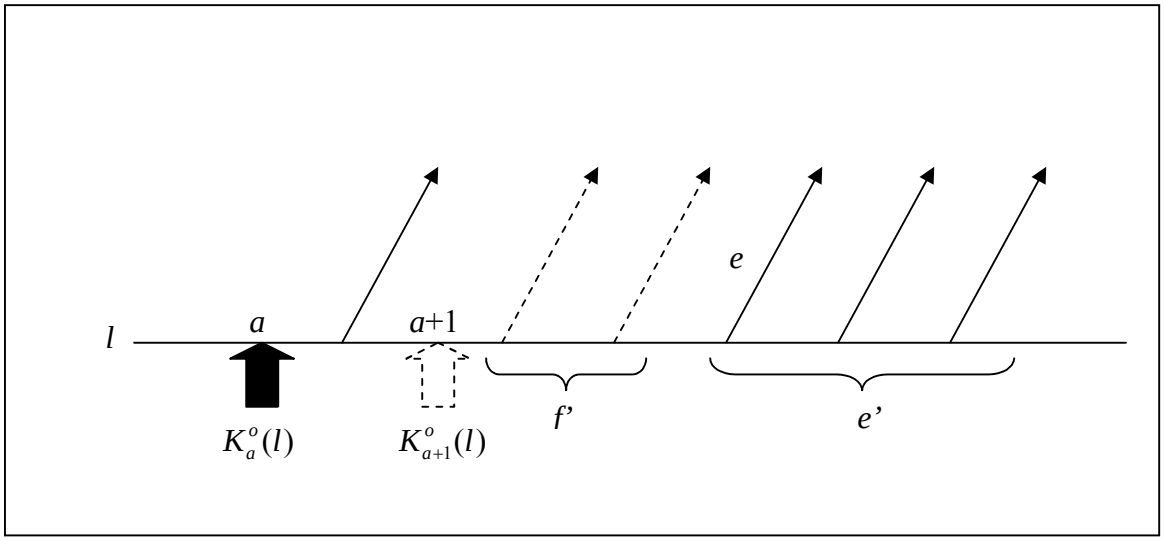

Figure 5: Commodity-pair FIFO Type I example

By comparing (9) with the general version (7) presented in Chapter 2, we can see that this formulation not only reduces the number of variables (connection variables) required to form constraints among originating commodities, but also reduces the number of constraints. In constraints (7), we form constraints among each pair of commodities arriving at each distinct arrival time point, while in the above formulation, it is sufficient to impose the FIFO relation only among the pair of commodities originating at consecutive time points. For example, if we impose FIFO constraints between commodities arriving at time point *a* and *a*+1, then they force each commodity arriving at *a*+1 to depart with or after the commodities arriving at *a*. Similarly, the constraints among the commodities arriving at *a*+1 and *a*+2 make sure that the commodities arriving at *a*+2 depart with or after the commodities arriving at *a*+1 and hence *a*. Thus, this eliminates the need to include constraints for every pair of originating commodities. This

reduction in the number of constraints is possible mainly because of the prior information about the arrival times of the commodities with respect to each other in case of FIFO Type I constraints.

Commodity-pair FIFO Type II constraints:

In these constraints, commodity *k* originates at location *l* and *k*' is a pass-through commodity at *l*, but we know a priori that *k*' can only arrive at *l* after *k*. Thus, even if we are not sure about the exact arrival time of *k*', we are sure that it has to depart with or after *k*. This scenario is similar to the FIFO Type I constraints, since we only need to control the departures of the commodities from *l*, so that they follow the FIFO order. At each location *l*, we denote the set of pass-through commodities at *l* as $K^{pt}(l)$. For each pass-through commodity $k \in K^{pt}(l)$, we store their earliest possible arrival time at *l* in the set $ETA(k,l)$.

Additional notation

$K^{pt}(l)$	Set of pass-through commodities at location $l \in L$
$ETA(k,l)$	Earliest arrival time of commodity $k \in K^{pt}(l)$ at location $l \in L$

Table 4: Commodity-pair FIFO Type II notation

With the additional notation defined above, we formulate the specialized FIFO Type II constraints as:

$$\sum_{\substack{e' \in AS(k,l) \\ :ts_{e'} \geq ts_e}} x_{e'}^{k} + \sum_{\substack{f' \in AS(k',l) \\ :ts_{f'} < ts_e}} x_{f'}^{k'} \leq 1 \qquad \text{for each } l \in L, k \in K^{o}(l), k' \in K^{pt}(l), EAT(k',l) > ta_k \quad \ldots(10)$$

Constraints (10) state that any pass-through commodity *k*' whose earliest arrival time at *l* is after the arrival time of an originating commodity *k* must depart with or after *k*. In Figure 6, *k* originates at location *l* at time *a*, and *k'* is a pass-through commodity

that can arrive at *l* using any of the *p* arcs that all arrive after time point *a*. Constraint (10) forces every such commodities *k*' to depart with or after *k* from location *l*.

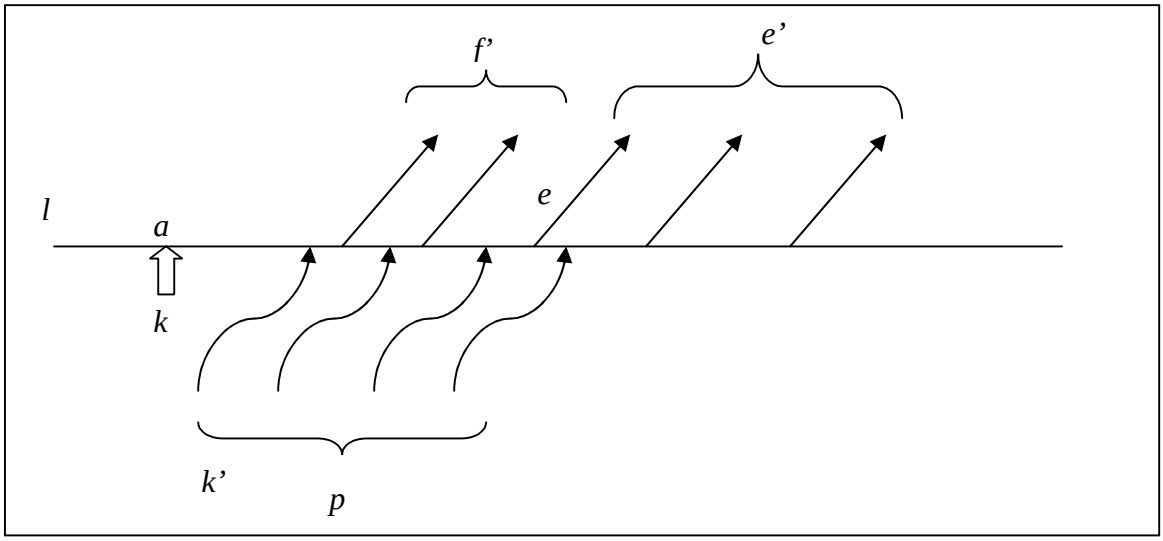

Figure 6: Commodity-pair FIFO type II example

In this case also, we can reduce the number of constraints by considering some special relations. Consider a situation where we have a set of commodities originating at location *l*, where t_e is the arrival time of the earliest originating commodity and t_u is the arrival time of the latest originating commodity in this set. If no pass-through commodity's earliest arrival time at location *l* is between t_e and t_u, we call this set of

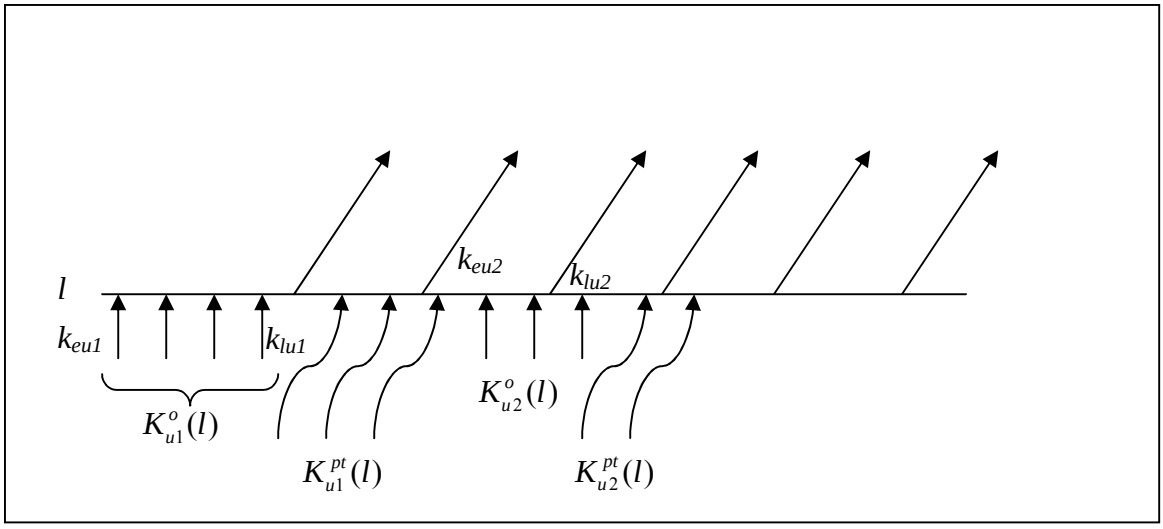

Figure 7: Commodity-pair FIFO Type II reduction example

originating commodities as the set of "*consecutive originating*" commodities. In general, there can be many such sets of "*consecutive originating*" commodities at location *l*. Let *U*(*l*) denote the number of such sets at *l*. The set $K^o_u(l)$ consists of the commodities in any "*consecutive originating*" set $u \in \{1,\ldots,U(l)\}$. The latest arriving commodity in $K^o_u(l)$ is denoted by k_{lu} and the earliest arriving commodity is denoted by k_{eu}. Finally, $K^{pt}_u(l)$ denotes the set of pass-through commodities whose earliest arrival times at location *l* lie between the arrival times of $k_{lu} \in K^o_u(l)$ and $k_{eu} \in K^o_{u+1}(l)$. In Figure 7, *l* has two "consecutive originating" sets, i.e., *U*(*l*)=2. The set $K^o_{u1}(l)$ consists of originating commodities corresponding to *u*1, and $K^o_{u2}(l)$ consists of originating commodities corresponding to *u*2. Finally, $K^{pt}_{u1}(l)$ consists of pass-through commodities whose earliest arrival times at *l* are after the latest arrival in set $K^o_{u1}(l)$ and before the earliest arrival in $K^o_{u2}(l)$.

In this case, it is sufficient to form FIFO Type II constraints between the latest arriving commodity k_{lu} in the set $K^o_u(l)$ and the commodities in $K^{pt}_u(l)$ only, i.e., we do not need to form constraints between each pair of commodities in $K^o_u(l)$ and $K^{pt}_u(l)$. This

is true because the FIFO Type I constraints between k_{lu} and the earlier originating commodities in $K_u^o(l)$ automatically impose the FIFO Type II constraints between all commodities in $K_u^o(l) \setminus \{k_{lu}\}$ and $K_u^{pt}(l)$. Table 5 shows the additional notation for the modified constraints (11).

Additional notation

$U(l)$	Number of "*consecutive originating*" commodity sets at location l
$K_u^o(l)$	Set of commodities in the *uth* "*consecutive originating*" commodities set at location l
k_{lu}	Latest arriving commodity in set $K_u^o(l)$
k_{eu}	Earliest arriving commodity in set $K_u^o(l)$
$K_u^{pt}(l)$	Set of pass-through commodities at location l whose earliest arrival time at l is between the latest arriving commodity in $K_u^o(l)$, and the earliest arriving commodity in $K_{u+1}^o(l)$

Table 5: Commodity-pair FIFO Type II notation - extended

The modified version of the FIFO Type II constraints is:

$$\sum_{\substack{e' \in AS(k_{lu}, l) \\ : ts_{e'} \geq ts_e}} x_{e'}^{k_{lu}} + \sum_{\substack{f' \in AS(k', l) \\ : ts_{f'} < ts_e}} x_{f'}^{k'} \leq 1 \quad \text{for each} \quad \begin{matrix} l \in L, u \in U(l), k' \in K_u^{pt}(l), k_{lu} \in K_u^o(l) \\ e \in AS(k_{ul}, l) \end{matrix} \quad \ldots(11)$$

This modification can lead to a significant reduction in the number of constraints if there are many sets of "*consecutive originating*" commodities at a location. For the example in Figure 7, the number of Type II constraints by using the formulation in constraint (10) is nearly 60 whereas only about 20 constraints (11) are needed.

Commodity-pair FIFO Type III constraints:

In these constraints, we are not sure if the pass-through commodity *k* arrives at location *l* before or after the originating commodity *k'*. This case is similar to the most general case, and hence we need to form some indicator variables to capture the arrival and departure of the pass-through commodity. As we discuss in Chapter 2, the indicator variable y_{ef}^{k} is 1 if the commodity *k* enters location *l* through arc *e* and departs using arc *f*, otherwise 0. The FIFO Type III constraints are:

$$x_e^k + x_f^k - y_{ef}^k \le 1 \qquad \text{for each } l \in L, k \in K^{pt}(l), e \in AE(k,l), f \in AS(k,l) \quad \ldots(12)$$
$$x_e^k + x_f^k - 2y_{ef}^k \ge 0$$

$$\sum_{\substack{p' \in AS(k,l) \\ :ts_{p'} \ge ts_p}} x_{p'}^{k'} + \sum_{\substack{e' \in AE(k',l), \\ f' \in AS(k',l) \\ :ts_{f'} \ge ts_f, te_e \le te_{e'} \le te_f}} y_{e'f'}^{k} \le 1 \quad \text{for each } l \in L, k \in K^{pt}(l), k' \in K^{o}(l), e \in AE(k,l),$$

$$f \in AS(k,l), p \in AS(k',l), te_e > ta_{k'} \text{ and } ts_f > ts_p \qquad \ldots(13)$$

In Figure 8, the pass-through commodity *k* can arrive using any of the arcs *e*; hence, its arrival time at location *l* determines whether *k* departs at or after *k'* when following the FIFO order. Constraints (12) set the indicator variables to capture the arrival and departure times of commodity *k*, and constraints (13) enforce the FIFO rule by forcing *k* to depart with or after *k'* if it arrives after the arrival of *k'*.

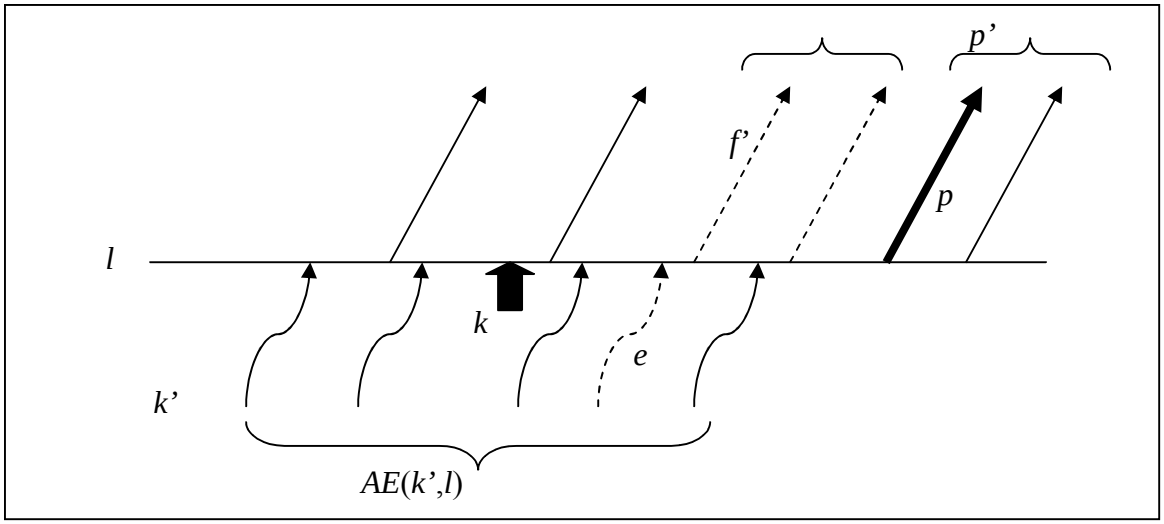

Figure 8: Commodity-pair FIFO Type III example

Commodity-pair FIFO Type IV constraints:

These are the most general type of FIFO specialized constraints since we have no a priori information about the relative arrival time order for commodities *k* and *k*'. Thus the needed FIFO constraints are similar to constraints (7) with the only difference being that they are created only between pairs of pass-through commodities at a location. The FIFO Type IV constraints are formulated as:

$$x_e^k + x_f^k - y_{ef}^k \le 1 \qquad \text{for each } l \in L, k \in K^{pt}(l), e \in AE(k,l),\ f \in AS(k,l) \qquad \ldots(14)$$
$$x_e^k + x_f^k - 2y_{ef}^k \ge 0$$

$$\sum_{\substack{f' \in AS(k,l) \\ : ts_{f'} \ge ts_f}} y_{ef'}^k + \sum_{\substack{q \in AS(k',l) \\ : ts_q < ts_f}} y_{pq}^{k'} \le 1 \quad \text{for each } \begin{array}{l} l \in L,\ k,k' \in K^{pt}(l), e \in AE(k,l), \\ p \in AE(k',l) \text{ and } te_p > te_e \end{array} \qquad \ldots (15)$$

Figure 9 presents a detailed example to understand the reduction in the number of constraints by using the specialized versions of FIFO constraints, instead of using the general version.

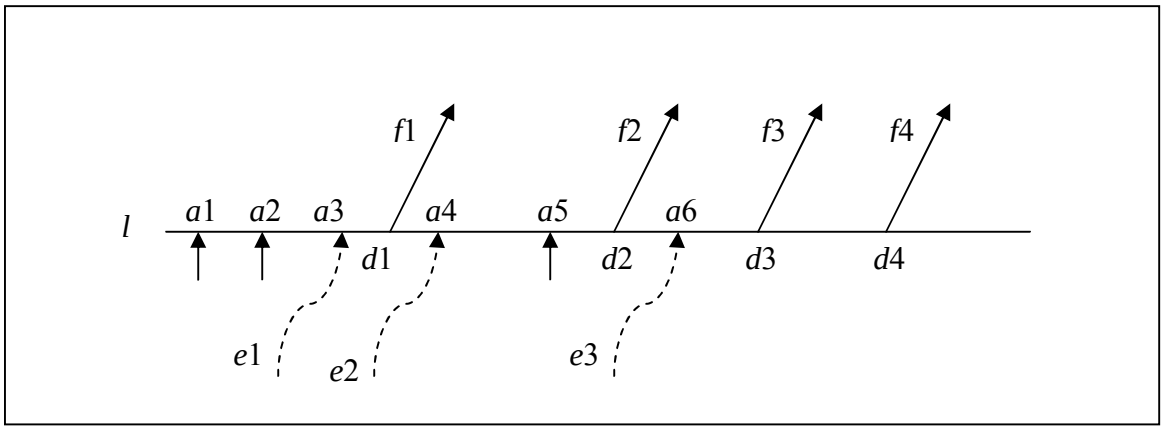

Figure 9: Specialized FIFO constraint – number of constraints reduction

In the example of Figure 9, suppose commodity $k1$ originates at time $a1$, $k2$ originates at time $a2$ and $k5$ originates at time $a5$ at location l. $k3$, $k4$ and $k6$ are pass-through commodities at l, with their earliest arrival times as $a3$, $a4$ and $a6$ respectively. Commodity $k3$ can arrive through any of the arcs $e1$-$e3$, commodity $k4$ can arrive through $e2$ and $e3$ and $k6$ can arrive only through $e3$. If we formulate the FIFO constraints using the general commodity-pair FIFO formulation (constraints (6) and constraints (7)), then we have 27 variables and nearly 50 constraints. On the other hand, if we use the specialized versions of the FIFO constraints, we have 9 FIFO Type I constraints, 10 FIFO Type II constraints, 10 FIFO Type III constraints and 8 FIFO Type IV constraints, yielding a total of 37 FIFO constraints. Thus, even for this small example, the specialized versions lead to a significant reduction in the number of constraints. The extent of reduction in number of constraints increases as the number of commodities and the number of incoming and outgoing arcs increase.

Although the commodity-pair FIFO formulation is quite tight, the number of FIFO constraints can be large, thus making it computationally inefficient for practical applications. In the next section we discuss another alternative formulation for the FIFO constraints which further reduces the number of constraints significantly.

4.2 LATEST DEPARTURE TIME FIFO FORMULATION

The commodity-pair FIFO formulation creates constraints to prevent assigning incompatible pairs of commodities to movement arcs. Another way to visualize the FIFO constraints is in terms of the "*latest departure times*" from each arrival time point at every location. The basic idea is to capture the latest departure time say q_a^l among all the commodities that can arrive at time *a* at a location *l*. Then any commodity arriving at *l* after this time point *a* must depart at or after q_a^l. Below, we provide the formulations for the latest departure time FIFO constraints and explain them using an example:

Latest departure time:

$$q_a^l \geq \sum_{p' \in AS(k,l)} ts_{p'} \left(\sum_{\substack{e \in AE(k,l) \\ :te_e = a}} x_e^k + x_{p'}^k - 1 \right) \text{ , for each } l \in L, a \in TA(l), k \in K(l,a) \qquad \ldots(16)$$

FIFO:

$$\sum_{p \in AS(k',l)} ts_p x_p^{k'} \geq \left(1 - \sum_{\substack{e \in AE(k',l) \\ :te_e \leq a}} x_e^{k'} \right) q_a^l \text{ , for each } l \in L,\ a, a+1 \in TA(l), k' \in K(l,a+1) \ldots(17)$$

Constraints (16) set the latest departure time from each arrival time point at each location. The constraint checks whether a commodity *k* arrives at the time point *a*; if it does, the constraint sets the latest departure time from *a* equal to the departure time of *k*,

if k departs after current q_a^l. Constraints (17) impose the FIFO constraints by forcing each commodity that can arrive at time point a+1 to depart after q_a^l.

This formulation reduces the number of constraints significantly compared to the commodity-pair FIFO formulation. The primary reason is that this formulation does not include constraints for every pair of commodities and every pair of arriving and departing arcs. We explain the reduction in number of constraints through the example in Figure 10.

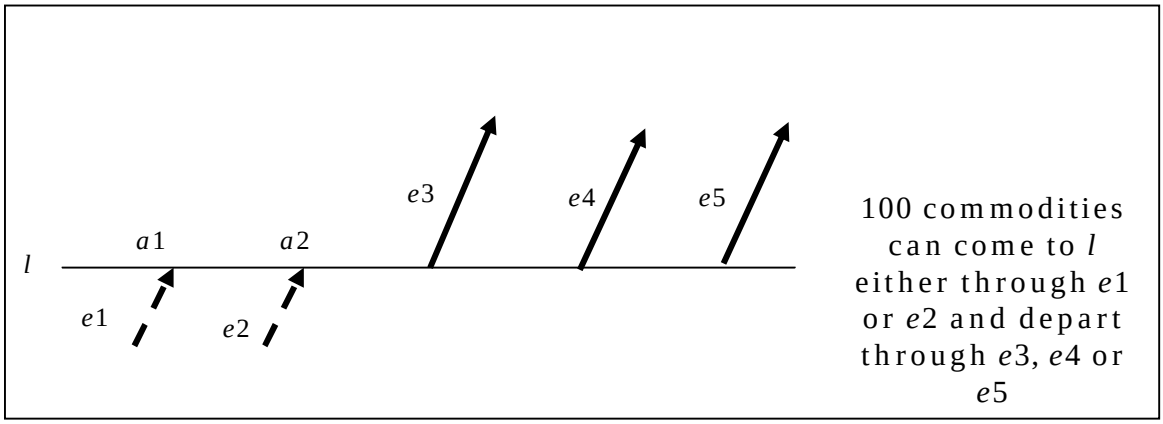

Figure 10: Latest departure time FIFO formulation – reduction in constraints

In this example, 100 commodities can arrive at location l either through arc $e1$ and $e2$ and depart through any of the departure arcs. We have two arrival time points, $a1$ and $a2$ corresponding to arcs $e1$ and $e2$ respectively. Even in this simple example, the commodity-pair FIFO formulation creates nearly 20,000 constraints while the latest departure time FIFO formulation creates only 300 constraints. Thus, we achieve a significant reduction in the number of constraints using this alternative formulation of the FIFO constraints.

Although this formulation performs well in reducing the number of constraints, the LP relaxation of this formulation is not as tight as the commodity-pair FIFO formulation.

***Proposition* 4.1:** The commodity-pair FIFO formulation is tighter than the latest departure time FIFO formulation.

Proof: For any two constraint systems $Ax \le b$ and $A'x \le b'$, we say they are *equivalent* if they admit exactly the same set of 0-1 solutions. The system $A'x \le b'$ is said to be *as tight as* system $Ax \le b$ whenever

$$\{x \in [0,1]^n \mid A'x \le b'\} \subseteq \{x \in [0,1]^n \mid Ax \le b\}. \quad \ldots(18)$$

We say that $A'x \le b'$ is *tight* if the containment in (18) is strict (Escudero & Munoz 1998). We prove the proposition by proving that some feasible solutions to the LP relaxation of the latest departure time FIFO formulation are not feasible for the commodity-pair FIFO formulation while, every feasible solution for commodity-pair FIFO is feasible for the latest departure time FIFO formulation. Let us consider the example in Figure 11, where a commodity *k* arrives at location *l* at time *a*1 through arc *e*1 and commodity *k*' arrives at *l* at time *a*2 > *a*1, through arc *e*2. The departure times of arcs *e*3, *e*4 and *e*5 is the corresponding departure order from the location *l*. Assume that 0.5 of commodity *k* departs from *l* through arc *e*3, 0.25 departs through arc *e*4 and 0.25 departs through arc *e*5 (i.e., $x^k_{e3} = 0.5, x^k_{e4} = 0.25, x^k_{e5} = 0.5$), then the corresponding value of $q^l_a \ge 1.75$. Similarly if 0.25 of commodity *k*' departs through arc *e*3 and 0.75 departs through arc *e*4 (i.e., $x^{k'}_{e3} = 0.25, x^{k'}_{e4} = 0.75$), then $q^l_{a+1} \ge 1.75$ and these assignments satisfy the FIFO constraint (17).

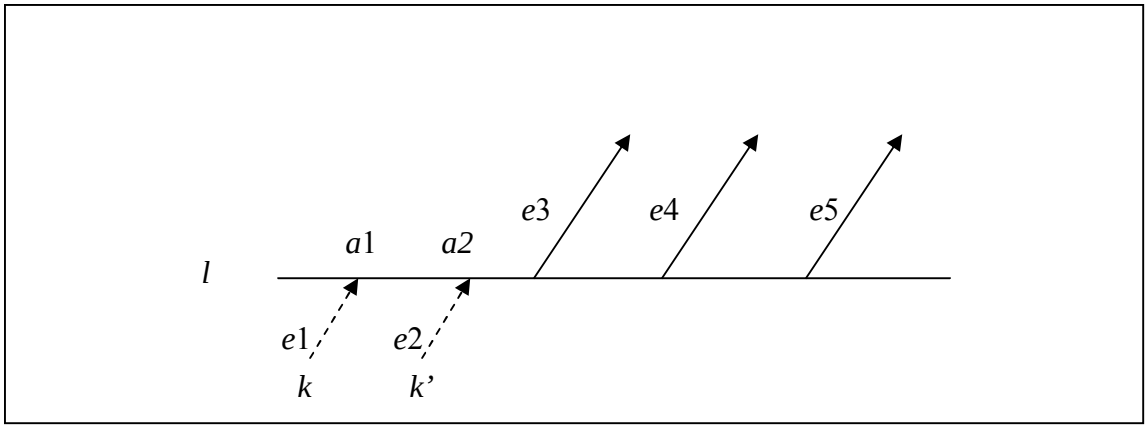

Figure 11: Latest departure time FIFO formulation – LP relaxation strength comparison

If we substitute the same assignments into the commodity-pair FIFO formulation, we get $y^k_{e1e3} \geq 0.5, y^k_{e1e4} \geq 0.25, y^k_{e1e5} \geq 0.25$ and $y^{k'}_{e2e3} \geq 0.25, y^{k'}_{e2e4} \geq 0.75$, by constraint (6a) and (6b). Clearly, these connection variables violate the FIFO constraint (7) and hence this set of assignments is infeasible for the commodity-pair FIFO formulation. On the other hand, any feasible solution for the commodity-pair FIFO formulation (for example, $x^k_{e1} = 1, x^k_{e3} = 0.5, x^k_{e4} = 0.5$ and $x^{k'}_{e2} = 1, x^{k'}_{e3} = 0.5, x^{k'}_{e4} = 0.5$) is also feasible for the latest departure time FIFO formulation. Thus, the feasible region of the commodity-pair FIFO formulation is smaller than the feasible region of the latest departure time FIFO formulation, which makes it a tighter formulation. Hence, even if we get a significant reduction in the number of constraints by this formulation, it may not be computationally efficient to obtain an optimal solution for the problem quickly.

■

Please note that we can also classify the latest departure time FIFO formulation into specialized types as we discuss in Section 4.1. For brevity, we do not provide the specialized formulations for each type here. In the next section, we discuss another alternative formulation for the FIFO model that uses similar concepts as the latest

departure time formulation to reduce the number of constraints, and is stronger than the commodity-pair FIFO formulation.

4.3 ARC-CONNECTION FIFO FORMULATION

In this formulation, we form FIFO constraints for each pair of arrival and departure time points at each location. We introduce a new indicator variable u_{ad}^{l} that is set to 1 if any commodity k arrives at location l at or before time point a and departs at or after time d. The sets $TA(l)$ and $TD(l)$ denote the set of arrival and departure time points at location l respectively. The set $K(a,l)$ denotes the set of commodities that can arrive at time point $a \in TA(l)$ at location l. We first create the connection constraints that set the variable u_{ad}^{l} by checking if any commodity in $K(a,l)$ arrives at or before a and departs at or after d. Then we have the FIFO constraints that force any commodity which arrives after a to depart at or after d if u_{ad}^{l} is set to 1.

4.3.1 Arc-connection FIFO constraints

ADDITIONAL NOTATION

$T(l)$	Set of arrival or departure time points at location l
$TA(l)$	Set of arrival time points at location l
$TD(l)$	Set of departure time points at location l
$K(a,l)$	Set of commodities which can arrive at location l at time point $a \in TA(l)$
u_{ad}^{l}	1 if any commodity arrives at location l at or before time point $a \in TA(l)$ and departs at or after departure time point $d \in TD(l)$

Table 6: Arc-connection FIFO constraint notation

With the above notation, we formulate the arc-connection FIFO constraints as follows:

Connection:

$$u_{ad}^{l} \geq \sum_{\substack{e \in AE(k,l) \\ :te_e = a}} x_e^k + \sum_{\substack{f \in AS(k,l) \\ :ts_f \geq d}} x_f^k \text{ -1 for each } l \in L, a \in TA(l), k \in K(a,l), d \in TD(l), d > a \quad \ldots(19)$$

$$u_{(a+1)d}^{l} \geq u_{ad}^{l} \quad \text{for each } l \in L,\ a, a+1 \in TA(l), d \in TD(l) \quad \ldots(20)$$

Arc-connection FIFO:

$$\sum_{\substack{e \in AE(k,l) \\ :te_e < a}} x_e^k + \sum_{\substack{f \in AS(k,l) \\ :ts_f \geq d}} x_f^k \geq u_{ad}^{l} \text{ for each } l \in L, a \in TA(l), k \in K(a,l), d \in TD(l), d > a \quad \ldots(21)$$

We explain the formulation through the example in Figure 12. In this example, the location *l* has two arrival time points *a* and *a*+1 and three departure time points *d*, *d*+1 and *d*+2. Assume that commodity *k*1 arrives at time *a* and departs at time *d*+1. Since commodity *k*1 arrives at or before *a* and departs at or after *d*+1, by definition, constraints (19) set the indicator variables u_{ad}^{l}=1 and $u_{a(d+1)}^{l}$=1. Constraints (20) enforce the precedence relationship between consecutive arrival time points. Since commodity *k*1 arrives at *a* and departs at *d*+1, the connection variables $u_{(a+1)d}^{l}$ and $u_{(a+1)(d+1)}^{l}$ are also equal to 1. Constraints (21) create a constraint for each commodity $k \in K(a,l)$ and for each pair of arrival departure time points at *l*. If the indicator variable $u_{ad}^{l} = 1$, then constraints (21) enforce that any commodity that can arrive at *a* **must** arrive at or before *a* or depart at **or** after *d*, in other words, no commodity can arrive after *a* **and** depart before *d* at location *l*. In the example, assume that commodity *k*2 arrives at time point *a*+1. Since $u_{(a+1)(d+1)}^{l}$=1, the constraint forces commodity *k*2 to depart at or after *d*+1, thus enforcing the FIFO order.

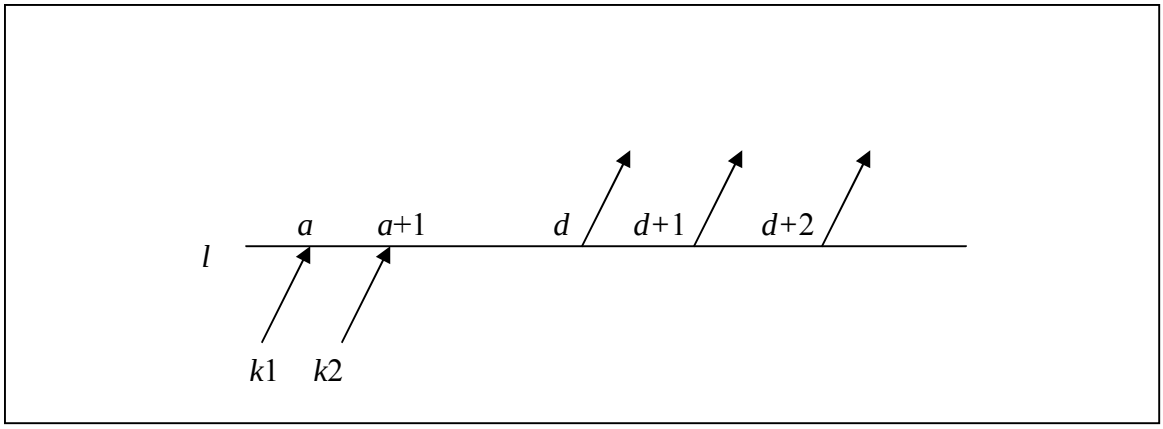

Figure 12: Arc-connection FIFO constraint example

The arc-connection FIFO formulation creates constraints only among each pair of arrival departure time points, and for each commodity that can arrive at a particular arrival point. Therefore, it contains much fewer constraints than the commodity-pair FIFO formulation. To understand the extent of reduction in the number of constraints, we take the example in Figure 13 in which there are two arcs arriving at location *l* and three arcs departing from *l*. Assume that 100 commodities can arrive at this location through either of the incoming arcs. For this simple example, the commodity-pair FIFO formulation includes nearly 20,000 constraints while the arc-connection FIFO formulation contains only around 900 constraints. Moreover, the arc-connection FIFO formulation also reduces the number of indicator variables. The commodity-pair FIFO formulation requires indicator variables for each commodity and each pair of arrival and departure arcs at each location, whereas the arc-connection formulation only requires indicator variables for each pair of arrival departure time points at each location, leading to significant reduction in the number of variables.

Furthermore, the arc-connection FIFO formulation is also tighter than both the commodity-pair and latest departure time FIFO formulations.

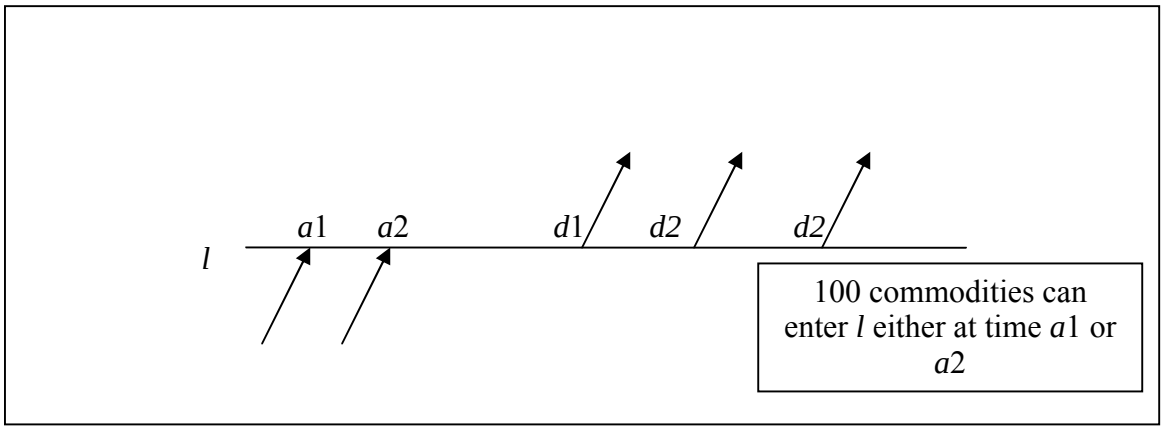

Figure 13: Reduction in number of constraints example

***Proposition* 4.2:** *The arc-connection FIFO formulation is tighter than the commodity-pair FIFO formulation and the latest departure time FIFO formulation.*

Proof: To prove this proposition we need to prove that every feasible solution to the LP relaxation of the arc-connection FIFO formulation is feasible to the LP relaxation of the commodity-pair FIFO formulation, but some feasible solutions to the LP relaxation of commodity-pair FIFO formulation are not feasible to LP relaxation of arc-connection FIFO formulation. Figure 14 shows a time-space network for location *l* with three incoming arcs and three outgoing arcs. Suppose commodity *k*1 arrives through arc *e*1 and departs through arc *f*2, and 0.3 of commodity *k*2 enters through *e*2 and 0.7 enters through arc *e*3. Thus, we have $x_{e1}^{k1}=1, x_{f2}^{k1}=1$, $x_{e2}^{k2}=0.3$ and $x_{e3}^{k2}=0.7$. If we take the commodity-pair FIFO formulation, the value of $y_{e1f2}^{k1}=1$, and the FIFO constraints set the value of $y_{e2f1}^{k2}=0$. In other words $x_{f2}^{k2}+x_{f3}^{k3}=1$, as *k*2 must depart either through arc *f*2 or *f*3. For the same example, the arc-connection FIFO formulation sets $u_{a1d1}^{l}=1$ and $u_{a1d2}^{l}=1$, which forces $x_{f1}^{k2}+x_{f2}^{k2}+x_{f3}^{k3}\geq 1$ and $x_{f2}^{k2}+x_{f3}^{k3}\geq 1$. Thus both the formulations force the commodity *k*2 to leave either through *f*2 or *f*3.

To prove that the feasible region of commodity-pair FIFO constraints is larger than the arc-pair FIFO constraints, we consider a more complicated example.

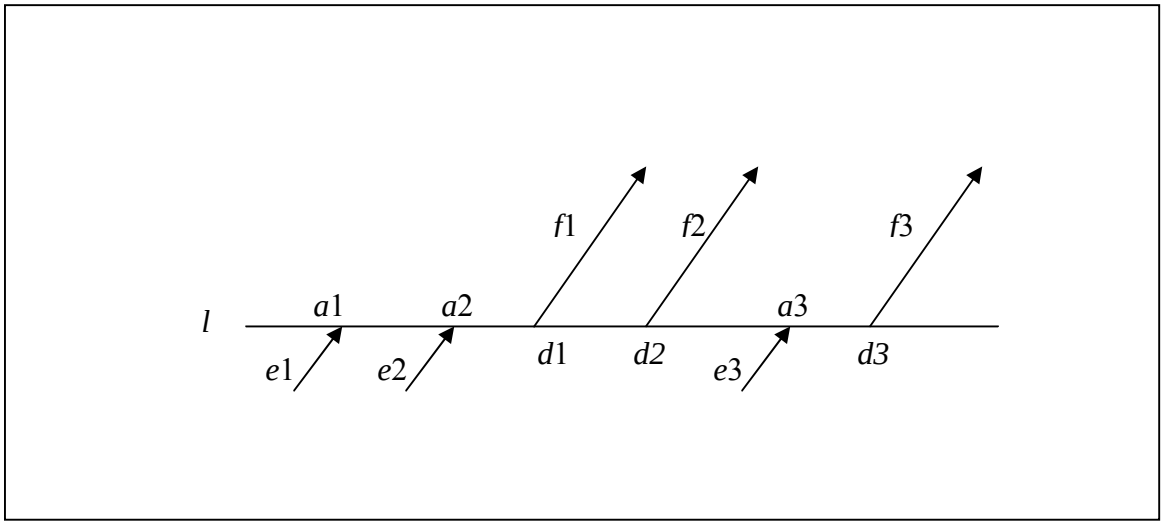

Figure 14: Example for Proposition 4.2

Suppose $x_{e1}^{k1}=0.75, x_{f2}^{k1}=0.5$, $x_{f3}^{k1}=0.25$ and $x_{e2}^{k2}=1$. For both formulations, we find the range of values that x_{f1}^{k} can take. The formulation in which the range of x_{f1}^{k} is smaller is a tighter formulation. Ideally, this variable should be zero, as any positive value causes FIFO violation but due to fractional assignments, a feasible LP solution may violate FIFO order.

For the commodity-pair FIFO formulation, we get the following range for the indicator variables based on the assignments above,

$$0 \le y_{e1f1}^{k1} \le 0.375$$

$$0 \le y_{e1f2}^{k1} \le 0.625$$

$$0 \le y_{e1f3}^{k1} \le 0.5.$$

Hence, the range of variable y_{e2f1}^{k2} from the FIFO constraints (7) is

$$0 \le y_{e1f3}^{k1} \le 0.75.$$

Thus, the range of values for the movement variable x_{f1}^{k} is $0 \le x_{f1}^{k2} \le 0.75$.

Now we take the case of arc-connection FIFO constraints. The indicator variables in this case are $u_{a1d1}^{l} \ge 0.5$, $u_{a1d2}^{l} \ge 0.5$ and $u_{a1d3}^{l} \ge 0$ from constraints (19). From constraint (20), the indicator variables $u_{a2d1}^{l} \ge 0.5$, $u_{a2d2}^{l} \ge 0.5$ and $u_{a2d3}^{l} \ge 0$. Thus from the FIFO constraints (21), we have

$$x_{f1}^{k2} + x_{f2}^{k2} + x_{f3}^{k2} \ge 0.5$$

$$x_{f2}^{k2} + x_{f3}^{k2} \ge 0.5.$$

and as the sum of the three assignment variables should be equal to 1, we get $0 \le x_{f1}^{k2} \le 0.5$, which is clearly lower than the allowed value in commodity-pair FIFO formulation. Hence, the arc-connection FIFO formulation is stronger than the commodity-pair FIFO formulation. By Proposition 4.1, the commodity-pair FIFO formulation is stronger than the latest departure time FIFO formulation; therefore, the arc-connection FIFO formulation is also stronger than latest departure time FIFO formulation.

■

4.3.2 Arc-connection FIFO formulation with inventory variables

The arc-connection FIFO formulation performs very well in terms of the number of constraints and strength of the LP relaxation, but the disadvantage of this formulation is its dense constraint matrix that makes it inefficient while using it in a commercial LP solver. As we discuss in the later section on computation results, the sparsity of the constraint matrix is another important factor in determining the performance of a model. We modify constraints (19) and (21) to make them sparser by replacing the summation of movement variables with inventory variables.

Connection:

$$u^l_{ad} \geq z^l_{kd} - \sum_{\substack{e \in AE(k,l) \\ :a < te_e \leq d}} x^k_e \quad \text{for each } l \in L, a \in TA(l), k \in K(a,l), d \in TD(l), d > a \qquad \ldots(22)$$

$$u^l_{(a+1)d} \geq u^l_{ad} \quad \text{for each } l \in L,\ a, a+1 \in TA(l), d \in TD(l) \qquad \ldots(23)$$

Arc-connection FIFO with inventory variables:

$$z^l_{kd} \geq u^l_{ad} - \sum_{\substack{e \in AE(k,l) \\ :te_e \leq a}} x^k_e - \sum_{\substack{e \in AE(k,l) \\ :te_e > d}} x^k_e \quad \text{for } l \in L, a \in TA(l), k \in K(a+1,l), d \in TD(l), d > a$$

$$\ldots(24)$$

The summation term in constraints (19), representing the departure of commodity *k* from location *l* at or after time point *d*, is represented in constraints (22) by the single inventory variable denoting the inventory of commodity *k* entering into time point *d*. Similarly, constraint (23) enforces that, for each commodity that can arrive at time *a*+1, the inventory entering into time *d* should be 1 if u^l_{ad} =1 provided it has not arrived before *a*+1 or after *d*. Thus, this formulation is equivalent to the arc-connection FIFO formulation in terms of the number of constraints and the LP relaxation strength, but has a much sparser constraint matrix. From Propositions 4.1 and 4.2, this formulation is also stronger than the commodity-pair and latest departure time FIFO formulations.

4.4 ALTERNATIVE MODELS FOR FLOW CONSERVATION CONSTRAINTS

In this section, we discuss two alternative formulations for the "simple" flow conservation constraints presented in Chapter 2.

4.4.1 Consolidated flow conservation formulation

In the "simple" flow conservation formulation, we create an inventory variable for each "event" at a location, i.e., for each arrival and departure time point at a location. Instead, we can consolidate consecutive arrival and departure time points into a single "event" and create a single inventory variable for that event. This consolidation leads to significant reduction in the number of inventory variables as well as the number of flow conservation constraints.

Figure 15 provides an example of this consolidation and the reduction obtained through it. As shown in the figure, we consolidate consecutive arrivals and departures

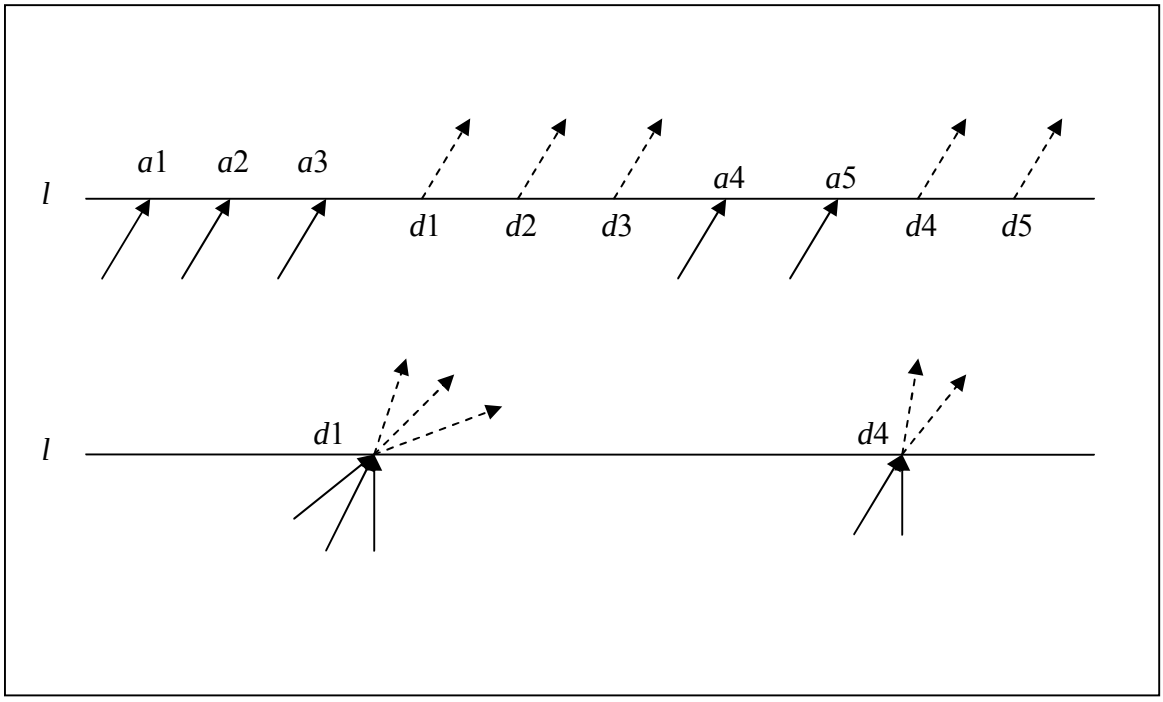

Figure 15: Consolidated flow conservation example

from a location l and create a "consolidated" event. We call the set of different "consolidated" events $T'(l)$ for any location l. Each consolidated event has a set of

consecutive arrival time points $TA'(t,l)$, and a set of consecutive departure time points $TD'(t,l)$. We only form inventory variables and constraints for each of the "consolidated" events instead of one for each individual event at a location, as we discuss next.

ADDITIONAL NOTATION

$T'(l)$	Set of "consolidated" time points after consolidating consecutive arrival and departure time points at location l
$TA'(t,l)$	Set of consecutive arrival time points in the "consolidated" time point $t \in T'(l)$ at location l
$TD'(t,l)$	Set of consecutive departure time points in the "consolidated" time point $t \in T'(l)$ at location l

Table 7: Additional notation for consolidated flow conservation constraints

Consolidated flow conservation constraints:

$$z_{kt}^{l} + \sum_{\substack{e \in AE(k,l) \\ :te_e \in TA'(t,l)}} x_e^k = z_{k(t+1)}^{l} + \sum_{\substack{f \in AS(k,l) \\ :ts_f \in TD'(t,l)}} x_f^k \quad \text{for } l \in L, k \in K(l), t \in T'(l) \qquad \ldots(25)$$

This formulation can reduce the number of flow conservation constraints drastically. For the example in Figure 17, the "simple" flow conservation formulation has 10 constraints while the consolidated flow conservation formulation has only 2 constraints.

Although, this formulation can reduce the number of constraints, it is also much denser than the "simple" flow conservation formulation, illustrating the trade-off between the different indicators of a good formulation.

4.4.2 Cumulative inflow-outflow flow conservation constraints

We can further reduce the model size by removing the inventory variables completely from the model. This model is even more attractive when we do not have any

cost for holding inventory, and the objective function is to minimize the total sum of delivery times of all the commodities at their destinations. The formulation for the cumulative inflow-outflow flow conservation constraints is:

$$\sum_{\substack{f' \in AS(k,l) \\ : ts_{f'} \geq ts_f}} x^k_{f'} \leq \sum_{\substack{e' \in AE(k,l) \\ : te_{e'} \leq ts_f}} x^k_{e'} \quad \text{for each } l \in L, k \in K(l), f \in AS(k,l). \qquad \ldots(26)$$

Constraints (26) ensure that a commodity leaves a location *l* on a departing arc $f \in AS(k,l)$, if and only if it reaches the location *l* on an arc before the departure time of *f*. The example in Figure 16 shows these constraints for a single commodity *k* at location *l*.

Note that in this case, we have formulated the constraints for each departing arc from location *l*, but we can consolidate the departing arcs as in Section 4.4.1 and create the constraints only for the consolidated departure time points. In the case of consolidation, the number of constraints is exactly the same as the number of constraints in the consolidated flow conservation formulation, and there are no inventory variables as well that further reduces the model size significantly.

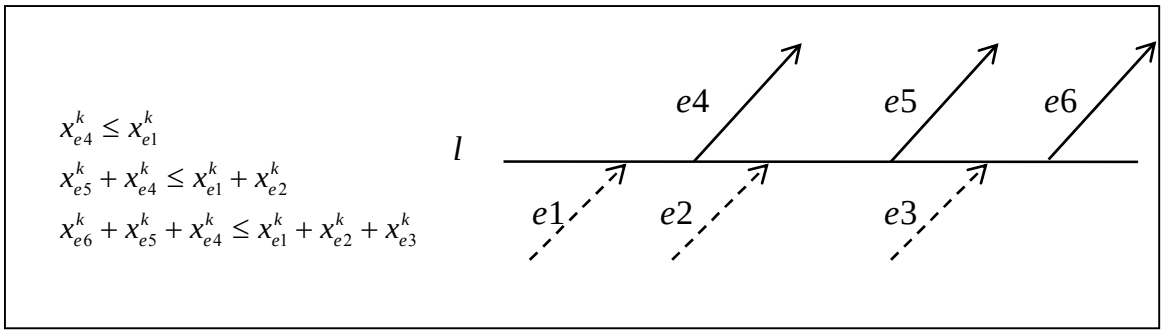

Figure 16: Cumulative inflow-outflow flow conservation example

Chapter 5: Algorithmic approaches

5.1 PROBLEM REDUCTION TECHNIQUES

In this section, we focus on techniques to reduce the problem size by eliminating variables and constraints before solving the IP model. The primary idea of the reduction techniques is to prove that a commodity *k* cannot be assigned to a particular arc in any optimal solution, and thus the corresponding movement variables can be removed a priori from the model.

We first describe some terms that we use below, and then discuss the problem reduction techniques in detail.

"*Uncapacitated*" resource: A resource is "*uncapacitated*" if the sum of the capacities required by all the commodities that can possibly use the resource is less than the available capacity of the resource. Thus, in any feasible solution, this resource always has some unused capacity. We refer to any resource that is not provably uncapacitated (using the above criterion) as a "*capacitated*" resource.

"*Uncapacitated*" arc: Any arc that uses only "*uncapacitated*" resources is an "*uncapacitated*" arc. If an arc uses at least one "*capacitated*" resource, then we refer to it as a "capacitated" arc.

"*Early originating*" commodities: Any set of commodities originating at a location *l*, whose arrival time is before the earliest possible arrival time of any pass-through commodity at *l* is called the set of "*early originating*" commodities.

"*Early departing*" arcs: Any set of arcs departing from a location, sorted in increasing order of departure time, whose total available capacity is less than the capacity required by the set of "*early originating*" commodities.

"*Crossing*" arc: Any arc *e*' which departs later than an arc *e* that has the same destination as *e*', but passes *e* to reach the destination earlier is called a "*crossing*" arc for *e*. Arc *e*' is also a "*crossing*" arc for any origin to destination path (O-D path) of a commodity that contains arc *e*.

***Proposition* 5.1**: In an optimal solution, a commodity cannot be assigned to any movement arc departing beyond the first "*uncapacitated*" origin to destination path, except if a later departing arc is a "*crossing*" arc.

Proof: In Figure 17, we have a commodity *k* with origin *l*1 and destination *l*3. The

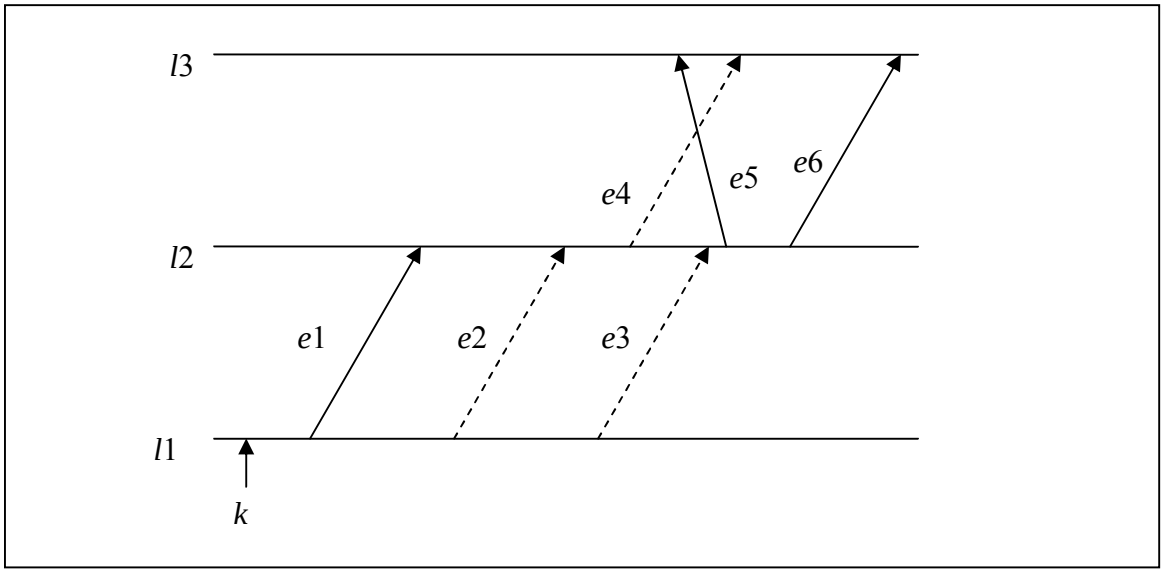

Figure 17: Problem reduction – Proposition 5.1 example

dashed arrows are the "*uncapacitated*" arcs and the solid arrows are the "*capacitated*" arcs. Thus, arcs *e*2 and *e*4 constitute the first "*uncapacitated*" O-D path for commodity *k*. Since we are minimizing the delivery time for each commodity, in the optimal solution, every commodity departs from each location at the earliest possible time, given enough available capacity. In Figure 19, arc *e*2 is "*uncapacitated*" and hence *k* always has enough capacity to depart location *l*1 using *e*2, and in any optimal solution it does not

wait until arc *e*3 departs. However, there is one exception to this rule, namely the "*crossing*" arcs. In the example, commodity *k* may wait until arc *e*5 at location *l*2 even though the first "*uncapacitated*" O-D path contains the arc *e*4. This is because the crossing arc *e*5 reaches its destination (or next intermediate location) earlier.

■

Thus, using Proposition 5.1, we can eliminate all the arc assignments for a commodity beyond the first "*uncapacitated*" O-D path except for the arcs that cross this path.

***Corollary* 5.1:** If the shortest origin to destination path for a commodity is "*uncapacitated*", it always travels on its shortest path in any optimal solution.

Proof: The proof follows directly from Proposition 5.1. If a commodity satisfies this corollary, then we can fix its route to its shortest path and eliminate all the corresponding assignment variables and constraints.

■

***Corollary* 5.2:** For any commodity *k*, if the shortest path from its origin to any intermediate location *l* is "*uncapacitated*", then in an optimal solution, it always travels on its shortest path up to location *l*.

Proof: The proof follows directly from Proposition 5.1. Since a commodity never departs beyond the first "*uncapacitated*" movement arc from any location (except in presence of a "*crossing*" arc), if the shortest path from origin to location *l* is "*uncapacitated*", it always takes the shortest path to the intermediate location *l*.

■

Thus, we can eliminate all the movement and inventory variables for this commodity at its origin and all other intermediate locations until location *l*. In other

words, we can consider commodity *k* to be originating from location *l*, at a time equal to its earliest possible arrival time at *l* (time to reach location *l* by its shortest path).

***Proposition* 5.2:** At any location *l*, no pass-through commodity can be assigned to the set of "*early departing*" arcs.

Proof: Figure 18 shows an example in which commodities *k*1 and *k*2 form the set of "*early originating*" commodities, and *k*3 and *k*4 are pass-through commodities at location *l*. Each of the arcs *e*1-*e*4 uses resources whose capacity is 1 unit. In this example, arcs *e*1 and *e*2 form the set of "*early departing*" arcs, since the total combined capacity of their available resources is not greater than the capacity required by the "*early originating*" commodities. To maintain the FIFO order of assignment, commodity *k*1 departs first, followed by *k*2, *k*3 and *k*4. Thus, *k*1 and *k*2 at least block the capacity of arcs *e*1 and *e*2, making them unavailable for *k*3 and *k*4. Note that we cannot fix the assignments of *k*1 and *k*2 to *e*1 and *e*2, since resources are shared across arcs (possibly originating from different locations). Thus, some other commodity from another location may use the capacity of resources belonging to *e*1 and *e*2, and delay the departure of *k*1 and *k*2 from *l*. But, we can eliminate the movement variables corresponding to the pass-through commodities at a location and the "*early departing*" arcs, thereby reducing the number of flow conservation constraints as well as FIFO constraints.

■

Based on the above propositions, we now describe the problem reduction algorithm that eliminates, a priori, the possible assignments of commodities to candidate movement arcs, and thus eliminates inventory variables, indicator variables, flow conservation constraints and FIFO constraints.

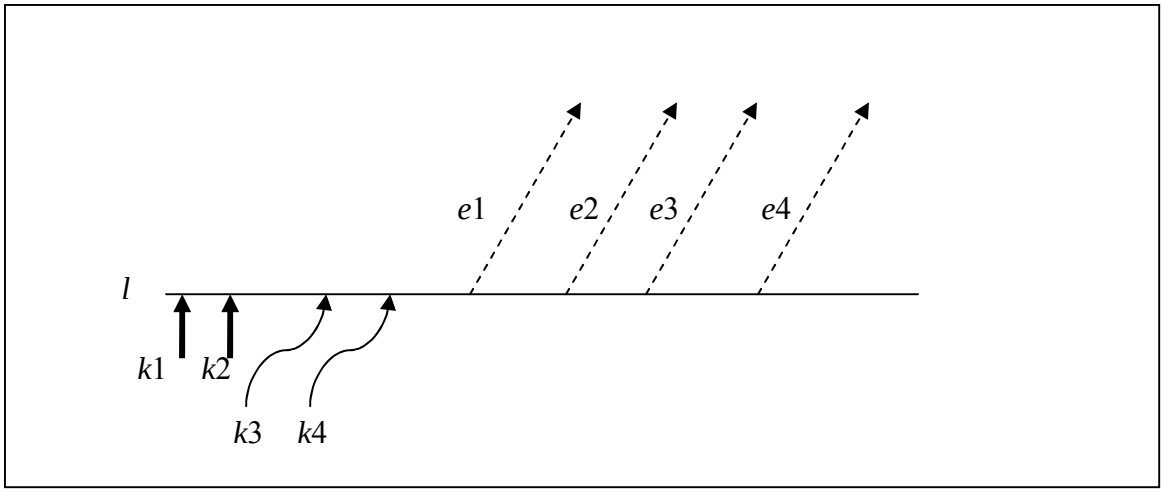

Figure 18: Example for Proposition 5.2

ADDITIONAL NOTATION

R_{uncap}	Set of "*uncapacitated*" resources in the model
A_{uncap}	Set of "*uncapacitated*" arcs in the model
$SP(k)$	Origin to destination shortest-path from each commodity $k \in K$
$SP_{uncap}(k)$	Origin to destination shortest "*uncapacitated*" path for commodity k, if it exists
$SP_{uncap}(k,l)$	Origin to intermediate location $l \in L(k)$ shortest "*uncapacitated*" path for commodity k, if it exists
$K(r)$	Set of commodities $k \in K$ which can use resource $r \in R$
$R(e)$	Set of resources $r \in R$ which can be used by arc $e \in A$
$e_{uncap}(k,l)$	Earliest departing "*uncapacitated*" arc from location l for commodity k
$e_{depart}(l)$	Set of "*early departing*" arcs from a location $l \in L$
$K^{pt}(l)$	Set of pass-through commodities at location $l \in L$
cap_r^m	Total capacity required by all the commodities that can use resource r for each metric m

Table 8: Additional notation for problem reduction algorithm

Table 8 introduces some additional notation that we use in the formal description provided in Table 9.

Algorithm: *problem_reduction*
Initialize:

$cap_r^m = 0$, for each $r \in R$

$R_{uncap} = \phi$

$A_{uncap} = \phi$

Begin:

// obtain the set of "*uncapacitated*" resources
For each $r \in R$,
    For each $k \in K(r)$,
        $cap_r^m = cap_r^m + u_k^m$, for each $m \in M$
    End for
    If $cap_r^m < v_r^m$,
        Update r to the set of "*uncapacitated*" resources, $R_{uncap} \leftarrow R_{uncap} \cup \{r\}$,
    End if
End for

// obtain the set of "*uncapacitated*" arcs
For each $e \in A$,
    If $R(e) \subseteq R_{uncap}$
        Update e to set of "*uncapacitated*" arcs, $A_{uncap} \leftarrow A_{uncap} \cup \{e\}$,
    End if
End for

// remove feasible arc assignments beyond "*uncapacitated*" arc
For each $k \in K$,
    Obtain $A_{uncap}(k) = A(k) \cap A_{uncap}$ // the network of "*uncapacitated*" arcs for k

    Find shortest "*uncapacitated*" path for k in $A_{uncap}(k)$ up to each intermediate location $l \in L(k) \cup \{d_k\}$, $SP_{uncap}(k,l)$

    For each $l \in L(k)$
        If $SP_{uncap}(k,l) \neq \phi$, shortest "*uncapacitated*" path up to l exists
            For each $e \in A(k,l)$
                Remove e from $A(k)$,i.e., $A(k) \leftarrow A(k) \setminus \{e\}$, where
                $ts_e > ts_{e_{uncap}(k,l)}$, and $e_{uncap}(k,l) \in SP_{uncap}(k,l)$
            End for

Table 9: Algorithm for problem reduction

```
                End if
        End for

// remove assignment of pass-through commodities to "early departing" arcs

For each l∈L,
        For each k∈K^pt(l),
                For each e∈ e_depart(l),
                        If e∈A(k,l),
                                Remove e from A(k), i.e., A(k) ← A(k) \ {e}
                        End if
                End for
        End for
End for
```

Table 9: Algorithm for problem reduction (continued)

The problem reduction algorithm first defines the set of "*uncapacitated*" resources by comparing the combined capacity requirement of all commodities that can use a particular resource, and the available capacity of the resource. Then, it finds the set of "*uncapacitated*" arcs by examining each resource that an arc can use. After defining these sets, the algorithm finds the shortest path for each commodity k from its origin to each of its intermediate locations and destination in the network of "*uncapacitated*" arcs. If such a shortest path exists for a commodity up to any location l, by Proposition 5.1, we can eliminate all the feasible arc assignments that depart after this shortest path, except for crossing arcs. If the shortest path in the "*uncapacitated*" network for a commodity k is same as the shortest-path for k in the original network, then by Corollary 5.1, this procedure automatically eliminates all the other feasible arc assignments for the commodity. Finally for each location we obtain the set of "*early departing*" arcs and, by Proposition 5.2, eliminate the feasible assignments of these arcs to any pass-through commodity, leading to further reduction in the number of constraints and variables.

5.2 PROBLEM DECOMPOSITION

In this section, we discuss a method to decompose the problem so that the decomposed smaller components can be solved independently to obtain optimal solutions. First, we prove a result that forms the basis of the decomposition algorithm described next.

We say that two commodities directly share resources if they have at least one common resource in their respective set of resources they can use. Suppose we have three commodities $k1$, $k2$ and $k3$. If $k1$ and $k2$ do not share any resources directly but both share resources with $k3$, then we say that $k1$ and $k2$ share resources through $k3$.

***Proposition* 5.3:** If two commodities do not share any "*capacitated*" resource, either directly or through any other commodity, their optimal arc assignments are independent of each other.

Proof: To prove this, we first prove that if two commodities do not share any resource, directly or through any other commodity, then their solutions are independent of each other. Then, we extend this result to the set of "*capacitated*" resources only. Assume that we have two commodities $k1$ and $k2$. Commodity $k1$ can use resources $R(k1)$ through its arc set $A(k1)$, and commodity $k2$ can use resources $R(k2)$ through its arc set $A(k2)$. Suppose, $R(k1) \cap R(k2) \neq \phi$, i.e., the commodities share some resources directly. Then, the solution of $k1$ and $k2$ cannot be independent of each other, since $k1$'s assignment to some resource may not leave enough capacity for $k2$ to take that resource and hence affecting $k2$'s assignment. Now suppose $R(k1) \cap R(k2) = \phi$, but there is another commodity $k3$ with its resource set $R(k3)$, and $R(k1) \cap R(k3) \neq \phi$ and $R(k2) \cap R(k3) \neq \phi$, i.e., $k1$ and $k2$ do not share a resource directly but through commodity $k3$. In this case also, $k1$ and $k2$'s assignments are not independent of each other, since $k1$'s assignments may affect $k3$'s assignments which in turn may affect $k2$'s assignments. Thus, we can

consider two commodities to be independent if they do not share any resource either directly or through any other commodity.

Now we extend the above result to only the set of "*capacitated*" resources. Suppose, for the same commodities $k1$ and $k2$, $R(k1) \cap R(k2) \neq \phi$, but the set of shared commodities $R(k1) \cap R(k2) \subseteq R_{uncap}$, i.e., the shared resources are "*uncapacitated*". In this case, even if $k1$ is assigned to any resource $r \in R(k1) \cap R(k2)$, it does not prevent $k2$ from taking r because r is "*uncapacitated*". The result extends to the case of commodities $k1$, $k2$ and $k3$ where $k1$ and $k3$ share resources that are "*uncapacitated*", $k3$ and $k2$ share resources that are "*uncapacitated*", and $k1$ and $k2$ do not share any resource. In this case also, the solution of $k1$, $k2$ and $k3$ are independent of each other. Thus, if two commodities do not share any "*capacitated*" resource, either directly or through any other commodity, then their arc assignments are independent of each other.

■

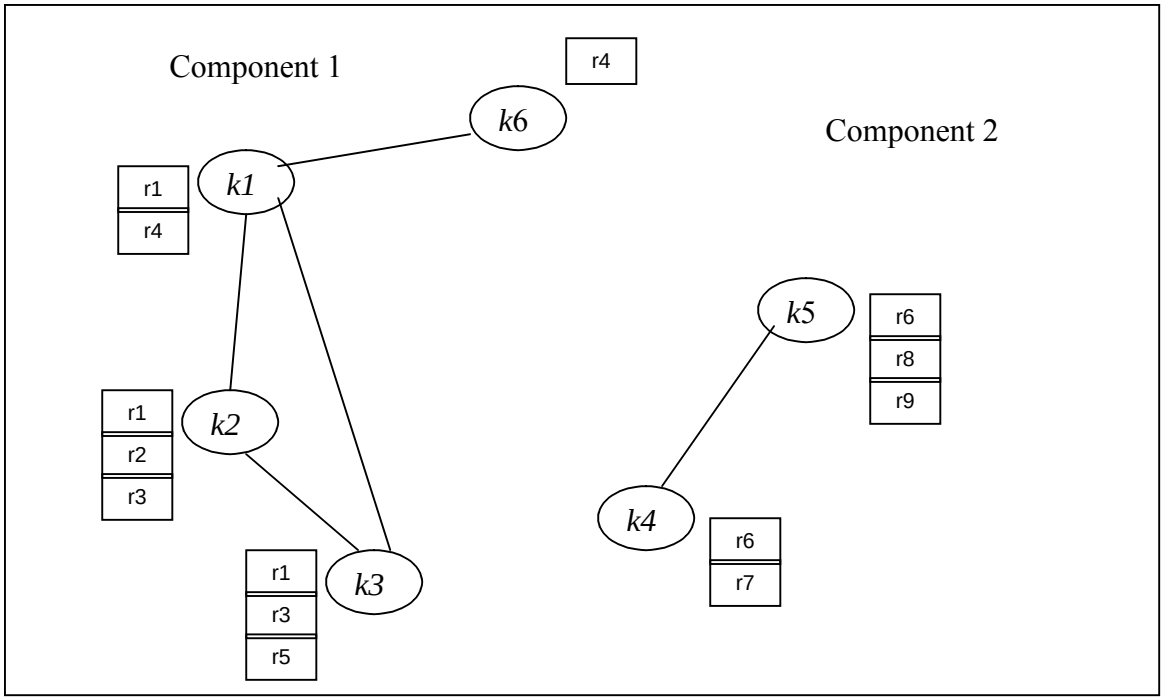

Figure 19: Problem decomposition example

Based on Proposition 5.3, we develop a problem decomposition algorithm as follows:

Step 1: Create a graph $G = (N,A)$, where $N=K$, i.e., each commodity in the model is a node in graph G .We create an arc $(i,j) \in A$, between any two nodes i and j if they directly share any "*capacitated*" resource ,i.e., $R_{cap}(i) \cap R_{cap}(j) \neq \phi$. Figure 19 shows an example of graph G created for 6 commodities.

Step 2: In this graph, find the set of maximal connected components.

Step 3: By Proposition 5.3, we can solve each of these components independently to obtain an optimal solution for the problem.

Hence, we decompose the problem into independent sets of commodities which do not influence each other's arc assignments, and solve smaller problems which can enhance the computational time and efficiency significantly.

5.3 HEURISTIC ALGORITHMS

In Chapter 4, we discuss several different formulations to model and solve the MCNF-FIFO problem using exact optimization methods. The formulations we discuss for FIFO and flow conservation constraints in the previous chapter reduce the number of variables and constraints considerably. Still, the problem size can be very big for some real world instances, and the exact algorithm may not be able to solve the problem in reasonable time. Heuristic algorithms become important in such situations as they not only provide good quality solutions quickly, but they also provide a good upper bound for exact optimization methods that can decrease the solution time.

In this section, we discuss three different heuristic algorithms to solve the MCNF-FIFO problem. The first one is an optimization-based heuristic algorithm in which we solve a relaxed problem using exact optimization methods, and then repair this solution to obtain a feasible solution. The other two methods are stand-alone construction heuristics in which a feasible solution is built from scratch.

5.3.1 Optimization-based heuristic algorithm: selective FIFO with repair heuristic

Our optimization-based heuristic algorithm primarily captures the interaction between the capacity constraints and their impact on FIFO violations. Note that if there are no capacity restrictions, then there cannot be any FIFO violations, since each commodity travels on its shortest path from its origin to destination. In presence of capacity restrictions, some commodities are forced to depart later and some are incentivized to leave early, thus violating the FIFO order. The empirical results from our computational tests show that nearly 70% of FIFO violations occur just upstream of a capacitated resource; moreover, nearly 99% of FIFO violations occur at arcs using at least one capacitated resource. Thus, if we can estimate a priori the resources which are most likely to be capacity constrained in an optimal solution, then the FIFO constraints

involving the arcs using these capacitated resources are most likely to be tight. This idea forms the basis of the selective FIFO with repair heuristic algorithm. The first step in the algorithm is to find a set of resources that are most likely to be capacity constrained in an optimal solution. Let us call this set R_{high_util}. We obtain this set of resources by examining the resources that are heavily utilized in the LP relaxation solution. Then, we determine the set of arcs A_{high_util} that can use these resources and apply FIFO constraints only on these arcs. We then solve the exact optimization model with only this limited set of FIFO constraints. Since we have applied constraints on the set of arcs where FIFO violations are most likely to occur, this model eliminates majority of the FIFO violations, and also improves the solution time as we are solving a relaxed model (partial FIFO constraint set). The solution to this relaxed optimization model may still contain some FIFO violations because we have not applied the complete set of FIFO constraints. These violations are then fixed using a repair heuristic to obtain a feasible solution.

The main advantage of the algorithm is that it applies FIFO constraints only on a selective set of arcs corresponding to the resources R_{high_util}. An important observation is that we need to apply FIFO constraints not only to the departure time corresponding to the capacitated arc (arc in A_{high_util}), but also to the immediate next departing arc, to be valid.

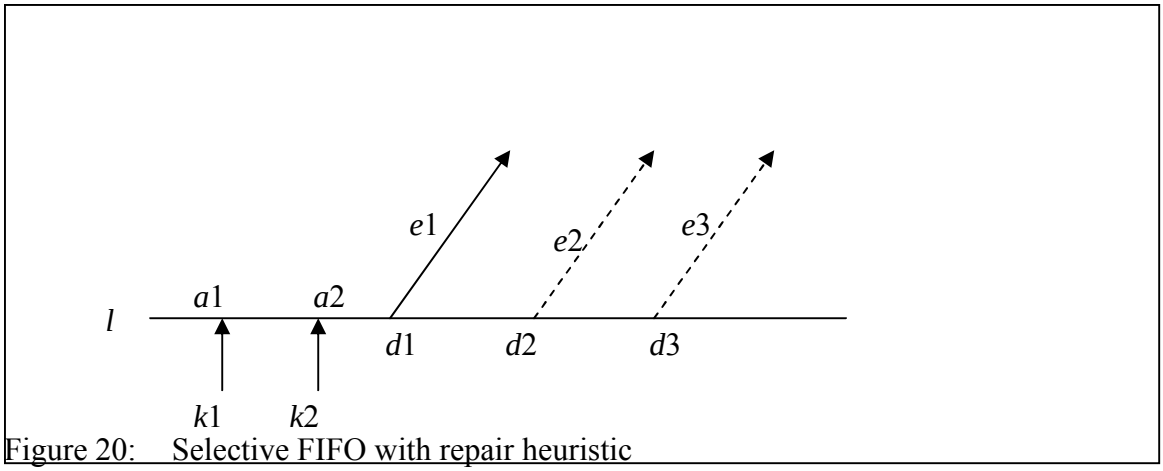

Figure 20: Selective FIFO with repair heuristic

Suppose in the example in Figure 20, the dashed arrows denote arcs which do not use any resource in R_{high_util} and the solid arrows denote arcs which use a resource in R_{high_util}. Assume that $k1$ departs at any time at or after $d1$. If we apply FIFO constraints only for the departure time of arc $e1$, then the constraint enforces that $k2$ must depart at or after $d1$. Thus, the constraint allows $k2$ to depart at $d1$ while $k1$ departs at $d2$, which is invalid. This situation can be handled if we also include the FIFO constraints for the departure time of the immediate next arc, in this case arc $e2$. As $e2$ is most likely to be uncapacitated, both $k1$ and $k2$ cannot stay at this location beyond $e2$, and thus we do not require any FIFO constraints after this time point in the example.

Based on the above discussion, we now provide a formal description for the selective FIFO with repair heuristic algorithm in Table 10. The repair heuristic algorithm is called only if the solution to the exact optimization model has some FIFO violations still remaining. The repair heuristic simply delays the departure of the commodity that violates FIFO from the location of violation, thereby resolving the current violation. It routes the FIFO violating commodity on the best available path from current location to

its destination. The heuristic procedure terminates when there are no more FIFO violations, and thus provides a FIFO and capacity feasible solution.

Algorithm *selective FIFO with Repair Heuristic*

Begin

Step1: Solve the LP relaxation of MCNF-FIFO.
Step 2: Obtain the resources which are above 95% utilized in LP solution, these form the set R_{high_util}
Step3: Identify the set of arcs A_{high_util} that use these resources
Step 4: Solve the IP model by selectively applying FIFO constraints on only the arcs $e \in A_{high\_util}$ and the immediate next arc.
Step 5: If IP solution has no FIFO violation, STOP, else call the repair heuristic

Repair heuristic:

While solution has no FIFO violation,
Sort FIFO violating commodities $K_{violators}$ in increasing order of arrival time at violation location
For each $k \in K_{violators,}$
Obtain the departure time of the commodity from current location t_{latest},
Create the earliest available path from current violation location to k's destination departing at or after t_{latest}
Update solution for the commodity and resource capacities
End for
End while

Table 10: Selective FIFO with repair heuristic algorithm

5.3.2 Commodity loading heuristic

The commodity loading heuristic (CLH) is a stand-alone construction heuristic that constructs a feasible solution for the MCNF-FIFO problem from scratch. The basic idea of the heuristic is to minimize the dwell time of a commodity at each location. In this algorithm, we maintain a list of all the commodities K, sorted in increasing order of their current arrival times $T(k)$ at their current location $CL(k)$. The algorithm traverses this list

of commodities in the sorted order, and loads each commodity to the next available arc e (which has capacity to carry it). Once a commodity is assigned to an arc e, its current location $CL(k)$ is updated to the destination of e, and current arrival time $T(k)$ is updated to the arrival time of e. If the destination of e is also the destination of commodity k, k is removed from the list K, or else, k is re-inserted into K according to its current arrival time $T(k)$, so as to maintain the sorted order of K. The procedure terminates when the list K is empty or all the commodities have reached their destination. Table 11 and 12 provide additional notation and a formal description of the algorithm.

In this algorithm, we do not need to worry about the FIFO order while assigning a commodity to an arc, as we are always assigning the commodity in a system-wide arrival time order. Thus, if a commodity arrives at a location earlier than another commodity, it is always given relative priority.

Another important point to note is the use of different sorting functions in the algorithm. We use two sorting functions, the *commoditySort*() and the *arcSort*() to break different ties in the algorithm presented in Table 10. The *commoditySort*() functions sorts the commodities according to their current arrival times $T(k)$, at their current locations. If multiple commodities arrive at the same time, then the volume of the commodity is used as the secondary sorting key. The *arcSort*() function sorts the arcs using the departure time as the primary key. If there are multiple arcs departing at the same time, then the departure times of the respective resources to be used by an arc are used as secondary keys. Finally, the arrival time at the destination is used as the tertiary key. These tie-breaking policies can have important impact on the quality of the solution, especially, if the input data has high symmetry.

ADDITIONAL NOTATION

K	Set of commodities sorted in their current arrival time order
$T(k)$	Current arrival time of commodity k at its current location
$CL(k)$	Current location of commodity k
$R(e)$	Resources that can be used by arc $e \in A$
$Sol(k)$	Solution arc assignments for commodity $k \in K$
s_e^m	Minimum capacity of a resource $r \in R(e)$ for each metric m
v_r^m	Capacity of a resource $r \in R$
ta_k	Arrival time of commodity k at its origin location
o_k	Origin location of commodity k
d_k	Destination location of commodity k

Table 11: Additional notation for commodity loading heuristic

Algorithm: *commodity loading heuristic*

Initialization:

For each $k \in K$, initialize the sets
$CL(k) = o_k$
$T(k) = ta_k$
End for

For each $e \in A$,
$s_e^m = \min(v_r^m),\ \forall r \in R(e)$, initialize the minimum capacity of arcs
End for

Begin:

Call *commoditySort*() to sort commodities in K in arrival time order

While $K \neq \phi$,

next $k \in K$,
Get $A(k,l)$, the set of arcs for commodity k at location $l=CL(k)$
Call *arcSort*(), to sort $A(k,l)$

For each $e \in A(k,l)$, iterate in sorted order
If $ts_e \geq T(k)$, then
If $u_k^m \leq s_e^m$, then
$Sol(k) \leftarrow Sol(k) \cup \{e\}$, update the solution arcs of k
$T(k) = te_e$, update arrival time of the commodity k
$CL(k) = d_e$, update current location of commodity k
For each $r \in R(e)$,
$v_r^m = v_r^m - u_k^m$, update capacity of resources
If $v_r^m < s_e^m$, then
Set $s_e^m = v_r^m$, update minimum arc capacity
End if
End for

Table 12: Commodity loading heuristic algorithm

If $CL(k) = d_k$, then ,if commodity has reached destination

Remove k from K, $K \leftarrow K \setminus \{k\}$

Else,

Insert k in K such that,

$K[i] = k$, and $T(k_{i-1}) \leq T(k) < T(k_{i+1})$

,insert k such that the arrival time sort order of K is maintained

End if

Else,

Remove e from $A(l)$, $A(l) \leftarrow A(l) \setminus \{e\}$

,remove e such that no other commodity arriving later can use this arc

End if

Next k

End

Table 12: Commodity loading heuristic algorithm (continued)

5.3.3 Arc loading heuristic

The arc loading heuristic is another class of stand-alone heuristic algorithms that generates a feasible solution for the MCNF-FIFO problem from scratch. It differs from the commodity loading heuristic because it aims at maximizing the utilization of an arc's capacity, instead of reducing the dwell time for each commodity at a location. In this algorithm we maintain a list of all the arcs A in the system in increasing order of their departure time. Similar to the commodity loading heuristic, we again use the *arcSort*() function to sort the arcs and break ties. For each arc e, the algorithm maintains a list of commodities $K(e)$ that can be assigned to e, in an increasing order of their arrival time. Again we use the *commoditySort*() function to break ties while maintaining this order. The algorithm traverses the list of sorted arcs in an increasing order, and loads each arc with as many commodities that can fit into it. The list of commodities that can be

assigned to each arc changes after the loading of one arc is complete. The procedure terminates when all the commodities have reached the destination or all the arcs have been scanned once. If there is enough capacity in the system such that each commodity has a feasible origin to destination path, then the algorithm is guaranteed to return a feasible solution for each commodity.

Since we are loading the arcs in increasing order of departure time, and each arc is assigned in a FIFO order, this algorithm implicitly imposes the FIFO order at each location. The notation for this algorithm is same the one provided in Table 11, and a formal description of the algorithm is given in Table 13.

It is interesting to note that even though the two stand-alone heuristic algorithms described above are similar in implementation, they are fundamentally different and perform differently for different objective functions. As we discuss above, the commodity loading heuristic tries to decrease the dwell time of a commodity at any intermediate location, and if the objective function is to minimize the total transit time or sum of delivery times, this algorithm performs better. However, when assigning a commodity to an arc and hence the resources using that arc, it does not care whether the commodities are assigned from different locations. On the other hand, the arc loading heuristic loads the arcs and hence the resources used by them in a homogeneous way, i.e., it loads the resources with commodities starting from the same location. This potentially can reduce the reclassification or other processing operations to be performed at any intermediate location. Thus, if our objective function is to minimize the cost of the work performed at each intermediate location, then the arc loading heuristic may provide a better solution than the commodity loading heuristic.

Algorithm: *arc loading heuristic*

Initialization

For each $k \in K$,

$T(k) = t_{ak}$, update the current arrival time to the arrival time at origin

$CL(k) = o_k$, update the current station of the commodity k

$KR(l) \leftarrow KR(l) \cup \{k\}$, where $l = o_k$, update $KR(l)$ with the commodities where l is the first location of commodity k

End for

For each $e \in A$,

$s_e^m = \min(v_r^m),\ \forall r \in R(p)$, initialize the minimum capacity of arcs

End for

Begin

Sort A in departure time order by calling *arcSort*(),

For each $e \in A$, iterate through A in the sorted order

Get $KR(l)$,where $l = o_e$,

Sort $KR(l)$ in current arrival time order based on $T(k)$,

For each $k \in K(l)$,

If $T(k) \le t_{se}$,

If $u_k^m \le s_e^m$,

$Sol(k) \leftarrow Sol(k) \cup \{e\}$, add e to the solution paths of k

$CL(k) = d_e$, update current location of commodity k

$KR(l) \leftarrow KR(l) \setminus \{k\}$, remove current commodity from $KR(l)$

$T(k) = te_e$, update the current arrival time of the commodity

If $CL(k) \neq d_k$.

$KR(d_e) \leftarrow KR(d_e) \cup \{k\}$, where d_e is the destination of arc e

End if

Table 13: Arc loading heuristic algorithm

For each $r \in R(e)$,

$v_r^m = v_r^m - u_k^m$, update capacity of resources

If $v_r^m < s_e^m$, then

Set $s_e^m = v_r^m$, update minimum arc capacity

End if

End for

Else,

Move to the next commodity k,

End if

End if

End for

End for

End

Table 13: Arc loading heuristic algorithm (continued)

Chapter 6: Computational results

In this chapter, our goal is to investigate the performance of the modeling enhancements and algorithmic approaches that we discuss in the previous chapters to solve the MCNF-FIFO problem. We focus on the following three major tasks:

- Comparison of different IP formulations in terms of their size and computational time to solve the problem (without any problem reduction)
- Assessing the impact of problem reduction and decomposition techniques on solution times.
- Evaluating the quality of upper bounds obtained by the different heuristic algorithms, and a comparative study of the computational effort required by these algorithms.

Moreover, we would like to understand whether these techniques can be extended to models that have the commodity routing problem with these special handling constraints as a sub-problem. Thus, it is important to understand not only the quality of solutions and the computational effort required to obtain them, but also to develop insights into how solutions from different approaches differ from each other.

We begin the section by introducing a real life problem instance and show that the problem sizes can be quite large. Then we discuss an approximation strategy to reduce the problem size and obtain a solution close to the optimal solution of the real problem. Finally, we provide a comparative analysis of the problem sizes and solution times of the different formulations and algorithms on three different real life datasets.

6.1 APPROXIMATION STRATEGIES

A real life instance of the MCNF-FIFO problem may have more than 40,000 commodities, nearly 1,000 locations, and around 20,000 resources. Our computational

results show that, for such instances, the number of variables and constraints for the formulations we deal with can be in the millions, and hence become computationally difficult to solve. To handle such problems, we introduce an approximation strategy that reduces the problem size and also mimics the real world operations more closely, thereby making it possible to obtain a feasible solution to this difficult problem, and implement it in practical settings.

Commodity aggregation

Two commodities travelling from the same origin to destination, following the same sequence of intermediate locations, and arriving at the origin location at the same time are most likely to follow the same route in their transit. From a real world operations point of view also, it may be more cost efficient to send them on the same route. Using this idea, we can combine or "aggregate" such commodities into a single commodity whose capacity requirement is equal to the sum of the capacity requirement of each individual commodity in the aggregated commodity. It is important to note that, commodity aggregation may provide a solution which is worse than the actual optimal solution, since we are constraining the routes for commodities by forcing a certain set of commodities to follow the same path. Thus, it may be important to understand the extent of approximation error introduced by commodity aggregation. The level of aggregation can be controlled by the user depending upon the level of approximation that is tolerable in the solution. This strategy can be a very effective when the data has a lot of symmetry, i.e., there are many commodities which can be aggregated, as it can lead to huge reduction in the number of variables and constraints without introducing too much approximation error. Table 14 outlines a basic algorithm for aggregating the commodities by following the logic described above. In the computational results that we present

below, we examine the effect of different levels of aggregation on solution quality and solution time.

algorithm: *commodity aggregation*

initialization:

Initialize *aggrFactor*, user defined aggregation level

$AK_{aggrFactor} = \phi$ // set of aggregated commodities

begin:

While $K \neq \phi$,
    Obtain $k \in$ K,
    Create an aggregated commodity *ak*
    Remove *k* from *K*, i.e., $K \leftarrow K \setminus \{k\}$
    For each $k' \in K$,
        If *k'* has the same origin-destination as *k*, follows the same sequence of intermediate location and arrives at the same time at origin,
        add *k'* to *ak*, i.e., $ak \leftarrow ak \cup \{k'\}$
        volume of *ak* = volume of *ak* + volume of *k'*
        Remove *k'* from *K*, i.e., $K \leftarrow K \setminus \{k'\}$
        If $|ak|$ = *aggrFactor*,
            break,
        End if
    End for
  next *k*,
End While

Table 14: Commodity aggregation algorithm

6.2 EXACT INTEGER PROGRAMMING FORMULATIONS

In this section we examine and compare the different IP formulations in terms of problem sizes and effectiveness in obtaining optimal or near-optimal solutions. All the exact optimization models are solved using CPLEX 12.1 using Concert technology. The tests are conducted on an 11 Dell Poweredge 2950 workstation with 3.73 GHz Xeon and

24 GB of shared memory under Ubuntu Linux system. All the computational results we discuss in this section are for a three-day planning horizon and aggregation of up to 2 commodities together. For all the IP models, the gaps are computed according to the objective function of the best know feasible solution, IP_{best}, as $(IP_{best} - LB)/LB$ where LB denotes the value of the best LP relaxation lower bound, and are reported as percentages. We set the termination criteria to a gap of 0.7% in CPLEX in all our computational tests. Also note that we set a maximum time limit of 4 hrs in CPLEX to obtain a solution after which the program automatically terminates.

We present the results here for three real-life datasets. All the instances have more than 40,000 commodities which reduce to around 25,000 commodities when we aggregate them with an *aggrFactor* of 2 as described in Section 6.1. Since we are considering a time horizon of 3 days, we only consider commodities that arrive within the first three days and resources that can be used by the arcs within 72 hours from the start of the horizon. After three days, each commodity that is still in transit travels on a shortest uncapacitated path to its destination.

Figure 21 shows the distribution of the FIFO violations, classified according to specialized types that we discuss in Section 4.1, when we solve the model without any FIFO constraints. As we can see, nearly 90% of the violations fall into the category of Type I or Type II. Thus, the specialized versions of FIFO constraints are likely to be quite effective since we can eliminate most of the FIFO violations with fewer constraints. Table 15 provides a detailed report on the problem sizes and solution times (if found) for different IP formulations for dataset 1. The results for other datasets are very similar. We provide the results for different combinations of FIFO and flow conservation formulations. When we have not found solution to an easier model, we have omitted the results for more constrained models. For example, the model was not able to find any

integer feasible solution in the given time limit for the commodity-pair FIFO Type I formulation with cumulative inflow-outflow flow conservation formulation; so we do not present the results for commodity-pair FIFO Type I & II formulation as it is a more constrained (difficult) model.

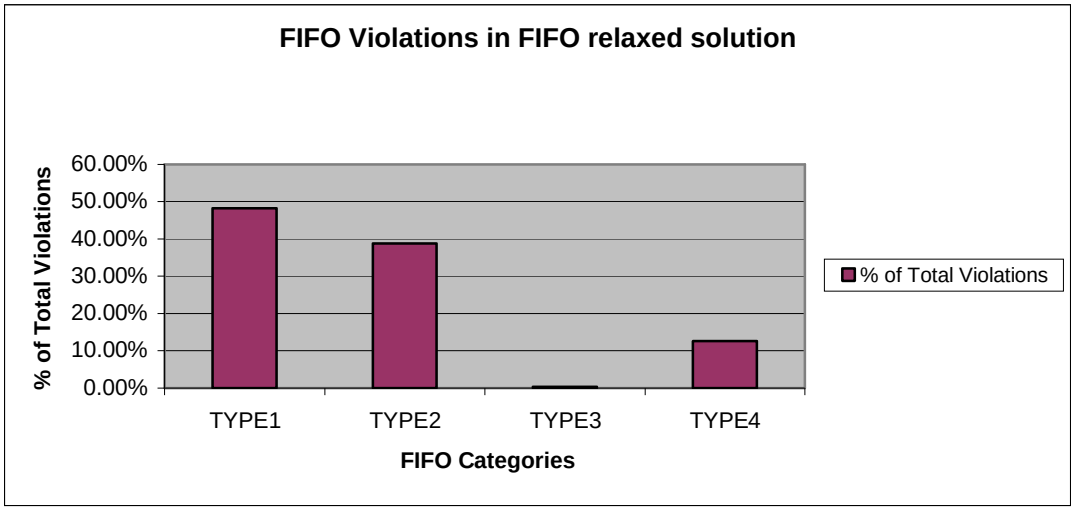

Figure 21: FIFO violations distribution according to classification types

FIFO constraint type	FIFO specialized type	Flow conservation constraints type	Number of variables	Number of constraints	LP time (in sec)	IP time (in sec)	FIFO violations[2]	Gap
Commodity -pair	Type I	Cumulative inflow-outflow	235K	423K	329	*NIS*[1]	NA	NA
Commodity -pair	Type I	Consolidated	340K	423K	215	*NIS*[1]	NA	NA
Commodity -pair	Type I	Simple	517K	551K	243	*NIS*[1]	NA	NA
Latest departure time	Type I	Cumulative inflow-outflow	240K	322K	220	*NIS*[1]	NA	NA
Latest departure time	Type I	Consolidated	345K	322K	295	*NIS*[1]	NA	NA
Latest departure time	Type I	Simple	522K	498K	265	*NIS*[1]	NA	NA
Arc-pair	Type I	Cumulative inflow-outflow	294K	408K	145	*NIS*[1]	NA	NA
Arc-pair	Type I	Consolidated	368K	408K	105	1,800	23.8%	0.7%
Arc-pair	Type I & II	Consolidated	368K	483K	130	2,700	6.3%	0.7%
Arc-pair	Type I & II, partial III & IV	Consolidated	379K	566K	210	4,500	3.6%	0.7%
Arc-pair	All	Consolidated	469K	613K	450	*NIS*[1]	NA	NA
Arc-pair with inventory variables	All	Simple	552K	741K	180	3,300	0	0.7%

[1] *NIS*: No integer solution found within 4 hrs of runtime
[2] *= (FIFO violations in current solution/FIFO violations in FIFO relaxed IP solution)*100

Table 15: Comparison of formulations for dataset 1

The arc-pair FIFO model with inventory variables performs the best as it finds an optimal solution within a gap 0.7% in a reasonable amount of time. Note that the arc-pair

FIFO model with inventory variables can only be formed with the "simple" version of the flow conservation constraints, since in this formulation we require the inventory information for each commodity at each arrival and departure time point. It is interesting and surprising that the arc-connection FIFO formulation with the consolidated flow conservation formulation is not able to find any integer feasible solution in the given time limit even though this model has fewer constraints than the arc-pair FIFO model with inventory variables. Also note that both the formulations are similar in terms of the LP relaxation strength. The difference in the formulations is the sparsity of the constraint matrix that makes it easier and faster to find an integer feasible solution to this difficult problem. The latest departure time formulation does not perform well even for a limited set of FIFO constraints (FIFO Type I) despite having the fewest constraints. The primary reason is the weakness of the LP relaxation (Section 4.2) that makes it difficult to find an integer feasible solution. Finally, the commodity-pair FIFO formulation is also not able to find any integer feasible solution within the given time even with only the FIFO Type I constraints. The primary reason in this case is the size of the constraint matrix which makes it difficult to obtain a solution. Our computational results show an interesting relationship between the three indicators of a good formulation that we discuss in Chapter 4. The results show that the sparsity of the constraint matrix can be an important factor in improving the performance of a formulation even when it increases the number of constraints. The strength of the LP relaxation also plays an important role to obtain solutions in quick time, but may not be very successful if the constraint matrix is very dense.

In the next section we discuss the impact of problem reduction and decomposition techniques presented in Chapter 5 and the effect of different aggregation levels on solution quality and solution time.

6.3 PROBLEM REDUCTION, DECOMPOSITION AND AGGREGATION

As we discuss in the previous section, the arc-pair FIFO model with inventory variables performs best in terms of computational time. Hence, in this section we use only this formulation to study the impact of problem reduction techniques on reducing the problem sizes and improving runtimes for all the three data instances. Figure 22 shows a histogram comparing the number of variables and constraints for the three instances before and after applying the problem reduction techniques. The problem reduction techniques reduce the number of variables and constraints by around 10% on an average across different datasets, and reduce the runtimes by around 40% on an average, which is a significant improvement. For example, for dataset 1, the problem reduction procedure reduces the movement variables by nearly 12%, the inventory variables by 10%, the flow conservation constraints by around 10%, and the FIFO constraints by around 9%. The overall runtime reduces around 40% from 55 minutes to 35 minutes. Similarly, for dataset 2 the variables reduce by around 8%, the constraints reduce by around 10% and the runtime reduces by 42%. It is interesting to note the significant reduction in the runtimes after applying the problem reduction techniques that make them quite effective for real life implementations.

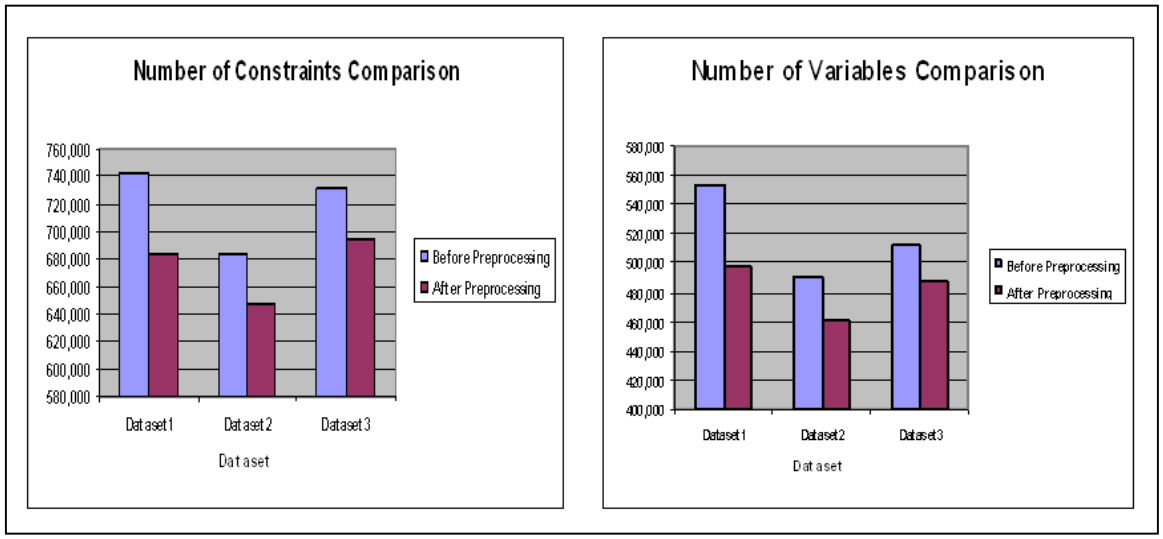

Figure 22: Problem reduction comparison (constraints and variables)

The problem decomposition technique described in Section 5.2 does not prove very effective in our test instances because the resources and commodities are highly connected in all these instances. We are able to decompose each dataset into multiple components, but in each case there is one major component which has most of the commodities. For example, in dataset 1 we have around 25,000 commodities that decompose into five components after applying the problem decomposition technique, but one component out of the five has around 24,000 commodities, and the rest are smaller components with around 200 commodities each.

Another important computational result is the effect of different levels of aggregation on solution quality and runtime. Table 16 shows the result for different datasets, the scaled best lower bound, the scaled best upper bound and the solution time with increasing aggregation level.

Dataset	Aggregation Level	Best Lower Bound*	Best Upper Bound*	Solution Time (in seconds)
1	2	100	101.04	2100
1	3	99.98	101.12	1890
1	5	100.28	101.37	735
1	6	100.27	101.35	525
2	2	100	100.95	1680
2	3	100.02	100.74	2940
2	5	100.06	101.09	1050
2	6	100.06	101.12	630
3	2	100	100.85	1560
3	3	100.01	100.88	1115
3	5	100.08	101.01	880
3	6	100.07	100.95	670

* = 100*(value/best lower bound of corresponding 2-aggr model)

Table 16: Solution quality and time with different aggregation levels

Interestingly, the objective value does not deteriorate much even with increasing the *aggrFactor* up to 6, while the runtime decreases significantly because of the reduction in the problem size. This is primarily because, even in a model with lower aggregation, the commodities follow the same routes as they are forced to follow in a higher aggregated model. This provides a good control in real life scenarios where an approximately good quality solution is required in a quick time.

6.4 HEURISTIC ALGORITHMS

We discuss three different classes of heuristic algorithms for the MCNF-FIFO problem in Section 5.3. The computational experiments with the exact optimization methods clearly show that it may not be always possible to obtain optimal or even good quality feasible solutions in an acceptable time. In this section we investigate the performance of the proposed heuristic algorithms in terms of the quality of solutions and the time it takes to obtain these solutions.

The first approach is the optimization-based heuristic algorithm we discuss in Section 5.3.1 called selective FIFO with repair heuristic. An important consideration in this algorithm is the method to determine the set of resources that are most likely to be capacity constrained in an optimal solution. We can obtain this set of resources from the shortest path solution, the LP relaxation solution or any of the stand-alone heuristic algorithm solutions described in Section 5.3. Our experiments show that the LP relaxation solution is the best indicator of the set of resources that are going to be capacity constrained in the optimal integer solution. We consider the resources which are utilized in excess of 95% in the LP solution as the set of resources most likely to be capacitated in the integer solution. Table 17 shows a detailed comparison of the results obtained for different models for dataset 1 and 2. The number in the predicted capacitated resources column denotes the resources that are utilized above 95% in the corresponding model. The FIFO violations column shows the percentage of FIFO violations still remaining in the relaxed IP solution, when FIFO constraints are "*selectively*" applied on the arcs using these resources. The newly capacitated resource column denotes the resources that are capacity constrained in the relaxed IP solution, but not present in the predicted capacitated resources. It clearly shows that the LP solution is a good predictor of capacitated resources as the number of new resources being capacity constrained in the relaxed integer solution is lower, and hence the numbers of FIFO violations are also lower. Since the number of FIFO violations is lower, the repair heuristic fixes the remaining FIFO violations in less than 10 iterations in every instance, and completes the repair procedure in less than 60 seconds. Table 17 shows only the solution time for the relaxed IP model, not including the time for solving the corresponding model to obtain the $R_{\text{high\_util}}$ set and for applying the repair heuristic.

Dataset	Model	Predicted capacitated resources	FIFO violations*	Newly capacitated resources	Solution time (in seconds)
1	Shortest-path	337	13.25%	86	360
1	LP relaxation	393	0.8%	8	840
1	Commodity loading heuristic	345	3.4%	70	720
2	Shortest-path	385	23.33%	105	240
2	LP relaxation	420	0.2%	9	480
2	Commodity loading heuristic	395	3.01%	43	600

*= (FIFO violations in current solution/FIFO violations in FIFO relaxed IP solution)*100

Table 17: Selective FIFO with repair heuristic: results with different resource sets

Finally, in Figures 23-25 and Tables 18-20, we compare the solution time and gap for the solutions obtained using the three different heuristic methods and the exact optimization procedure, with the corresponding exact optimization lower bound for the three datasets. As shown, the selective FIFO with repair heuristic performs the best among all the methods. It obtains a solution for each test instance within 1.5% gap of the IP lower bound within an acceptable time. Note that the time for the selective FIFO with repair heuristic includes the time for solving the initial LP relaxation, the relaxed IP model and the repair heuristic. The two stand-alone heuristic algorithms provide quick results but the objective function value of the solution is 3 to 6% larger than the best lower bound.

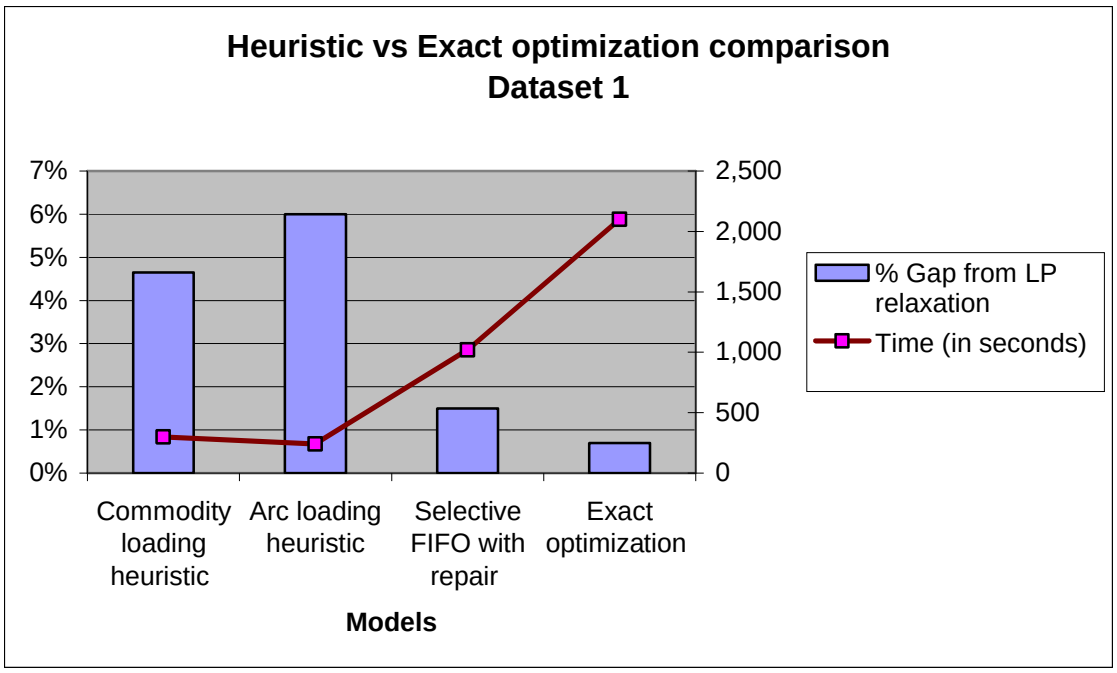

Figure 23: Heuristic & exact optimization comparison (dataset 1)

Dataset 1

Model	% gap from LP relaxation	Time(in seconds)
Commodity loading heuristic	4.65%	300
Arc loading heuristic	6.00%	240
Selective FIFO with repair	1.50%	1,020
Exact optimization	0.70%	2,100

Table 18: Heuristic & exact optimization comparison (dataset 1)

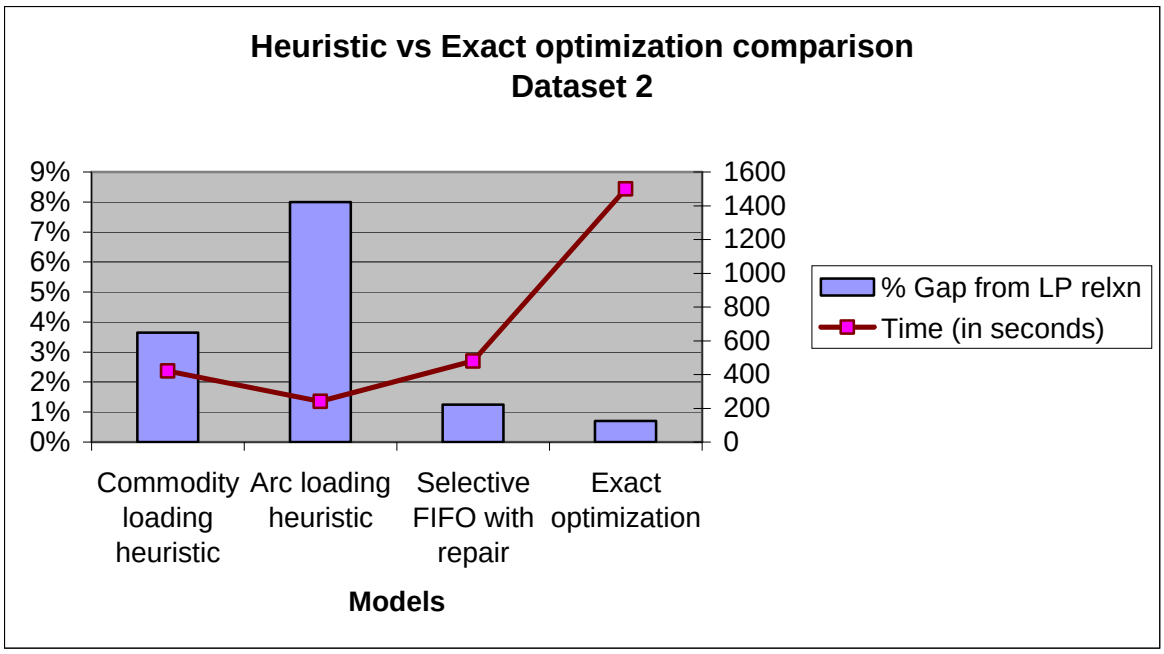

Figure 24: Heuristic & exact optimization comparison (dataset 2)

Dataset 2

Model	% gap from LP relaxation	Time(in seconds)
Commodity loading heuristic	3.65%	420
Arc loading heuristic	8.00%	240
Selective FIFO with repair	1.25%	480
Exact optimization	0.70%	1500

Table 19: Heuristic & exact optimization comparison (dataset 2)

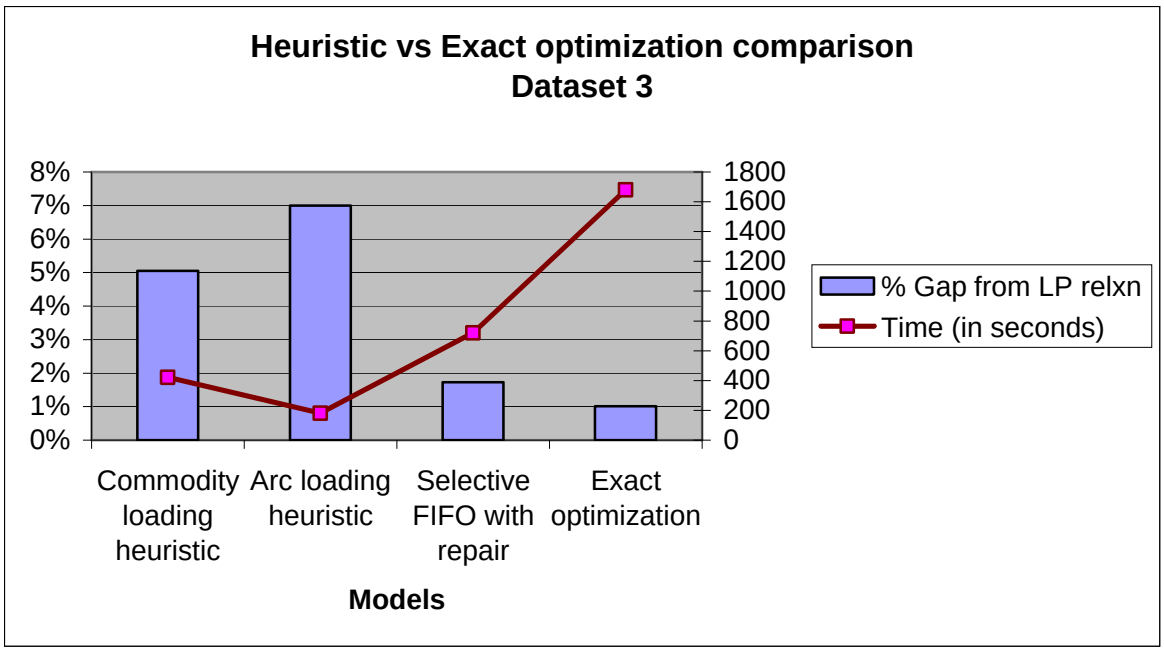

Figure 25: Heuristic & exact optimization comparison (dataset 3)

Dataset 3

Model	% gap from LP relaxation	Time(in seconds)
Commodity loading heuristic	5.05%	420
Arc loading heuristic	7%	180
Selective FIFO with repair	1.73%	720
Exact optimization	1.01%	1680

Table 20: Heuristic & exact optimization comparison (dataset 3)

Chapter 7: Conclusions and further research

In this research we study an extension of the general capacitated multicommodity network flow problem in which the commodities have to follow a unique path from origin to destination and there are FIFO handling policies at transshipment locations. The FIFO handling policies govern the order in which commodities are processed at a location based on their arrival times.

We model the problem on a time-space network and formulate it as an integer min-cost multicommodity network flow problem with FIFO handling policies as side constraints (MCNF-FIFO). Since the problem size can be huge even for a medium size real world instance, we provide several different modeling enhancements to alternatively model the FIFO constraints and the flow conservation constraints. We discuss three different alternative formulations for the FIFO constraints and two alternative formulations for the flow conservation constraints, each with their pros and cons, primarily in terms of the number of constraints, the LP relaxation tightness and the sparsity of the constraint matrices. Since the exact optimization methods may not always provide good solutions in a satisfactory amount of time (in terms of real world operations), we develop several algorithmic strategies to obtain quick solutions. The first strategy is to develop specialized versions of the FIFO constraints by exploiting relationships between various commodities, and thus reducing the number FIFO constraints. The second strategy is to develop a sophisticated problem reduction and problem decomposition technique which fixes variables and removes constraints prior to solving the model, reducing the runtime drastically. To obtain near-optimal feasible solutions quickly and to generate good upper bounds for the exact optimization methods, we develop three different heuristic algorithms. The first one is an optimization-based

heuristic algorithm called the selective FIFO with repair heuristic in which we predict the set of resources most likely to be capacity constrained in an optimal solution. Then, we "selectively" apply FIFO constraints on only the arcs that use these resources and solve a relaxed IP model. As the number of constraints is small, this model solves quickly and provides a solution with very few remaining FIFO violations that are repaired through a repair heuristic. The other two heuristic algorithms are stand-alone construction algorithms that build a feasible solution for the problem from scratch. Finally, we investigate the performance of all the different combinations of the model enhancements, specialized formulations, problem reduction techniques and heuristic algorithms on real world instances. We show that the arc-pair FIFO with inventory variables formulation outperforms other exact optimization models in a large problem instance and the problem reduction methods reduce the runtime by 40% on average. The selective FIFO with repair heuristic algorithm performs the best as it provides near optimal solution within an acceptable time limit.

The next steps in this research are to develop some valid inequalities and intelligent branching rules which can reduce the time to obtain solutions by exact optimization methods. The success of the optimization-based heuristic also encourages us to investigate the possibility of LP rounding based heuristic methods that may outperform the current algorithm both in terms of runtime and solution quality. Finally, we hope to study other handling policies that may govern the choice of commodities routes in a time-space network.

We believe that this research will lead to further studies into incorporating more challenging real world operational constraints in routing and scheduling problems and the modeling and algorithmic strategies that we discuss here can be very effective in obtaining practical solutions to real-life problems.

References

Ozdaglar, A.E. & Bertsekas, D.P., 2003. Optimal solution of integer multicommodity flow problems with application in optical networks. In *Proc. of Symposium on Global Optimization*.

Agarwal, R. & Ergun, O., 2008. Ship scheduling and network design for cargo routing in liner shipping. *Transportation Science*, 42(2), 175-196.

Aggarwal, A., 1995. A heuristic solution procedure for multicommodity integer flows. *Computers & Operations Research*, 22(10), 1075-1087.

Ahuja, R.K., Liu, J., Orlin, J.B., Sharma, D. & Shugharth, L.A., 2005. Solving real-life locomotive-scheduling problems. *Transportation Science*, 39(4), 503-517.

Assad, A.A., 1978. Multicommodity network flows—a survey. *Networks*, 8(1), 37-91.

Awerbuch, B. & Leighton, T., 1993. Multicommodity flows: a survey of recent research. In *Algorithms and Computation*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 297-302. Available at: http://dx.doi.org/10.1007/3-540-57568-5_260.

Barnhart, C., Hane, C. & Vance, P., 1996. Integer multicommodity flow problems. In *Integer Programming and Combinatorial Optimization*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 58-71. Available at: http://dx.doi.org/10.1007/3-540-61310-2_5.

Barnhart, C., Jin, H. & Vance, P.H., 2000. Railroad blocking: A network design application. *Operations Research*, 48(4), 603-614.

Bellmore, M., Bennington, G. & Lubore, S., 1971. A multivehicle tanker scheduling problem. *Transportation Science*, 5(1), 36-47.

Booler, J.M.P., 1980. The solution of a railway locomotive scheduling problem. *Journal of the Operational Research Society*, 31(10), 943-948.

Chen, Y.L. & Chin, Y.H., 1992. Multicommodity network flows with safety considerations. *Operations Research*, 40(1-Supplement-1), S48-S55.

Crainic, T.G., 2000. Service network design in freight transportation. *European Journal of Operational Research*, 122(2), 272-288.

Crane, R.R., Brown, F.B. & Blanchard, R.O., 1955. An analysis of a railroad

classification yard. *Journal of the Operations Research Society of America*, 3(3), 262-271.

Escudero, L. & Munoz, S., 1998. On characterizing tighter formulations for 0–1 programs. *European Journal of Operational Research*, 106(1), 172-176.

Florian, M., Bushell, G., Ferland, J., Guérin, G., & Nastansky, L., 1976 The engine scheduling problem in a railway network, Infor, 14(2), 121-138.

Forbes, M.A., Holt, J.N. & Watts, A.M., 1991. Exact solution of locomotive scheduling problems. *Journal of the Operational Research Society*, 42(10), 825-831.

Haghani, A. & Oh, S., 1996. Formulation and solution of a multi-commodity, multi-modal network flow model for disaster relief operations. *Transportation Research Part A: Policy and Practice*, 30(3), 231-250.

Holmberg, K. & Yuan, D., 2003. A multicommodity network-flow problem with side constraints on paths solved by column generation. *INFORMS Journal on Computing*, 15(1), 42-57.

Jha, K.C., Ahuja, R.K. & Şahin, G., 2008. New approaches for solving the block-to-train assignment problem. *Networks*, 51(1), 48-62.

Kennington, J.L., 1978. A Survey of Linear Cost Multicommodity Network Flows. *Operations Research*, 26(2), 209-236.

Kim, D., Barnhart, C., Ware, K., Reinhardt, G., 1999. Multimodal Express Package Delivery: A Service Network Design Application. *Transportation Science*, 33(4), 391-407.

Kwon, O., 1998. Routing and scheduling temporal and heterogeneous freight car traffic on rail networks. *Transportation Research Part E: Logistics and Transportation Review*, 34(2), 101-115.

Qi, X., Bard, J.F. & Yu, G., 2004. Class scheduling for pilot training. *Operations Research*, 52(1), 148-162.

Rao, M.R. & Zionts, S., 1968. Allocation of transportation units to alternative trips--a column generation scheme with out-of-kilter subproblems. *Operations Research*, 16(1), 52-63.

Smith, M., 1993. A new dynamic traffic model and the existence and calculation of dynamic user equilibria on congested capacity-constrained road networks. *Transportation Research Part B: Methodological*, 27(1), 49-63.

Vaidyanathan, B., Jha, K.C. & Ahuja, R.K., 2007. Multicommodity network flow approach to the railroad crew-scheduling problem. *IBM Journal of Research and Development*, 51(3), 325-344.

White, W.W. & Bomberault, A.M., 1969. A network algorithm for empty freight car allocation. *IBM Systems Journal*, 8(2), 147-169.

Wollmer, R.D., 1971. Multicommodity networks with resource constraints: The generalized multicommodity flow problem. *Networks*, 1(3), 245-263.