Summer 8-6-2019

# Extreme-point Tabu Search Heuristics for Fixed-charge Generalized Network Problems

Angelika Leskovskaya
*Southern Methodist University*, aleskovs@smu.edu

EXTREME-POINT TABU SEARCH HEURISTICS FOR
FIXED-CHARGE GENERALIZED NETWORK PROBLEMS

Approved by:

Dr. Richard Barr

Dr. Michael Hahsler

Dr. Richard Helgason

Dr. Eli Olinick

Dr. Halit Üster

EXTREME-POINT TABU SEARCH HEURISTICS FOR

FIXED-CHARGE GENERALIZED NETWORK PROBLEMS

A Dissertation Presented to the Graduate Faculty of the

Lyle School of Engineering

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Doctor of Philosophy

with a

Major in Operations Research

by

Angelika Leskovskaya

(M.A., Belarusion State University, 1993)

August 6, 2019

## ACKNOWLEDGMENTS

There are many people that have earned my gratitude for their contribution of time, support and encouragement throughout all those years to achieve progress in my research. I would like to thank several group of people without whom this thesis would not be possible: my advisor, my thesis committee members, my family, and all people that I have a privilege to meet in my life during the time of research discoveries.

This research would not be possible without my advisor and mentor Dr. Richard Barr. With his help and guidance even the most complicated approaches became clear and possible to implement. I am indebted to his relentless support and unparalleled opportunities I was given while working on research and teaching at SMU. I am extremely thankful to his profound belief in my abilities and patience that cannot be underestimated while working on the solver for the generalized network models.

Besides my advisor, I would like to thank my committee members - Dr. Michael Hahsler, Dr. Richard Helgason, Dr. Eli Olinick, and Dr. Halit Üster. I am endlessly thankful to Dr. Olinick for his help, advice, and ingenious suggestions in handling CPLEX and for the best classes in Network Flows and Integer Programming that gave me the basics for my research. I am deeply grateful to Dr. Helgason for his support and encouragement throughout the duration of this research and also for sharing his love of mathematical proofs and the Theory of Optimization. I am quite appreciative of Dr. Michael Hahsler's insightful comments about explaining the detailed subject matter with examples and graphs. I would like to extend my deepest gratitude to Dr. Halit Üster for his constant support and assistance in completing the dissertation.

My family is the best I have and I am deeply thankful for their support and unconditional love all those years. Egor, Danila, and Artem - you have always been my best “constraints” and it has made me more determined and believing in myself. I am grateful to my husband Igor, his support and encouragement. I am also thankful to my brother Sergey and his statistical insights and discussions with me. Mom and Dad - I love you very much and I am proud to be your daughter.

I would like to extend my sincere thanks to Tammy Sherwood who was a tremendous support and always knew how to resolve any organizational issues. Many thanks to all faculty, including but not limited to, Narayan Bhat, Sila Cetinkaya, David Matula, Margaret Dunham, Peter Moore and Gheorghe Spiride who I had as my professors and who have shaped my character and knowledge base, and fellow doctoral students who shared jokes, discoveries, knowledge, projects, conferences, as well as successes and failures over those years.

I would like to offer special thanks to Dr. Jeffrey Kennington, who, although no longer with us, continues to inspire me by his example and dedication to students he taught over the course of his career. I am deeply thankful for his decision to accept me into the program and for his guidance in every small or big problem I had a privilege to work on with him.

Leskovskaya, Angelika M.A., Belarusion State University, 1993

Extreme-point Tabu Search Heuristics for Fixed-charge Generalized Network Problems

Advisor: Professor Dr. Richard Barr

Doctor of Philosophy degree conferred August 6, 2019
Dissertation completed July 29, 2019

While researchers have studied generalized network flow problems extensively, the powerful addition of fixed charges on arcs has received scant attention. This work describes network-simplex-based algorithms that efficiently exploit the quasi-tree basis structure of the problem relaxations, proposes heuristics that utilize a candidate list, a tabu search with short and intermediate term memories to do the local search, a diversification approach to solve fixed-charge transportation problems, as well as a dynamic linearization of objective function extension for the transshipment fixed-charge generalized problems. Computational testings for both heuristics demonstrate their effectiveness in terms of speed and quality of solutions to these mixed-integer models. Comparisons with a state-of-the-art solver, CPLEX, show that the extreme-point search algorithm runs, on average, for transportation problems five orders of magnitude faster and produces integer solution values within 2.2% of the optimal solution reported by CPLEX; for transshipment problems, the heuristic solver ran 1,000 faster and the solution values were within 2.5% of the optimal reported by CPLEX.

## TABLE OF CONTENTS

## LIST OF FIGURES

# LIST OF TABLES

Table Page

*This dissertation is dedicated to the memory of Dr. J. Kennington who gave me love for generalized networks and curiosity as to how to implement algorithms to solve them*

# Chapter 1

## Introduction and Overview

Researchers and practitioners in the fields of operations research and computer science are familiar with two main categories of network flow models, *pure network flow models* and *generalized network flow models* and some of their special forms that include shortest path, maximum flow, transportation, and assignment problems. This widely used class of optimization models gained its popularity due to visual representation, model flexibility and comprehensiveness, and solvability [54]. The special-structure linear programming (LP) models can be described pictorially, as in Figure 1.1, as a set of circles or *nodes* with a defined amount of some commodity supplied or demanded at some or all of the nodes. Nodes are connected pairwise by a set of arrows or *directed arcs*, across which commodity units flow while incurring costs.

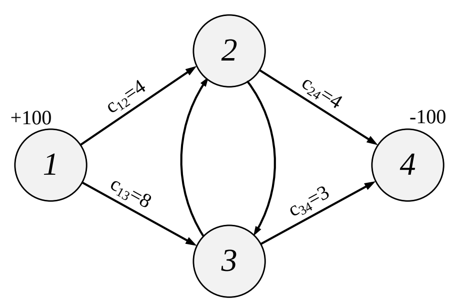



Figure 1.1. Network Flow Model

The objective is to route the flow from source nodes with supplies of commodity, shown as adjacent positive requirement, through the available arcs as *flows*, to sink nodes with commodity demands, shown as adjacent negative requirements for flow, while minimizing the total cost and maintaining conservation of flow at each node.

The diversity of problems from fields such as engineering, communication sociology, archaeology, government regulatory policies, financial planning, and production falls into the network domain. Figure 1.2 depicts a pure network model for a multi-period supply-chain network with inventories and backorders. It also represents an LP model with 26 variables, 14 flow-balance constraints, and 52 individual variable constraints. Its solution determines the minimum-cost plan for manufacturing level by plant, distribution to warehouses, and sales for each period, and inventory levels between periods [12]. The cost values are shown above arcs with lower and upper bounds for flows positioned below arcs with default values $c_{ij} = 0$, $l_{ij} = 0$, $u_{ij} = \infty$ if not shown.

The mathematical formulation for the minimum cost pure network model is defined as:

$$\text{Minimize} \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \tag{1.1}$$

$$\text{subject to:} \sum_{j:(i,j)\in\mathcal{A}} x_{ij} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji} = b_i, \forall i \in \mathcal{N} \tag{1.2}$$

$$0 \leq x_{ij} \leq u_{ij}, \forall (i,j) \in \mathcal{A} \tag{1.3}$$

where $\mathcal{N}$ = the set of problem nodes, $\mathcal{A}$ = the set of all problem arcs, $b_i$ = requirement at node $i$, $u_{ij}$ = upper bound on arc $(i,j)$, $c_{ij}$ = cost per unit of flow on arc $(i,j)$, and $x_{ij}$ = flow on arc $(i,j)$. Each directed arc in a network corresponds to a variable whose flow value is to be determined. Since constraint (1.2) ensures that the total flow into and out of a node match its supply or demand requirements (or zero if neither), each arc appears in the constraints of only its two endpoint nodes (one coefficient

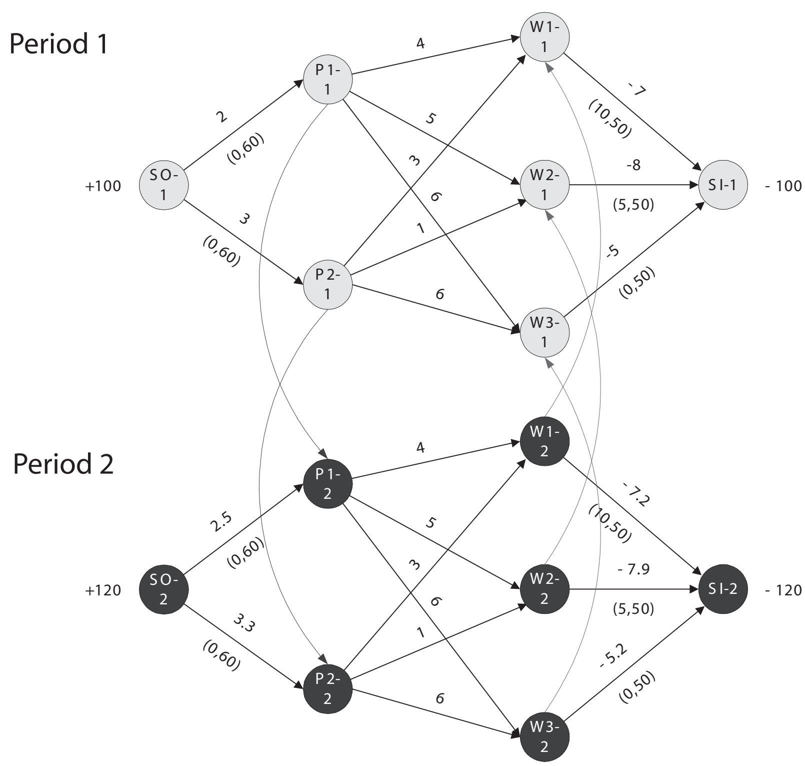



Figure 1.2. Multi-period manufacturing process, with inventories and backorders

is $+1$ , one $-1$). These equations and variables thereby form a *node-arc incidence matrix* with an example provided in Section 1.1.2.

When a network model comes from the fact that all nodes are partitioned into two subsets $N_1$ and $N_2$ of possibly unequal cardinality such that each node in $N_1$ is a source node, each node in $N_2$ is a sink node, and for each arc $(i, j) \in \mathcal{A}$, $i \in N_1$ and $j \in N_2$, the embodied structure is called a *transportation* or *bipartite* network. It is a special case of the minimum cost flow problem above. More complex practical network problems typically have intermediate, or *transshipment* nodes, through which commodities may be shipped en route to their final destination.

Regarding the solvability property, efficient algorithms and codes are available to quickly solve large-scale instances of pure network flow models. These techniques exploit the special structure of the constraint coefficient matrix and the ability to represent a basic solution as a spanning tree of the nodes [10, 11, 22, 31, 32, 56, 61, 70, 71, 77].

Pure network flow problems and algorithms have been studied extensively with their extensions to incorporate fixed-charges on all or some of the arcs. *Fixed-charge networks* (FC) have a special property for a fixed-charge arc: if the arc is permitted to transmit flow, a charge is incurred that is independent of the amount of flow. The problem was originally discussed by Hirsch and Dantzig in [64] and a variety of solution approaches and algorithms were proposed in [9, 13, 60, 76, 79, 85, 90, 92, 101, 109] for transportation problems. The fixed-charge network flow problem has many practical applications in transportation, network design, plant location problems, production scheduling, investment and distribution problems. The main decision is whether or not to use an arc in the network modeled by adding fixed charges on appropriate arcs.

The mathematical formulation for the pure fixed-charge network model is the following:

$$
\begin{aligned}
FCP: \quad & \text{Minimize } z = \sum_{(i,j)\in\mathcal{A}} (c_{ij}x_{ij} + f_{ij}y_{ij}) && && (1.4)\\
& \text{subject to: } \sum_{j:(i,j)\in\mathcal{A}} x_{ij} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji} &=\;& b_i, \forall i \in \mathcal{N} && (1.5)\\
& \qquad\qquad 0 \le x_{ij} &\le\;& u_{ij}y_{ij}, \forall (i,j) \in \mathcal{A} && (1.6)\\
& \qquad\qquad 0 \le y_{ij} &\le\;& 1, \forall (i,j) \in \mathcal{A} && (1.7)\\
& \qquad\qquad y_{ij} & & \mathit{integer}, \forall (i,j) \in \mathcal{A} && (1.8)
\end{aligned}
$$

where $y_{ij} = 1$ if arc $(i, j)$ is active, 0 otherwise, and $f_{ij}$ is the fixed charge associated with the activation of flow on arc $(i, j)$.

*Generalized networks* (GN) are enhancements of the pure-network modeling form described above. These LP models have much of the same structure but allow flow to be modulated by a *multiplier* associated with each arc, such that for each unit of flow entering an arc a multiple of it arrives at its destination node. If an arc's multiplier is 1, it operates exactly like a pure network arc. If the multiplier is greater/less than 1, the arriving units are greater/fewer than the entering quantity. This characteristic enlarges the modeling applications and enables the flow units to grow, shrink, or change units within the network. Generalized networks therefore expand modeling capabilities of many practical problems that cannot be adequately captured by a pure network representation [54]. For a simple illustration, Figure 1.3 represents a 2-period example in which beginning cash at the start of period 1 can be carried over either by 1-period investment with 8% return or 2-period investment with 11% return. When a loan is paid out of future funds, it can be modeled with the backward generalized arc and a multiplier of 1/1.1 for this example.

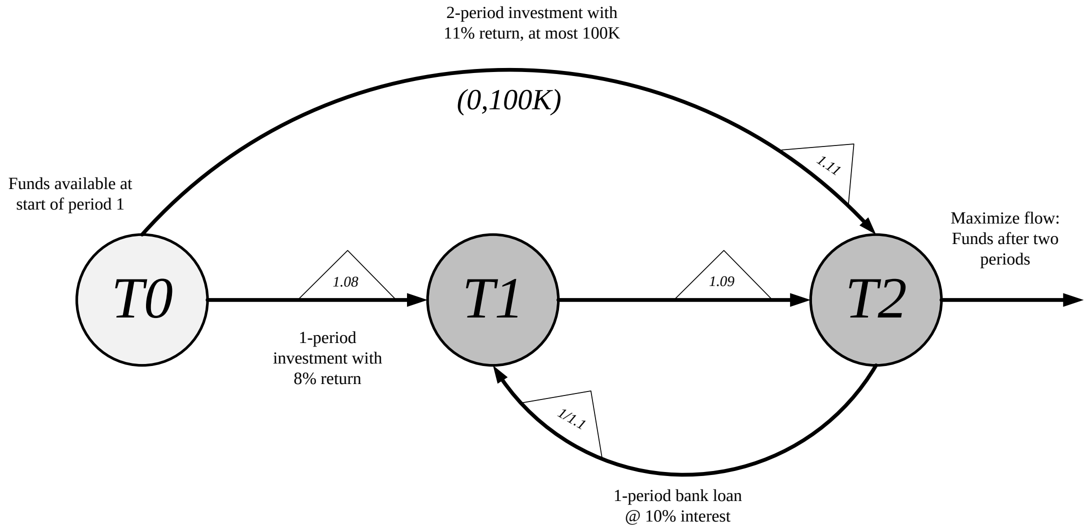



Figure 1.3. Investment Model

In optimization, the generalized network problem represents a linear program whose constraint coefficients also have the form of a node-arc incidence matrix, but without the requirement that two non-zero column entries be $-1$ and $+1$, but rather $-1$ and a positive real number. Generalized networks can be viewed as a practical extension of pure networks that have applications in resource allocation, production, distribution, scheduling, capital budgeting, and other settings [53].

While generalized network flow problems and algorithms have been studied extensively, the extension of linear generalized network models to generalized network problems with fixed charges has received scant attention. For the transportation problem for example, a fixed cost could be incurred for origin-destination shipment. In the facility location problem a fixed amount of investment may result in the establishment or expansion of a plant or warehouse. Production planning problems can be modeled by imposing fixed charges on appropriate arcs of the network. Unlike fixed-charge transportation problems for pure networks, to the best of our knowledge there are no computational results or test problems for fixed-charge generalized transportation problems available. No computational study was found for fixed-charge generalized transshipment network problems in the published literature. The main contribution of this research is to adapt and extend methods and best practices for pure fixed-charge network models and generalized networks to a broader problem class – fixed-charge generalized network flow problems with transportation and transshipment structure variations. In the coming chapters, these models, motivated by practical applications and termed *fixed-charge generalized networks*, are formulated mathematically and solution methods are proposed, implemented, and tested computationally.

## 1.1. Mathematical Formulations

Our research addresses the following problem: an investigation of fixed-charge generalized network problems, including an implementation and computational testing of a series of fixed-charge generalized networks heuristics for transportation and transshipment structures. As background to the discussion, the following sections provide notation, mathematical formulations, and a literature review of applications, algorithms, solution methods, and computer implementations for generalized networks and fixed-charge generalized networks.

### 1.1.1. The Problem's Notations

Following conventional notation[1], a *generalized network* $G(\mathcal{N}, \mathcal{A})$ is a collection of *nodes*, $\mathcal{N}$, *arcs*, $\mathcal{A} = \{(i,j)|i,j \in \mathcal{N}\}$, and associated parameters. The GN problem is a linear program of the form: minimize $\mathbf{cx}$, subject to $\mathbf{Ax} = \mathbf{b}$, $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$. Matrix $\mathbf{A}$ has the form of a node-arc incident matrix defined as in [52, 77]:

$$A_{ik} = \begin{cases} +1 & \text{if arc } k \text{ is directed away from node } i \\ -\mu_k & \text{if arc } k \text{ is directed toward node } i \\ 0 & \text{otherwise} \end{cases}$$

and it has a full row rank. If matrix $\mathbf{A}$ is not a full rank, then the problem corresponds to the pure network problem as shown in [51].

The vector $\mathbf{b} \in \Re^{|\mathcal{N}|}$ is the requirements vector. A supply (demand) at node $i \in \mathcal{N}$ is represented by positive (negative) $b_i$. The decision variable vector $\mathbf{x}$ represents the flows on arcs in the network. The element $x_{ij}$ represents the flow on arc $(i,j) \in \mathcal{A}$. Parameters associated with each arc $(i,j) \in \mathcal{A}$ are: a multiplier $\mu_{ij}$, a variable unit cost $c_{ij}$, an arc capacity or upper bound $u_{ij}$, and an arc lower bound $l_{ij}$. All such parameters, except $c_{ij}$, are assumed to be non-negative unless stated otherwise. Pure network models have $\mu_{ij} = 1$ for all arcs $(i,j)$.

[1] As in [14], matrices are denoted by boldface uppercase Greek or Roman letters, such as $\mathbf{A}, \mathbf{B}, \mathbf{N}, \mathbf{\Phi}$; vectors by boldface lowercase Greek or Roman letters or numerals, such as $\mathbf{a}, \mathbf{b}, \mathbf{x}, \mathbf{1}$, $\boldsymbol{\lambda}$; and scalars and set members by italicized lowercase Greek and Roman letters or numerals that are not boldface, such as $a, b, 1, \varepsilon$. The element in the $i^{th}$ row and $j^{th}$ column of matrix $\mathbf{A}$ is denoted by $a_{ij}$. Sets are denoted by italicized uppercase letters, such as $\mathcal{N}, \mathcal{A}$. All vectors are assumed to have an orientation consistent with their use: all premultiplied vectors are row vectors and all postmultiplied vectors are column vectors. The vector of all zeroes is denoted by $\mathbf{0}$.

### 1.1.2. Generalized Networks

Origins of generalized network models and their solution methods go back to the 1960s when Dantzig presented *The Weighted Distribution Problem*, discussed the linear graph structure of the basis, and proposed a simplex-based algorithm to solve the general weighted distribution problem [30]. The generalized network simplex algorithm was later discussed by Kennington and Helgason [77], Jensen and Barnes [68], Elam, Glover, and Klingman [35], and Brown and McBride [20]. Nonsimplex solution approaches (primal-dual, dual, and relaxation) were developed by Jewell [69], Jensen and Bhaumik [67], and Bertsekas and Tseng [17]. According to the Mathematical Programming Glossary, published by the INFORMS Computing Society [1], a "generalized network is a network in which the flow that reaches the destination could be different from the flow that leaves the source." This can be viewed as an advantage of a generalized network over a pure network in which an arc does not allow flow to change its magnitude from a source node to a sink node.

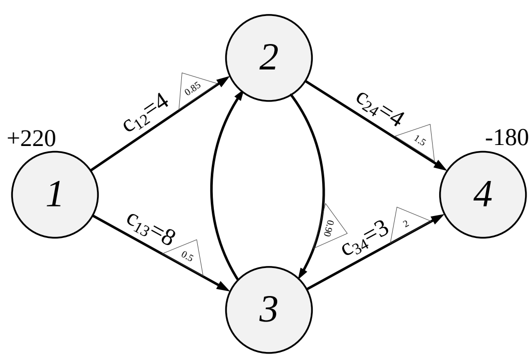



Figure 1.4. Example Generalized Network Flow Model

In the GN formulation's node-arc incidence matrix, instead of $+1$ for a source row $i$ and $-1$ for a destination row $j$, there is $+1$ for a source row $i$ and $-\mu_{ij}$ for a destination row $j$. An example generalized network is depicted in Figure 1.4 where triangles on each arc provide arc multiplier values and the node-arc incidence matrix is the following:

$$\mathcal{A} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ -0.85 & 0 & 1 & 1 & -1 & 0 \\ 0 & -0.5 & -0.9 & 0 & 1 & 1 \\ 0 & 0 & 0 & -1.5 & 0 & -2 \end{bmatrix}$$

Coefficients of change $\mu_{ij}$ are called *multipliers* and arcs with non-unity multipliers are called *generalized* arcs. As assumed in [5], each arc multiplier $\mu_{ij}$ is a positive rational number. If $\mu_{ij}$ is less than 1, the arc is *lossy*; if $\mu_{ij} > 1$, the arc is *gainy*, and if $\mu_{ij} = 1$, it is a pure network arc. Generalized arcs as well as ordinary arcs have attached costs and bounds. Costs and bounds are applied to the original arc flow, before it is transformed by a multiplier [54].

Mathematically, the generalized minimum cost network flow model is defined as:

$$GN: \quad \text{Minimize} \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \tag{1.9}$$

$$\text{subject to:} \sum_{j:(i,j)\in\mathcal{A}} x_{ij} - \sum_{j:(j,i)\in\mathcal{A}} \mu_{ji}x_{ji} = b_i, \forall i \in \mathcal{N} \tag{1.10}$$

$$l_{ij} \le x_{ij} \le u_{ij}, \forall (i,j) \in \mathcal{A} \tag{1.11}$$

The network diagram labeling convention for a GN arc is shown in Figure 1.5. Thus, in a GN node-arc incidence matrix, a column with non-zero components of $+1$ and $-\mu_{ij}$ in rows $i$ and $j$, respectively, corresponds to the generalized arc $(i,j)$. The continuous values are shown with parentheses around lower $l_{ij}$ and upper $u_{ij}$ bounds.

The variable cost is shown above the arc and the multiplier is represented pictorially by attaching it to an arc within a triangle, placed next to the node that receives the flow. To make lower bounds all zeros, the above model can be reformulated via the variable substitution $\overline{x}_{ij} = x_{ij} - l_{ij}$. See, for example, the description in [5, 112].

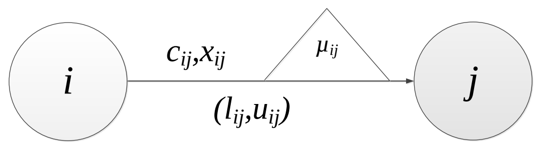



Figure 1.5. Generalized Network Arc

### 1.1.3. Fixed-charge Networks

The *fixed-charge problem* (FCP) is one of the more challenging problems in the area of mathematical programming. The fact that the objective function is piece-wise linear, and a fortiori nonlinear, makes it difficult to apply linear programming methods directly. It is an NP-hard problem [85]. The fixed-charge network flow problem has many practical applications in transportation, network design, plant location problems, production scheduling, investment and distribution problems [13] with the main decision to use an arc in the network or not using it by adding fixed charges on appropriate arcs. The FCP was originally formulated by Hirsch and Dantzig in 1954 [64] and provided two fundamental results: (1) the objective function of FCP is concave and (2) an extreme point optimum exists. A large variety of exact and approximation methods were proposed since to solve the fixed-charge transportation problem for the pure network. As shown in [63], each extreme point is a local minimum for FCP with strictly positive fixed charges, therefore the existence of an extreme point optimum does not assume a straight-forward procedure for its attainment as with pure problem. Chapter 2 provides the literature review for the solution approaches to the fixed-charge problem.

The overall model for the fixed-charge network problem is a mixed-integer program with the following analytical model:

$$
\begin{aligned}
FCP: \quad & \text{Minimize} \sum_{(i,j)\in\mathcal{A}} (c_{ij}x_{ij} + f_{ij}y_{ij}) && && (1.12)\\
\text{subject to:} \quad & \sum_{j:(i,j)\in\mathcal{A}} x_{ij} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji} &=& \; b_i, \forall i \in \mathcal{N} && (1.13)\\
& 0 \le x_{ij} &\le& \; u_{ij}y_{ij}, \forall (i,j) \in \mathcal{A} && (1.14)\\
& 0 \le y_{ij} &\le& \; 1, \forall (i,j) \in \mathcal{A} && (1.15)\\
& y_{ij} & & \; integer, \forall (i,j) \in \mathcal{A} && (1.16)
\end{aligned}
$$

where $y_{ij} = 1$ if arc $(i, j)$ is active, and 0 otherwise and $f_{ij}$ being fixed charge associated with the activation of flow on arc $(i, j)$. The node balance equation maintains the flow balance at each node. The objective function minimizes the overall cost and includes variable and fixed cost as shown in Figure 1.6.

### 1.1.4. Fixed-charge Generalized Networks

The inclusion of fixed charges to generalized network models received small attention in the network flow domain especially in the area of computational studies. Most of available research literature covers fixed-charge transportation networks and addresses the solution methods, exact and approximation, for that type of problems with proposed testbed parameters and generated problems, computational results, and comparisons with state-of-the-art commercial solvers. While some solution approaches and methods proposed for fixed-charge transportation networks can be reasonably applied to the generalized network problems, the modifications should be

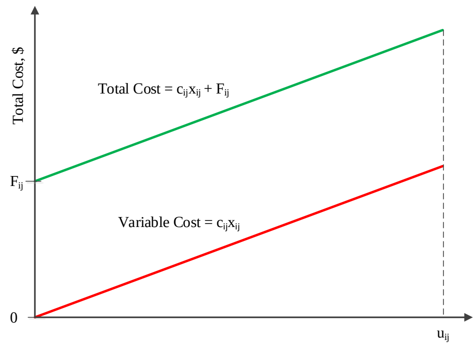



Figure 1.6. Total Cost vs Variable Cost

made due to the structural differences in the basis forest. To fill the gap, this work proposes a new heuristic, suggests a new set of testbed problems, and conducts a computational study for the set of generated *fixed-charge generalized transportation problems* (FCGT). Chapter 2 provides the description of the approach and computational results to FCGT. In addition, to the best of our knowledge, no research provides a computational study for the *fixed-charge generalized transshipment network problem* (FCGN). Chapter 3 of this work proposes meta-heuristic for solving such a class of problems. The overall model for the fixed-charge generalized network problem is a mixed-integer program with the following mathematical model:

$$
\begin{aligned}
FCGN: \quad & \text{Minimize} \sum_{(i,j)\in\mathcal{A}} (c_{ij}x_{ij} + f_{ij}y_{ij}) && (1.17)\\
\text{subject to:} \quad & \sum_{j:(i,j)\in\mathcal{A}} x_{ij} - \sum_{j:(j,i)\in\mathcal{A}} \mu_{ji}x_{ji} = b_i, \forall i \in \mathcal{N} && (1.18)\\
& 0 \le x_{ij} \le u_{ij}y_{ij}, \forall (i,j) \in \mathcal{A} && (1.19)\\
& 0 \le y_{ij} \le 1, \forall (i,j) \in \mathcal{A} && (1.20)\\
& y_{ij} \quad integer, \forall (i,j) \in \mathcal{A} && (1.21)
\end{aligned}
$$

where $y_{ij} = 1$ if arc $(i, j)$ is active, and 0 otherwise and $f_{ij}$ being fixed charge associated with the activation of flow on arc $(i, j)$. The objective function minimizes the overall cost. The node balance equation maintains the flow balance at each node.

### 1.2. Review of Generalized Networks Applications

An important extension of the traditional minimum cost flow problem is the generalized minimum cost flow problem in which each arc has a positive multiplier called a gain or loss factor that transforms the flow on the arc. As stated in [110], the problem has a distinguished history since it was introduced by Kantorovich in his 1939 paper [74], where optimization was justified as a planning and production tool, and mainly when Dantzig [30] extended his network simplex method to handle generalized flow. Other publications provided example applications including machine loading, metal-processing, the aircraft route allocation, financial budgeting, warehousing with "breeding" or "evaporation," catering problems with losses, and the two-equation capacitated linear program [69]. This section describes some well-known applications for generalized networks and fixed-charge generalized networks.

There are numerous real-world applications for generalized networks and fixed-charge generalized networks. Generalized networks have wider application than pure networks due to the fact that arc multipliers allow significantly richer models [66]. Arc multipliers can modify the amount or level of flow [53]. Generalized networks therefore are used to model perishable goods held in inventory, water flowing in irrigation channels, investments gains and losses, electrical power carried on transmission lines, crops planted and harvested, and livestock raised for market. One example of such network with multipliers that change the flow level is the investment problem as shown in Figure 1.7. Glover, Klingman, and Phillips in *Network Models in Optimization and Their Applications in Practice* [54] introduced several applications for generalized networks such as financial models, production and inventory applications, and machine scheduling. These applications all include quantities that can naturally grow or diminish.

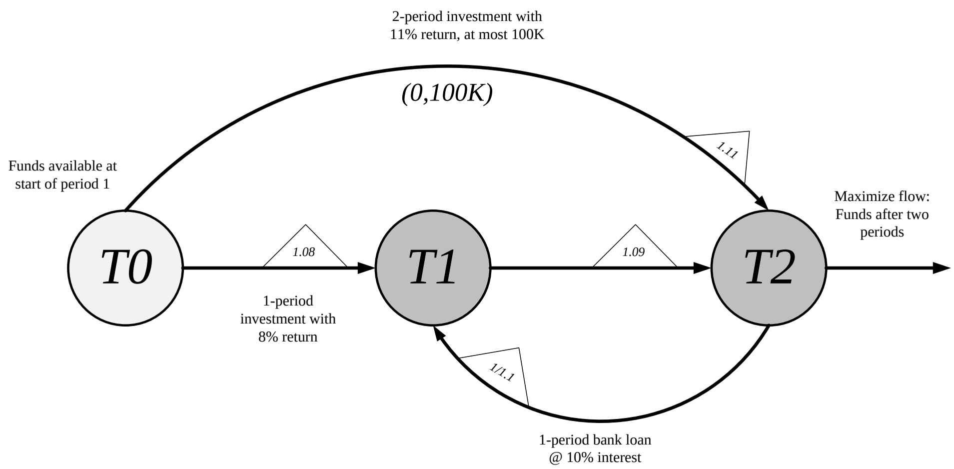



Figure 1.7. Generalized Arc Multipliers that Change Flow Level

Glover, Klingman, and Phillips [54] also discussed the re-expression process, where conceptual transformations rather than physical changes are produced. Examples of such models are a bus or a plane that can be re-expressed in terms of the number of passengers it can carry or a job in terms of hours required to be completed. Arc multipliers are used in such models to transform one type of commodity into another. This interpretation is used to model product manufacturing stages (RM$\rightarrow$WIP$\rightarrow$FG), conversion of water power into electricity, conversion of one type currency for another, and distillation of a liquid to produce a gas.

One of the earliest well-known examples is an application to the Machine Loading problem known as the Generalized Transportation problem. It was introduced by Charnes and Cooper [24] and formally defined by Lourie in [87] as follows: for a given number of different machine types with a specified availability and different products to be produced in a required amount, determine how much of each product to produce on each machine at a minimum total production cost and with satisfied production requirements. Assigning production of machines to products with different efficiencies is the example of using generalized arcs multipliers that transform units of flow as shown in Figure 1.8.

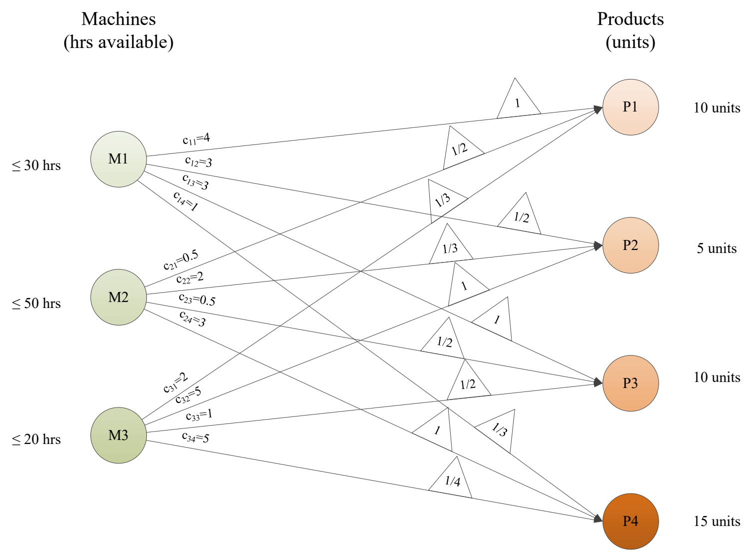



Figure 1.8. Generalized Arc Multipliers that Transform Flow Units

Another example of possible uses of generalized networks is described by the model of a water distribution system with losses concerning the movement of water through canals to various reservoirs and in which multipliers represent the losses from evaporation and seepage [67]. Kim [83] represents copper refining processes by a large D-C electrical network with multipliers representing the appropriate resistance which allows analysis of the effect of short circuits in the refining process. The warehouse funds-flow model by Charnes and Cooper [24] examines multi-period sales, production, and inventory of both products and cash. In this model the multipliers represent conversion rates between cash and products. Mulvey [94] proposed a generalized network model to assign aircraft to high altitude jet routes over the U.S. Another important application is the model for the distribution of natural gas for the Alaskan Pipeline that was reformulated as a generalized network by Glover, Hultz, Klingman, and Stutz [48]. Because gas in the pipeline is used to drive pumps, gas is lost while it moves along the pipelines that can be represented with multipliers for the GN model. See [48] for an extensive list of applications.

The optimization model with a generalized network structure and its application to a generation expansion planning problem was proposed in [81]. The model generates decisions for what types and sizes of generating plants should be brought into an electric power system. The problem is referred to as the Generation Planning Problem (GPP) and is becoming increasingly critical for the electric power sector.

The North American natural gas system is an example of market connected via a pipeline network structure and that includes Canada, the United States, and Mexico [37]. The natural gas system of North America database [37] has 17,000 natural gas reservoirs, each with up to 200 variables, represented in 23 production supply regions. Natural gas is used in residential, commercial, industrial, and electric power consumption sectors.

In the chemical industry, production and manufacturing applications, process design and synthesis, multi-component blended-flow problems, and production planning are suitable for generalized network optimization and were investigated and modeled by Kallrath [73], Kelly [75], Klingman, Mote, and Phillips [84], and Lee [86]. The model developed by Klingman, Mote, and Phillips for one of the nation's largest suppliers of phosphate-based chemical products includes production, distribution, multiperiods, and multi-commodity stages and uses generalized network components [84].

In supply chain management, one is concerned with managing the physical flow of goods (and the virtual flow of information) throughout a network of physically distinct production, distribution, and retail stages [38]. The term flows refers to two types of flows: material and information flows. The overarching goal of supply chain management is to maximize the profitability of the entire supply chain and not that of any one single stage in the chain such as transportation or material handling. Such applications are modeled as generalized networks with multipliers as conversion units.

## 1.3. Solving Generalized Networks

The generalized network flow problem represents a large class of LP problems. Even in the early days of LP research, specialized solution methods were developed to exploit the sparse mathematical structure of these models. In the early 1960s, Dantzig [30] extended his primal network simplex method to generalized networks. Jewell [69] also provided solution techniques for generalized networks, called process-flow networks or flow with gains, using a primal-dual algorithm. In 1954 to solve a transportation problem Charnes and Cooper [23] proposed the stepping-stone method with the procedure to resolve a degeneracy. Authors acknowledged that the stepping-stone method may not be applicable to solving all linear programming problems while other methods, such as the simplex method proposed by Dantzig, can be used. The paper also provided comparison and explanation of both methods to solve a sample transportation problem with three origins, five destinations, and a total of 16 products [23]. The attempt to extend the approach to some generalized network problems was discussed by Charnes and Raike [25] with two one-pass algorithms provided. Hadley [62] made one important observation about generalized transportation problems: because the coefficient matrix is of full rank, division cannot be eliminated and an optimal basic solution may involve fractional values, therefore the integrality property of pure networks does not hold for generalized networks. Following sections discuss the special structures of the generalized networks and outline the main steps of the primal network simplex algorithm for GN.

### 1.3.1. Special Structures of GN

The generalized primal network simplex algorithm, also known as the generalized network simplex algorithm or primal network simplex algorithm for generalized networks, is an adaptation of the linear programming simplex method developed by Dantzig in 1947 [30, 31]. This specialization allows simplex operations to be performed directly on the network graph instead of performing matrix calculations. As stated first by Dantzig in [30], then investigated in early research by Balas and Ivanescu [7, 8], Eisemann [34], Glover and Klingman [55], and Lourie [87], any basis **B** extracted from a generalized network $G$ can be placed in a block-diagonal form by simple permutation of rows and columns:

$$B = \begin{bmatrix} B_1 & & & & & \\ & B_2 & & & & \\ & & \ddots & & & \\ & & & B_i & & \\ & & & & \ddots & \\ & & & & & B_q \end{bmatrix} \tag{1.22}$$

with each square submatrix component $B_i$, $i = 1, 2, ..., q$ being upper triangular or nearly upper triangular with only one element below the diagonal. Furthermore, each component $B_i$ corresponds to a connected subgraph of $G$. Summaries for generalized networks basis structure and algorithms were provided in other papers and textbooks such as [6, 14, 16, 20, 35, 48, 88, 96, 102]. The component or subgraph corresponding to each $B_i$ is a *quasi-tree* or a tree with an added single arc which creates exactly one circular path or loop in the quasi-tree.

The generalized network $G = (\mathcal{N}, \mathcal{A})$ is a directed network that may have self-loops and multiple arcs joining the same pair of nodes with same or different orientation [55]. A *self-loop* is an arc that leads from a node [30, 50] and represents a slack variable in order to change an inequality to an equality. While for pure networks a basis forms a spanning tree of nodes, generalized bases have a more complicated structure and require elaborate processing techniques to fully exploit. For the generalized network problem, its basis forms a forest of quasi-trees where each quasi-tree is either a tree rooted at a slack node, called a *rooted tree*, or a graph with a single cycle, called a *one-tree* [30, 50, 55], as illustrated in Figure 1.9.

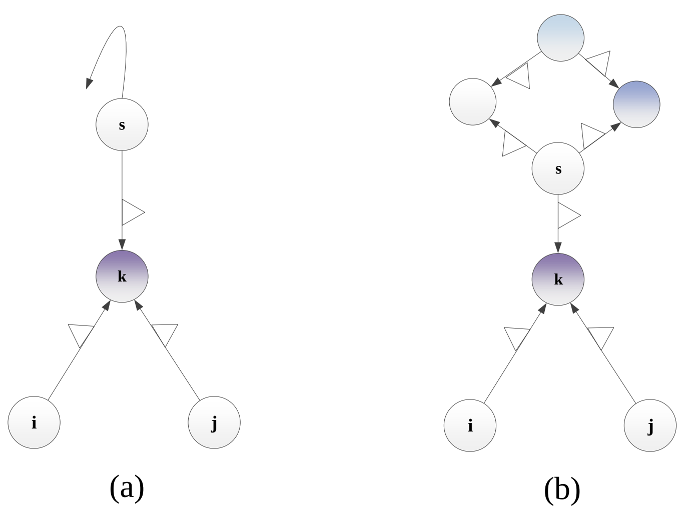



Figure 1.9. Sub-graphs as (a) a rooted tree and (b) an one-tree

Jewell [69] discussed GN bases' special features that are different from pure networks: absorbing and generating loops, which can destroy or create flow, respectively. Though no general formulas to efficiently calculate such flows were provided, it was stated that "these special features make the construction of a special-purpose algorithm an interesting problem" [69]. The explicit formula for calculating flows on a loop in a component of graph $G$ was first provided by Dantzig [30], however it still required the solution of parametric equations, which made computer implementation challenging. Another solution approach discussed by Eisemann [34] considers loops and slacks with a *loop absorption factor*, which summarizes the capacity of the loop to either absorb surplus or to generate a deficiency within the cycle. As for which direction to traverse a loop, Eisemann suggested chosing the direction arbitrarily and then hold it fixed as long as the loop exists [34].

The first efficient methods for determining duals for the simplex pricing-out procedure and the change of basis were proposed by Glover and Klingman [50], using the expression of the loop factor and the loop direction as in [34]. These authors also proposed a method for efficiently updating the basis representation, flows, and dual evaluators in transportation and network optimization problems [55]. Those pivotal additions in algorithm development led to the efficient computer implementation of NETG summarized by Elam, Glover, and Klingman in [35].

The following definitions are adopted for the current discussion:

- **Reduced cost:** There is a *dual variable* or *node potential* $\pi_i$ for each node $i \in \mathcal{N}$ and the *reduced cost* of an arc $(i, j)$ is defined as $c_{ij}^{\pi} = c_{ij} - \pi_i + \mu_{ij}\pi_j$.

- **Loop factor:** For a GN basis loop, its *loop factor* is defined as the ratio:

$$\frac{R}{F} = \frac{\prod \mu_{ij}, \text{for arcs traversed in the reverse direction}}{\prod \mu_{ij}, \text{for arcs traversed in the forward direction}}. \tag{1.23}$$

- **Cycle factor:** A cycle factor used for computing flows and duals for a GN basis quasi-tree is defined as follows:

$$(1 - \frac{R}{F}) = 1 - \text{loop factor} \tag{1.24}$$

### 1.3.2. Primal Network Simplex Algorithm for GN

Generally, the primal network simplex method for pure networks is an iterative procedure that "moves" from one basic solution to an improving or equal-valued adjacent basis (spanning tree structure) until an optimal spanning tree solution is identified [24]. Similarly, the generalized network simplex method "moves", or *pivots*, from one basic solution (forest of quasi-trees structure) to an adjacent basis until an optimal one is located. At every iteration of the generalized network simplex algorithm the following main operations are performed: (1) identify a potentially improving arc to enter the basis, (2) select an arc to leave the current solution based on flow changes, and (3) exchange these variables by updating the solution and the basis forest structure. At every iteration, the generalized network simplex algorithm maintains a feasible basis and transforms it successfully into an improved feasible basic quasi-forest structure until the optimal solution is identified [50].

The generalized network simplex algorithm maintains a partitioning of the arc set $\mathcal{A}$ into $(\mathcal{F}, \mathcal{L}, \mathcal{U})$, called a basis forest structure. The arcs in $\mathcal{F}$ correspond to those in a basis forest of quasi-trees, and the arcs in $\mathcal{L}$ and $\mathcal{U}$ are nonbasic arcs with flow at their lower and upper bounds.

#### *1.3.2.1. Obtaining an Initial Basis Structure*

The primal simplex network algorithm requires a starting basic feasible solution. Start procedures and strategies for solving generalized networks were discussed by Glover, Hultz, Klingman, and Stutz [48], Brown and McBride [20], and Jarvis, Ratliff, and Trick [65]. A starting basis may be constructed (following the "Big M" method) of high-cost artificial arcs that are driven from the solution by the standard simplex pivoting process. An "artificial starting solution" can be constructed with problem arcs to minimize the number of artificial variables that are to be eliminated to achieve feasibility. The disadvantage of the second method is the possibility of spending too much time trying to calculate and determine the optimal basis, while the first method may require more pivots to obtain the optimal solution.

As discussed in [65], conditions for constructing an initial basis are:

- There is one arc for each node in the basis
- Each quasi-tree has exactly one cycle, which may be a self-loop
- The net flows into and out of the node equals the node's supply/demand
- The flow on any non-basic arc is either zero or the arc's capacity
- The flow on any arc is between zero and the arc's capacity

Bixby [18] provides a discussion for constructing an initial starting basis for pure network problems by describing four alternate initial basis approaches. These approaches can be extended for generalized networks.

#### *1.3.2.2. Optimality Testing and Entering Arc*

If the objective function can be improved by changing the flow on a nonbasic arc, that arc is a candidate for entry into the basis. Pivot strategies define the rules for identifying such entering arcs [95]. The earlier defined reduced cost $c_{ij}^{\pi}$ for a non-basic arc is used to determine if optimality has been reached:

- ***Basis Forest Structure Optimality Conditions***. A feasible basic forest structure $(\mathcal{F}, \mathcal{L}, \mathcal{U})$ with the flow $\boldsymbol{x}^*$ is an optimal basis forest structure if for some vector $\boldsymbol{\pi}$ of node potentials, the pair $(\boldsymbol{x}^*, \boldsymbol{\pi})$ satisfies the following optimality conditions:

$$c_{ij}^{\pi} = 0, \forall (i,j) \in \mathcal{F} \tag{1.25}$$

$$c_{ij}^{\pi} \geq 0, \forall (i,j) \in \mathcal{L} \tag{1.26}$$

$$c_{ij}^{\pi} \leq 0, \forall (i,j) \in \mathcal{U} \tag{1.27}$$

There are several pivot rules for entering arc: Dantzig's pivot rule, which selects the arc with the largest in magnitude value of reduced cost, or $|c_{ij}^{\pi}|$; the first eligible arc pivot rule, selects the first arc with nonzero reduced cost encountered in examining the arc list; and the candidate list pivot rule, which maintains a candidate list of arcs with nozero reduced costs and selects the arc with the largest in magnitute reduced cost from the candidate list [49, 93]. Greenberg discussed pivot selection tactics in [61], and also defined sum of infeasibility approach when a basic variable violates its lower or upper bounds. More pivot rules are discussed by Terlaky and Zhang [107].

*1.3.2.3. Identifying the Leaving Arc and Updating the Basis*

The ratio test process finds the representation of the incoming arc with the respect to the current basis and then determines the leaving arc. The algorithmic description can be found in [35, 48, 89]. Because of the GN basis structure, a pivot operation depends on the relation of the incoming and leaving arcs. It can modify the composition of the quasi-trees in a variety of ways. The endpoints of the incoming arc can be in the same quasi-tree or in two separate quasi-trees while at the same time each can be on the quasi-tree's loop or in a tributary tree (trees that arise upon suppression of all quasi-tree loop arcs).

A basic exchange step is to replace the selected outgoing basis arc with the incoming nonbasic variable. Different types of pivots discussed and examples are provided by Ali, Charnes, and Song [6] and Jarvis, Ratliff, and Trick [65].

Survey of generalized networks applications, background material for developing the primal network simplex algorithm for the generalized network flow problems, and details of generalized network simplex algorithm procedures are summarized by Ahuja et al. [5] Textbooks as well as their references such as Kennington and Helgason [77], Jensen and Barnes [68], and Murty [96] provide descriptions of data structures, algorithmic steps, and computational advice for a generalized network algorithm implementation.

The next section reviews the literature on implementations for generalized networks. It also describes terminology, concepts that are now standard in network programming literature [5, 14], and data structures for efficient algorithm implementation. Chapter 2 provides the literature review for fixed-charge problem solution approaches.

## 1.4. Literature Review of GN Implementations

The literature review discusses computationally efficient algorithms and implementations for generalized networks. The first specialized software for GN problem was developed in 1970's. As discussed by Kennington and Lewis in the *Encyclopedia of Optimization*, many of the computer codes that have been developed for GN are specializations of the *primal simplex algorithm* which exploit the graphical structure of the basis and solve system of equations on a graph rather that with matrix operations [78]. Survey of generalized network codes with a partial list with names, a year developed, authors, and the language (Assembly, FORTRAN, and C) can be found in [78] and in Table 1.1.

Table 1.1. Partial List of Generalized Network Codes [78]

| Code | Year | Language | Authors |
|---|---|---|---|
| NETG | 1973 | FORTRAN | F.Glover, D.Klingman, J.Stutz [48] |
| GENNET | 1984 | FORTRAN | G.Brown, R.McBride [20] |
| GRNET | 1985 | FORTRAN | M.Engquist, M.Chang [36] |
| LPNETG | 1985 | FORTRAN | J.Mulvey, S.Zenios [95] |
| ACS | 1986 | FORTRAN | I.Ali, A.Charnes, T.Song [6] |
| GNO-PC | 1988 | C | W.Nulty, M.Trick [98] |
| GENFLO(parallel, primal) | 1989 | FORTRAN | M.Ramamurti [102] |
| NETPD(primal-dual) | 1994 | FORTRAN | N.Curet [29] |
| RAMSES(dual) | 1997 | C | J.Kennington, R.Mohamed [80] |

The network primal simplex algorithm can be used to solve special cases of the minimum cost network flow models and has the advantage of solving a broad class of problems [88]. It has proven to be extremely effective in solving large scale network flow problems. The generalized network primal simplex algorithm is similar to the primal network simplex algorithm for pure networks. It maintains a good feasible basis structure at every iteration and by pivoting transforms the solution into improved basis structure. Orlin [100] has shown that using the entering variable with a minimum reduced cost and following a lexicographic rule for the leaving variable as proposed independently by Charnes [22] and Dantzig, Orden, and Wolfe in [31], then summarized by Terlaky in *Encyclopedia of Optimization* [106], the maximum number of pivots for the assignment or the shortest path problem and the maximum number of consecutive degenerate pivots for the minimum cost network flow problem is $\mathcal{O}(|\mathcal{N}|^2|\mathcal{A}|\log|\mathcal{N}|)$ for a directed graph $G = (\mathcal{N}, \mathcal{A})$.

The primal simplex algorithm can be specialized for generalized networks with a basis represented graphically as a collection of *quasi-trees*. As mentioned in [77] and

[80], the key operations for the network simplex algorithm implementations can be directly performed on the quasi-forest using appropriate data structures. While data structure should distinguish between subset of nodes (self-loops and one-trees), it also facilitates the three essential operations of a primal simplex pivot such as 1) pricing, or determination of an entering arc, 2) the representation of this incoming arc with respect to the current basis and the ratio test operation to determine the leaving arc, and 3) the updating the basis structure, flows, and node duals when needed. The data structure first proposed by Johnson in 1966 is known as the triple-label or augmented predecessor index (API) method [71]. For this method each node has three labels: predecessor, successor, or down-left, and brother, or right, if the root node is pictured as being at the top of the tree. An improved data structure, called the augmented threaded index (ATI) method, was discussed by Glover, Klingman, and Stutz [55] with computational simplifications in [50, 56], then also by Elam, Glover, and Klingman [35] and Glover, Hultz, Stutz, and Klingman [45, 48]. Other data structures were later presented by Ali, Charnes, and Song [6], Brown and McBride [20], Engquist and Chang [36], and Jarvis, Ratliff, and Trick [65].

The design of a solution method necessarily depends on the data structure chosen to represent the basis. For this research a basis for a generalized network is maintained as a quasi-forest using the following node labels: predecessor, thread, reversed thread, depth, number of nodes in quasi-tree $T_i$ with a root $i$, last node in quasi-tree $T_i$, and a loop factor value if a node is on a cycle of an one-tree root. As for arcs, an arc in the form $(i, pred(i))$ is considered to be the forward arc with a multiplier $\mu_{i,pred(i)}$, while $(pred(i), i)$ is the reverse arc with a multiplier $\mu_{pred(i),i}$. There is a link for each node to the corresponding basic arc and a list of all network arcs together with their data, such as from-node, to-node, cost, multiplier, and upper and lower bounds. Next paragraphs discuss implementations and their contributions to the field.

The first efficient implementation of the primal simplex algorithm for GN problems was developed by Elam, Glover, and Klingman in [35, 50, 55] known as a strong convergent primal simplex algorithm for generalized networks. The developed code NETG written in FORTRAN exploits the special structure of a basis using the *Extended Augmented Predecessor Index* (EAPI) method and it is based on the triple label representation for trees introduced by Johnson [71] for pure networks. Glover, Klingman, and Stutz [56] utilized a preorder thread index in their *Augmented Threaded Index* method for pure networks and showed that it is more efficient than triple-label representation with less storage space required. By using simple ordered lists, NETG only stores cost parameters, multipliers, and upper bounds values for each arc, or a column of the coefficients matrix. The main advantage of using these ordered lists is that there is no need to determine and store the inverse of the basis matrix which usually requires computations and storage to maintain and update.

The algorithm address the degeneracy problem that arises in the attempt to solve GN and GN-related problems by updating and maintaining the EAPI basis structure that also ensures convergence of the algorithms. The strongly convergent primal simplex algorithm for GN was proposed with the specifications for efficient implementations procedures of determining and updating duals as described in [50, 55]. The contributions of the algorithm are: all bases have the special topological structure, the algorithm is finitely convergent without reliance on techniques such as lexicography or perturbation, and a special screening criterion is available for non-degenerate basis exchanges. These mathematical differences over the original simplex methods also contributed to several computational advantages over other codes developed for solving GN problems.

The development of NETG demonstrated an advantage of representing and solving GN on graphs such as ability to characterize 1) the nonzero elements of the representation of an incoming nonbasic arc and the signs of these elements to identify the leaving arc efficiently and 2) which node potentials to recalculate and how these values are altered after the basis pivot. By investigating rules for the starting strategy, the pivot selection criteria, and degeneracy handling, Glover, Hultz, Klingman, and Stutz [48] enhanced NETG in 1978. For the testing purposes, a generalized network problem generator (NETGENG) was developed with all parameters retained and the ability to specify the range of values for arc multipliers. The problem varied in size up to 1,000 nodes and 7,000 arcs with the complete specifications and test results discussed in [47]. In addition, factors affecting solution speed such as start procedures, pivot selection techniques, degeneracy, tolerance level, the "Big-M" value, and pivot tie-breaking rules were computationally tested within NETG. Solution strategies and computational tests results are provided in [47, 48]. In many later implementations, the code NETG is considered the state-of-the-art algorithm and NETGENG is used to generate problems for test purposes.

In 1984, the Augmented Threaded Index method was used in GENNET written in FORTRAN by Brown and McBride [20] with a description of an efficient primal simplex method for the generalized network problem. Benchmark problems tested have up to 1,000 nodes and up to 7,000 arcs for generalized randomly generated with NETGENG network problems and compared with NETG. The algorithmic approach is based on a pre-order traversal method in addition to the predecessor, depth, and cycle factor to represent the basis. For enhanced algorithmic performance, Brown and McBride included 1) a dual basis aggregation technique that maintained explicit values of depth, dual, and pre-order traversal labels only for nodes with successors, 2) a dynamic candidate queue that is a dynamic list of interesting arcs and nodes scanned in a cyclic manner to chose an entering arc [20], and 3) a starting strategy using the "Big-M" method. Authors claimed GENNET has proven to be a worthy successor of GNET with more efficient performance on real models than on randomly generated test problems of nominally equivalent size.

Orlin [100] stated that it has been established by Elam et al. [35] and Brown and McBride [20] that in practice simplex-type procedures are very efficient for solving generalized network flow problems. Orlin also showed equivalency of the "strongly convergent" pivot rule developed in [35] with the lexicographic rule [22, 31, 106] for avoiding cycling.

In 1985, Engquist and Chang [36] provided a brief description for implementing GRNET written in FORTRAN, the primal simplex code for generalized networks that builds on the labeling procedures used in one of the fastest pure network codes developed by Barr, Glover, and Klingman [10]. GRNET implementation also uses the thread as in GENNET by Brown and McBride, however it does not make use of the rooted loop orientation as in NETG [35, 50] in the quasi-tree representation. The method of such representation is useful when applied to processing networks. Five cases for updating the basis graph are discussed based on the location for end nodes of incoming and leaving arcs. Also some techniques to increase updating efficiently are considered such as retracing the predecessor path of the incoming arc while updating flows on arcs. GRNET employs the reverse thread function, which eliminates searching in the basis update, and also uses an artificial start basis while the advanced start could improve the solution time [36]. The testing and comparison with MINOS on ten problems generated with NETGENG shows that GRNET is about 60 times faster than MINOS.

Mulvey and Zenios [95] investigated the efficiency of different internal programming techniques, such as pivot strategies, column normalization factors, and the "Big-M" starting method for the primal simplex generalized network code LPNETG written in FORTRAN. The underlying primal simplex algorithm used in LPNETG is described by Kennington and Helgason [77], and McBride [89]. A detailed procedure for generating the input parameters for NETGENG is provided to allow the reproducibility of the experiment. Mulvey and Zenios stated "a computer code testing cannot be considered complete unless it establishes the efficiency of the program with respect to other algorithms." The program comparisons revealed that, on average, LPNETG was 18% faster than NETG, while for some of the smallest problems NETG was more efficient, which indicated mostly that with the basic algorithmic approach unchanged some internal programming techniques may improve the efficiency of a computer code.

In 1986 Ali, Charnes, and Song [6] computationally tested their code written in FORTRAN to establish the suitability of the data structures for efficient implementation and provided the detailed algorithmic specification of the primal simplex algorithm for the generalized network problem. Testing indicated that generalized network algorithms are on the order of 2.5 to 3.5 times slower than pure network algorithms.

In 1988, Nulty and Trick [98] developed the first primal simplex code written in the C language that uses the predecessor, thread, reverse thread, and the level node labels presented in Kennington and Helgason [77] to represent the basis.

While the primal simplex method has been computationally superior to primal-dual simplex and out-of-kilter methods for solving large-scale generalized network linear programs, NETPD written in FORTRAN was developed by Curet in 1994 [29] with a specialization of the dual simplex algorithm. It employs a dynamically sized subbasis matrix to monotonically decrease the number of infeasible node constraints while simultaneously optimizing a dual program.

In 1997, a specialization of the dual simplex algorithm for the generalized network problem that uses a dual two phase method along with efficient dual partial pricing schemes and specialized routines for the dual ratio test was implemented in RAMSES written in the C language by Kennington and Mohamed [80].

The relaxation method proposed by Bertsekas and Tseng in 1988 also has been extended for the generalized network problem [17]. The algorithm adopts the non-linear programming relaxation method based on the iterative improvement of the dual cost while maintaining a flow that satisfies complementary slackness. The first fast combinatorial solution to the generalized minimum cost flow problem that provided strongly polynomial approximation schemes was proposed by Wayne [111]. The algorithm solves the generalized circulation problem (also knows as the generalized maximum flow problem) with supplies and demands being zero. Though it was shown that the best interior point methods are faster for computing optimal solutions, the proposed combinatorial algorithm proved to be faster for computing approximate optimal solutions for large problems. While the minimum ratio and the scaling minimum ratio circuit-canceling algorithms were discussed, no implementation details or computational results were provided. As for parallel algorithms, GENFLO proposed by Ramamurti [102] and GRNET2 developed by Clark and Meyer [26, 27] for solving generalized networks used a gradient penalty method to find a starting feasible solution.

In summary, while different solution methods based on the primal network simplex algorithm and heuristics were proposed, implemented and tested for GN and for pure networks, currently to the best of our knowledge there has been no solution method proposed and computationally tested for fixed-charge generalized networks (FCGN).

## 1.5. Expected Contributions and Algorithm Approach

The extensive literature review of GN applications, algorithms, solution methods, and implementation in previous sections revealed that currently there is no computational testing results available for fixed-charge generalized networks. Although several papers have been presented discussing the algorithm development and software implementations for solving generalized networks, none of these consider the case of fixed-charges on some or all of the arcs. This research investigates a type of network problems, called the *fixed-charge generalized network* problem, which is an extension of the classic generalized network formulation that adds fixed-charges to the arcs, solves models with proposed meta-heuristics, and provides computational results.

The expected contribution of this research is to (1) develop a heuristic approach that incorporates the fixed-charge into the simplex pivoting process for the fixed-charge generalized transportation networks, (2) expand the approach to solve fixed-charge generalized transshipment networks of large size with up to $10,000$ nodes and $100,000$ arcs, (3) propose testbed problems parameters for FCGT, compatible with testbed problems for classical pure fixed-charge transportation networks, and testbed problems for transshipment structures, (4) conduct computational studies by testing the heuristic algorithms on newly created testbed problems and compare results with exact solutions, when possible, for both transportation and transshipment structures. It is expected that the proposed heuristics will be effective in terms of solution quality with dramatically faster processing speed compared to more general solution methods.

The algorithm approach for solving FCGT and FCGN problems in this research builds on extreme-point tabu search ideas developed by Walker [109] and investigated in [15, 19, 72, 44, 105, 104]. The algorithm extends the approach used for fixed-charge transportation problems in [44, 104] to develop heuristics for fixed-charge generalized networks. It uses the primal network simplex method for generalized networks by exploiting the special structure of the problem bases, including quasi-tree forest solution representation and a streamlined processing of the simplex steps, such as pricing of the entering arc, ratio test, and pivoting. The code that solves generalized network relaxation is developed in FORTRAN by the author. The following chapters describe proposed heuristic solution approaches, new sets of parameters and testbed generated instances for generalized transportation and transshipment problems, and provide detailed statistical analysis of computational results and a comparison between proposed heuristics and the state-of-the-art commercial software CPLEX.

# Chapter 2

# Extreme-point Tabu Search Heuristic for Solving Fixed-charge Generalized Transportation Problems

This chapter focuses on solution methods for generalized transportation problems that involve fixed costs. Generalized networks have enjoyed a wider and richer range of applications than pure networks because their arcs' flow multipliers can modify the amount or level of flow [53, 66]. The addition of fixed charges to arcs further expands these modeling capabilities and fills a gap in published research.

The fixed-charge generalized transportation problem (FCGT) is a bipartite network consisting of a set of source or origin nodes ($\mathcal{O}$), each with supply $a_i, i \in \mathcal{O}$, and a set of sink or destination nodes ($\mathcal{D}$), each with a demand $b_j, j \in \mathcal{D}$ for a single commodity. Each source is connected to one or more sinks by directed arcs ($\mathcal{A}$) with flow multipliers, upper bounds, variable costs, and fixed costs, which are assessed if the arc has positive flow. The objective is to route the flow from the source nodes' supplies across arcs to meet the sinks' demand requirements at minimum total cost, as follows.

$$
\begin{aligned}
FCGT: \qquad \text{Minimize} \sum_{i\in\mathcal{O}}\sum_{j\in\mathcal{D}}(c_{ij}x_{ij}+f_{ij}y_{ij}) &= FC(x) && (2.1)\\
\text{subject to: } \sum_{j\in\mathcal{D}} x_{ij} &\leq a_i, \forall i\in\mathcal{O} && (2.2)\\
\sum_{i\in\mathcal{O}} \mu_{ij}x_{ij} &= b_j, \forall j\in\mathcal{D} && (2.3)\\
0 \leq x_{ij} &\leq u_{ij}y_{ij}, \forall (i,j)\in\mathcal{A} && (2.4)\\
0 \leq y_{ij} &\leq 1, \forall (i,j)\in\mathcal{A} && (2.5)\\
y_{ij} & \quad integer, \forall (i,j)\in\mathcal{A} && (2.6)
\end{aligned}
$$

where for each arc $(i,j)\in\mathcal{A}$, $x_{ij}$ is its flow, $\mu_{ij}$ is the flow multiplier, $u_{ij}$ the upper bound, $c_{ij}$ the unit cost, $f_{ij}$ the fixed cost, and binary variable $y_{ij}=1$ if the arc is active or 0 otherwise. The objective function (2.1) minimizes the total fixed and variable cost. The node balance equations, (2.2) and (2.3), maintain the flow balance at each node, while (2.4) enforces the flow bounds and connects arc activity to the fixed charges.

In the absence of published algorithms for FCGT, we develop an extreme-point Tabu-search (EPTS) heuristic solution algorithm that capitalizes on and exploits the special structure of a simplex basis for the Balinski's linear approximation [9], GT:

$$
\begin{aligned}
GT: \qquad \text{Minimize} \sum_{i\in\mathcal{O}}\sum_{j\in\mathcal{D}} x_{ij}(c_{ij}+f_{ij}/u_{ij}) &= LC(x) && (2.7)\\
\text{subject to: } \sum_{j\in\mathcal{D}} x_{ij} &\leq a_i, \forall i\in\mathcal{O} && (2.8)\\
\sum_{i\in\mathcal{O}} \mu_{ij}x_{ij} &= b_j, \forall j\in\mathcal{D} && (2.9)\\
0 \leq x_{ij} &\leq u_{ij}, \forall (i,j)\in\mathcal{A} && (2.10)
\end{aligned}
$$

Computational testing demonstrates the effectiveness of the EPTS approach by identifying near-optimal solutions over five orders of magnitude faster than a state-of-the-art commercial optimizer. The methodology relies on many components of the primal simplex method as specialized for generalized transportation problems [50, 55].

The sections below present example applications for GT and FCGT, a summary of the primal simplex method for linear generalized transportation problems, the EPTS heuristic for FCGT that is based on this algorithm, and the results of computational testing.

## 2.1. Applications and Algorithmic History

An early well-known example of GT is its application to a machine loading problem introduced by Charnes and Cooper [24] and formally defined by Lourie in [87] as follows: for a given number of different machine types with a specified availability and different products to be produced in a specified amount, determine how much of each product to produce on each machine at a minimum total production cost and with satisfied production requirements. Assigning production of products to machines with different efficiencies is the example of using generalized arcs multipliers that transform units of flow from hours into finished products, as shown in Figure 2.1.

Linear generalized networks have been also used to model perishable goods held in inventory, water flowing in irrigation channels, investments gains and losses, electrical power carried on transmission lines, crops planted and harvested, and livestock raised for market [5, 45, 46, 48, 54]. Extensive modeling techniques and example applications can be found in Glover et al. [54].

Generalized network algorithms have been devised to exploit the special structure of their simplex bases as documented in [6, 20, 35, 55, 80, 77, 96].

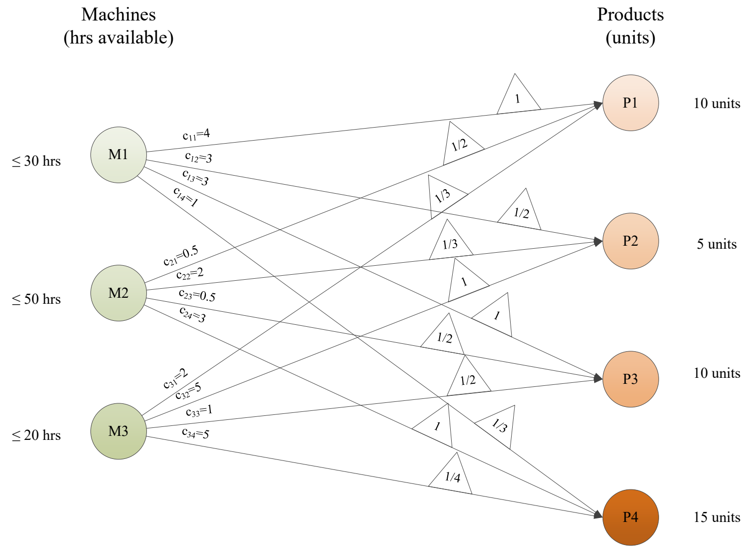



Figure 2.1. Generalized Arc Multipliers that Transform Flow Units

Although fixed-charge models can be formulated as a mixed integer programming problem and solved using general-purpose exact methods, the exponential growth of the required computational effort limits their range of application. Two approaches to ameliorating such barriers have been the development of specialized exact algorithms for certain problem classes and the creation of heuristic approximation algorithms. Both have been successful, particularly methods for network-related classes, and heuristics have dramatically extended the addressable problem sizes and the range of fixed charges, but only for pure networks (without flow multipliers).

Exact methods for fixed-charge pure networks have been studied extensively since Hirsch and Dantzig [64] observed that adding fixed charges to the objective function makes it difficult to directly apply linear programming methods. They proved that for the case of non-negative fixed charges, the objective function is concave and the optimal solutions occur at extreme points and therefore the search for the optimal solution may be restricted to the extreme points of the feasible region. The first exact method for the fixed-charge transportation problem (FCTP) used extreme point ranking and was proposed by Murty [97] and formalized by Gray [60]. Degeneracy issues were addressed in [90] by proposing a modified vertex ranking procedure. Exact methods proposed over the years included dynamic programming [82] and more often algorithms were based on branch-and-bound methods [13, 79, 91, 92, 101]. Recently, Roberti et al. in [103] proposed an exact branch-and-price algorithm based on a a new integer programming formulation and tested it on randomly generated FCTP involving up to 70 sources and 70 sinks.

The practical limitations of computational effort for solving problems by exact methods led to the development of approximation approaches [9], heuristics and meta-heuristics [108]. One of the strategies, to employ an extreme point search technique, was discussed by Walker [109]. Another heuristic approach for FCTP was described by Sun and McKeown [105] that used a tabu search approach. Tabu search as a meta-heuristic was introduced by Glover and developed by him over several years [39, 41, 59, 43, 57, 58]. Adlakha and Kowalski [2] proposed a simple heuristic for small FCTP problems. Later authors provided a new approximation method of obtaining lower bounds for the optimal solution within a 5% error as compared to CPLEX for 10 x 10 and 10 x 15 size problems [3, 4]. A successive linear approximation procedure for generalized fixed-charge transportation problems with resource losses in the situations of evaporation with liquid commodity, heat losses in an electrical distribution network, or deterioration losses with perishable commodities such as food items was proposed by Diaby in [33]. The procedure is not based on extreme point enumeration and consists of solving a sequence of pro-rated problems. Using preprocessing of the data and/or adding set covering constraints to strengthen the formulation was suggested by McKeown and Ragsdale in [92].

Two of the most successful papers with computational results are from Sun [104] and Glover et al. [44], both of which are based on extreme-point search. Glover et al. developed a parametric ghost image process heuristic which to the best of our knowledge represents the state-of-the-art heuristic method for FCTP [42, 44] and uses meta-heuristics to search for a better extreme point non-adjacent to the current optimal solution. The iterated local search heuristic based on utilization of reduced costs for guiding the restart phase was proposed recently in [21] for fixed-charge transportation problems.

The fixed-charge pure networks have characteristics that can be readily exploited by both exact and heuristic methods. The most notable are the less-than-full-row rank of their basic solutions and the resultant spanning tree structure of their bases. As detailed below, generalized transportation bases have a more complicated structure and require more elaborate processing techniques to fully exploit.

## 2.2. Characteristics of Generalized Transportation Problems

Even in the early days of linear programming research, specialized solution methods for linear network flow problems were developed to exploit the sparse mathematical structure of these models. The most successful methods have been based on Dantzig's primal simplex algorithm, which starts with a basic solution and proceeds through a series of bases until the optimal solution is identified.

As stated first by Dantzig in [30], then investigated in early research by Balas and Ivanescu [7, 8], Eisemann [34], Glover and Klingman [55], and Lourie [87], any basis $\mathbf{B}$ extracted from a generalized network $G_N = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of network nodes, can be placed in a block-diagonal form by simple permutation of rows and columns:

$$B = \begin{bmatrix} B_1 & & & & & \\ & B_2 & & & & \\ & & \ddots & & & \\ & & & B_i & & \\ & & & & \ddots & \\ & & & & & B_q \end{bmatrix} \tag{2.11}$$

with each square submatrix component $B_i$, $i = 1, 2, ..., q$, being upper triangular or nearly upper triangular with only one element below the diagonal. Furthermore, each component $B_i$ corresponds to a connected subgraph of $G_N$. Summaries for generalized networks basis structure and algorithms are provided in other papers and textbooks such as [6, 14, 16, 20, 35, 48, 88, 96, 102].

The generalized transportation problem $GT = (\mathcal{O}, \mathcal{D}, \mathcal{A})$ is a directed network that may have self-loops and multiple arcs joining the same pair of nodes with the same orientation [55]. A *self-loop* is an arc that leads from a node [30, 50] and represents a slack variable for (2.2) in order to change an inequality to an equality. For GT, its basis $B$ forms a forest of quasi-trees, or trees with one additional arc; each component quasi-tree, $B_i$, is either a tree rooted at a slack node, called a *rooted tree*, or a graph with a single cycle, called a *one-tree* [30, 50, 55], as illustrated in Figure 2.2.

As Jewell [69] observed, these basis quasi-trees have absorbing and generating loops that can destroy or create flow, respectively. The method for calculating flows on such loops was first provided by Dantzig [30], however it still required the solution of parametric equations. Eisemann [34] characterized these loops and slacks with a *loop absorption factor*, reflecting the loop's capacity to either absorb surplus or to

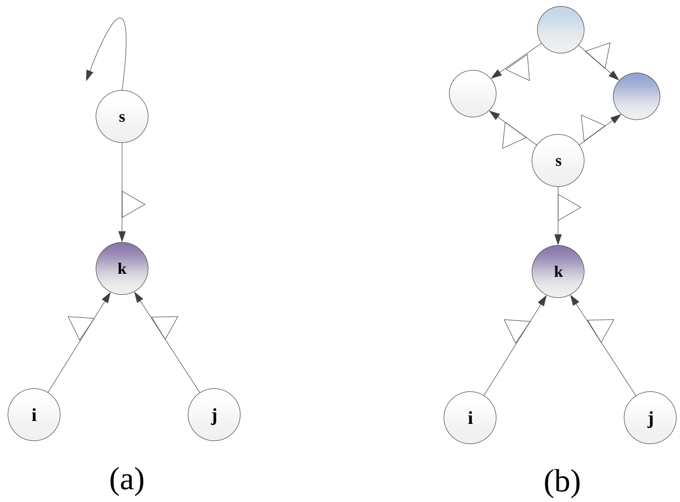



Figure 2.2. Sub-graphs as (a) a rooted tree and (b) an one-tree

generate a deficiency within the cycle [34].

These characteristics have been studied extensively and used within the primal simplex method for generalized transportation problems. This algorithm is summarized in the following sections as a prelude to a description of the EPS heuristic for problems with fixed costs on the arcs.

## 2.3. Primal Simplex for Generalized Transportation Problems

The primal network simplex method for generalized networks solves GNs by traversing a sequence of basic solutions until the optimal solution is reached. A GN problem has full row rank and the mathematical structure of a basic solution is that of a forest of quasi-trees, which can be manipulated to streamline computer implementation.

The algorithm begins by constructing an initial basic solution and iterating through a sequence of adjacent bases until reaching the optimal solution. A pivot operation performs the change of basis by removing a currently basic arc, adding a nonbasic one, and adjusting the flows accordingly. The main steps of an iteration are:

1. *Pricing and selection.* Price out nonbasic arcs ($NB$) to identify attractive candidates for inclusion in the current basis. Select one as the *incoming arc* $a^+ \in NB$ that could improve the current solution value. If none are found, the current basis is optimal; stop.

2. *Ratio test.* Using the representation of $a^+$ with respect to the current basis ($B$), apply the ratio test to identify the *leaving arc* $a^- \in B$, and $\delta$, the level of flow increase on $a^+$ that forces $a^-$'s flow to zero.

3. *Pivot.* Execute a pivot to update the basis flows resulting from adding the incoming arc at level $\delta$, adjusting the flows in its representation, removing the leaving arc, and updating $B$'s forest of quasi-trees.

### 2.3.1. Representation of a Nonbasic Arc

The *representation* of a non-basic arc defines that variable's equivalence to a linear combination of variables in the current basis. This concept is used throughout the primal simplex method for generalized transportation problems, including pricing, the ratio test, pivoting, and updating an implementation's data structures. It is also central to the EPTS heuristic developed later.

The problem of finding the representation of a non-basic arc $(i, j)$ in terms of the basis arcs is computationally equivalent to finding the flow decreases on the current basic arcs to satisfy node requirements of $-1$ unit of available supply at node $i \in \mathcal{O}$ and $-\mu_{ij}$ units of demand at node $j \in \mathcal{D}$. (Flow increases are negative in a representation.) Thus the subgraphs containing nodes $i$ and $j$ are identified to compute the flow changes in each to satisfy the indicated requirements. The affected portions of those subgraphs form the *basis equivalent path* (BEP) for the entering arc. Algorithm 2.1 identifies $(i, j)$'s basis equivalent path and basis representation [50].

---

**Algorithm 2.1** Identify nonbasic arc $(i, j)$'s basis equivalent path and representation, $\tilde{A}_{ij}$.

---

1: Assume arc $(i, j)$ is nonbasic at its lower bound. Consider the effect of increasing its flow by 1 unit.[a]

2: At source node $i$, one unit of flow is withdrawn from the current basis and the basic arcs' flows in the quasi-tree's unique path from $i$ to its root node $k$ are adjusted accordingly, resulting in a deficit or surplus of flow, $\lambda_k$, at $k$.

3: The $\lambda_k$ flow adjustment is accommodated by a flow change in the root arc at $k$ or changes in arcs of the one-tree's cycle, which passes through $k$.

4: Similarly, $\mu_{ij}$ units of flow are injected at sink node $j$ and the basic arcs' flows in the quasi-tree's unique path from $j$ to its root node $\ell$ are adjusted accordingly, resulting in a deficit or surplus of flow, $\lambda_\ell$, at $\ell$.

5: The $\lambda_\ell$ flow adjustment is accommodated by a flow change in the root arc at $\ell$ or changes in arcs of the one-tree's cycle, which passes through $\ell$.

6: The combined set of arcs affected by steps 1–5 form the basis equivalent path of $(i, j)$. The representation of $(i, j)$ in terms of $B$ is the vector of flow reductions (negative if flow increases) in the BEP's arcs. [*Note that steps 3 and 5 will both adjust some of the same arc(s) if the two quasi-trees are not distinct.*]

[a] Arcs that are nonbasic at their upper bounds can be evaluated similarly, with flow injected into the basis at node $i$ and withdrawn at node $j$.

---

For example, Figures 2.3 and 2.4 show different cases for endpoints of the entering arc $(i, j)$—either having a common root in the same one-tree component or different roots in the case of endpoints of entering arc being in two different components.

Thicker black arcs indicate the basis equivalent path arcs.

Figure 2.5 shows an example basis quasi-tree with flows and calculations for determining the representation $\tilde{A}_{ij}$ of nonbasic arc $(i, j)$ with flow $x_{ij}$. This is constructed from the basic arcs in the two paths $P_i$ and $P_j$ that start, respectively, at nodes $i$ and $j$ and end at their common root at node $s$ and whose union forms the basis equivalent path of $(i, j)$.

In this example, the effects of increasing flow $x_{ij}$ by 1 on incoming arc $(i, j)$ are: (a) on path $P_i$, change flow $x_{ik}$ by $+1$, $x_{sk}$ by $+8/4$, and $x_{ss}$ by $-8/4$; and (b) on path $P_j$, to change $x_{mj}$ by $-2/4$, $x_{mk}$ by $+1/2$, $x_{sk}$ by $-5/4$, and the $x_{ss}$ by $+5/4$. By combining these calculations, the representation $\tilde{A}_{ij}$ of arc $(i, j)$ with respect to basis $\mathbf{x}_B$ is shown to be:

$$\mathbf{x}_B = \begin{pmatrix} x_{sk} \\ x_{ik} \\ x_{mj} \\ x_{mk} \\ x_{ss} \end{pmatrix}, \tilde{A}_{ij} = \begin{pmatrix} -8/4 \\ 1 \\ 0 \\ 0 \\ 8/4 \end{pmatrix} + \begin{pmatrix} 5/4 \\ 0 \\ 2/4 \\ -1/2 \\ -5/4 \end{pmatrix} = \begin{pmatrix} -3/4 \\ 1 \\ 1/2 \\ -1/2 \\ 3/4 \end{pmatrix}$$

### 2.3.2. Step 1: Pricing and Selection

Pricing a nonbasic variable with zero flow involves determining the marginal change in the solution value if its level is increased by one and its representation is adjusted correspondingly. For linear generalized networks, pricing a nonbasic arc $(i, j)$ to determine its $\overline{c}_{ij}$ can be accomplished two ways:

1. *Tracing.* Using Algorithm 2.1, trace $(i, j)$'s BEP and accumulate the arcs' costs (adjusted by the flow multipliers) to determine the total impact on the basis' variable cost, $\alpha_{ij}$. The marginal cost is then $\overline{C}_{ij} = C_{ij} - \alpha_{ij}$.

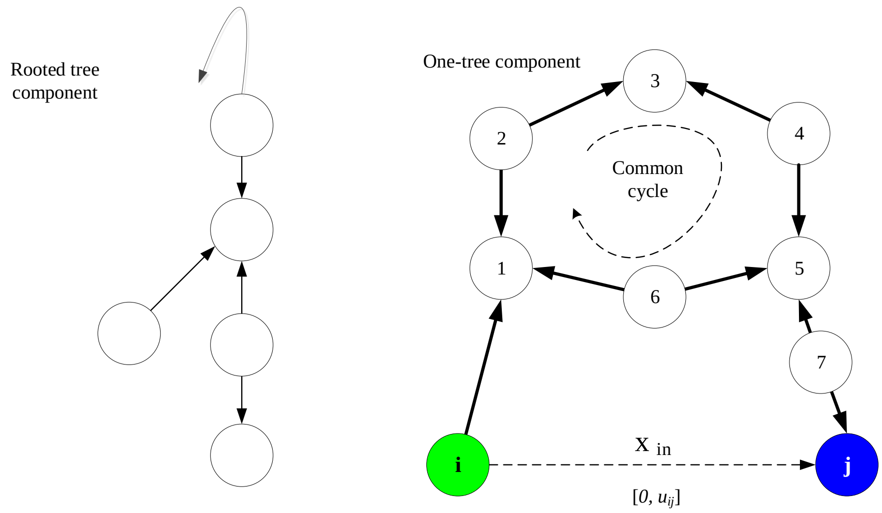



Figure 2.3. BEP of Incoming arc for one-tree component

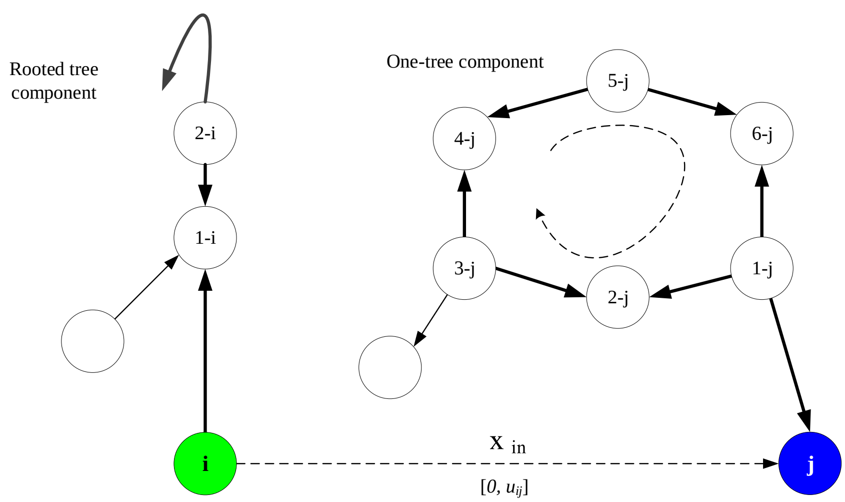
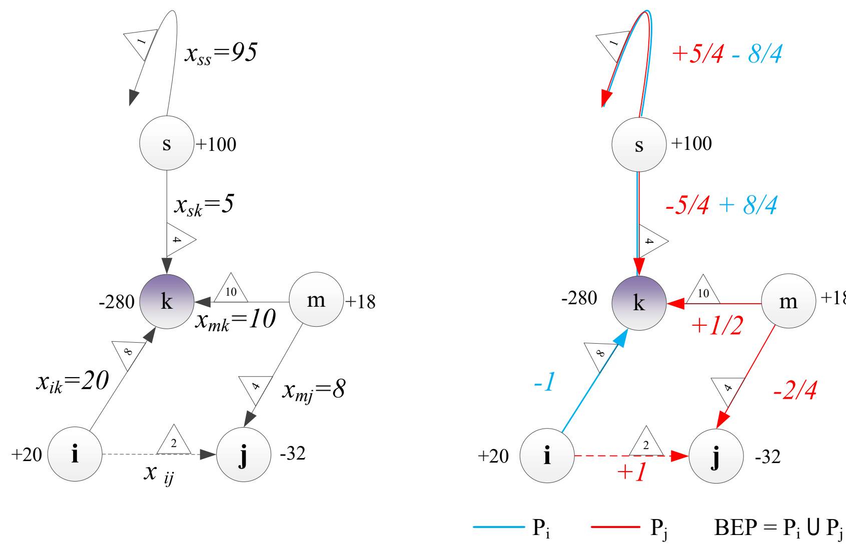



Figure 2.4. BEP of Incoming arc between two components




(a) (b)

Figure 2.5. (a) Basis with flows and nonbasic $(i, j)$, (b) Color BEP and representation of $(i, j)$

2. *Duals calculation.* Compute the marginal cost using the node dual values, $R_i$ and $K_j$, using: $\bar{c}_{ij} = c_{ij} - R_i - \mu_{ij} K_j$, based on GT's dual formulation, GT-D.

$$\textit{GT-D}: \quad \text{Maximize } \sum_{i=1}^{m} a_i R_i + \sum_{j=1}^{n} b_j K_j + \sum_{(i,j)\in A} U_{ij} w_{ij} \tag{2.12}$$

$$\text{subject to: } R_i + \mu_{ij} K_j \leq c_{ij}, \forall (i,j) \in A \tag{2.13}$$

$$R_i, K_j \text{ unrestricted in sign} \tag{2.14}$$

$$w_{ij} \leq 0, \forall (i,j) \in A \tag{2.15}$$

where $R_i$ the dual variable associated with source node $i \in O$, $K_j$ is the dual variable for sink node $j \in D$, and $w_{ij}$ is the dual associated with the upper bound of arc $(i,j) \in A$.

In either case, $\bar{c}_{ij} < 0$ indicates the potential of arc $(i,j)$ to improve the current basis' value. While the results of both methods are equivalent, approach 2 is clearly more efficient if node duals are available.

Since pricing can reveal many candidates for the incoming arc, there are a variety of techniques to choose one. Many selection rules have been offered, including candidate lists, first encountered, steepest descent, DEVEX, best overall, most improving, etc. This is discussed later in more detail.

Once the incoming arc has been selected, the flow change $\delta$ and the leaving basic arc must be computed prior to executing a pivot. Both are determined using the ratio test.

### 2.3.3. Step 2: Ratio Test

For an incoming nonbasic arc $(i,j)$, the ratio test uses the arc's representation to (a) determine $\delta$, the arc's level of flow in the new basis, and (b) the arc to leave the basis. These are found by applying Algorithm 2.1 and determining, for each basic arc $(r,s) \in BEP$, the maximum allowable flow decrease (depending on orientation), $\delta_{rs}$.

To ensure feasibility of the new basic arc flows, the flow change $\delta = \min_{(r,s)\in BEP}\{\delta_{rs} | \delta_{rs} \geq 0\}$, and the leaving arc's $\delta_{rs} = \delta$. (Ties should be handled as described in [35].) The actual solution value improvement from performing the pivot will be $\bar{c}\delta$.

Using our previous example from Figure 2.5 with the basis $\mathbf{x}_B$ and basis flows $\mathbf{b}$, the leaving variable has the minimum ratio from $\mathbf{x}_B \div \tilde{A}_{ij}$, using only the positive representation values.

$$\mathbf{x}_B = \begin{pmatrix} x_{sk} \\ x_{ik} \\ x_{mj} \\ x_{mk} \\ x_{ss} \end{pmatrix} : \mathbf{b} \div \tilde{A}_{ij} = \begin{pmatrix} 5 \\ 20 \\ 8 \\ 10 \\ 195 \end{pmatrix} \div \begin{pmatrix} -3/4 \\ 1 \\ 1/2 \\ -1/2 \\ 3/4 \end{pmatrix} = \begin{pmatrix} na \\ 20 \\ 16 \\ na \\ 380 \end{pmatrix}$$

Therefore arc $x_{mj}$ has the minimum ratio of 16 and will have zero flow when the flow increase on $(i, j)$ is $\delta = 16$.

After calculating $\delta$, we refer to any arc for which $\delta_{rs} = \delta$ as a *blocking arc*. The strongly feasible basis technique was proposed for pure networks by Barr, Glover, and Klingman in [11] and independently by Cunningham [28] and then generalized by Elam et al. in [35] for generalized networks. The strongly convergent algorithm specifies the selection procedure for the leaving arc $(r, s)$ in the case of ties to be: (1) the last blocking arc, in the set of basic arcs to be decreased to its lower bound, encountered in traversing BEP and if there is no such arc then (2) the first blocking arc in the set of the entering arc $(i, j)$ itself and basic arcs to be increased to its upper bound, encountered in traversing BEP. Our approach to FCGT applies similar technique to maintain the strongly feasible basis at each basis exchange step that reduces the number of degenerate pivots and converges to the optimal solution in a finite number of iterations.

### 2.3.4. Step 3: Pivot Execution

Updating the basis for FCGT depends on the relation of the entering and leaving arcs. It can modify the structure of the quasi-trees in a variety of ways because the endpoints of the entering arc can be in the same quasi-tree, in two separate quasi-trees, can be on a loop, or in a tributary tree, or the tree that arises upon suppression

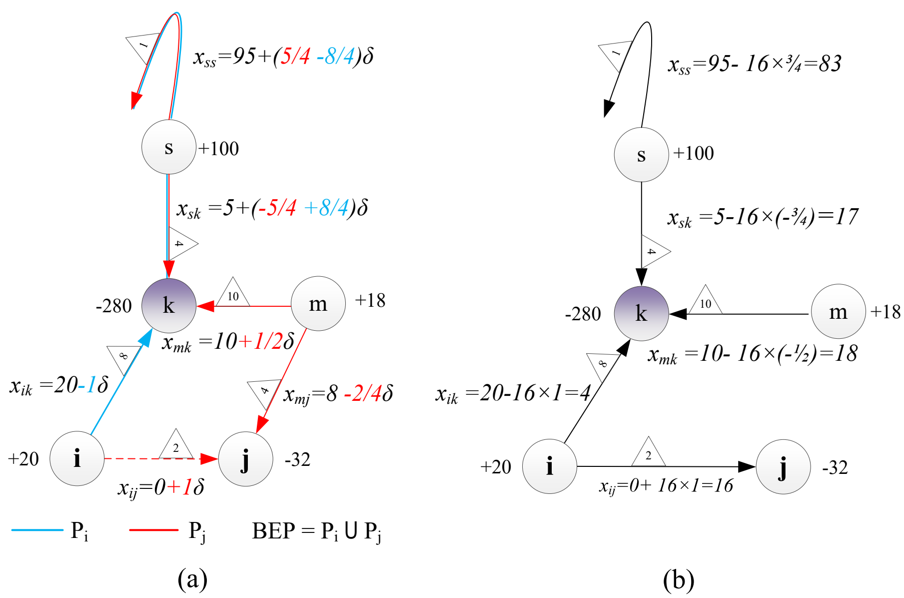



Figure 2.6. (a) Expressions for ratio test of incoming arc $(i, j)$, (b) New basis after adding setting $x_{ij} = \delta = 16$ and removing $x_{mj}$

of all quasi-tree loop arcs. The step of pivoting and updating the basis forest of quasi-trees can be performed by changing data structures, recomputing affected flows using computational simplifications similar to the pricing-out step for one-tree components. For rooted-tree components, updating flows and node duals is a straightforward procedure. For one-trees, when one value of the flow is determined, all other values can be calculated by a single traversal of the loop arcs in the subgraph that contains node $i$. Using the same procedure, flows can be calculated in the subgraph that contains node $j$ and then two resulting sets of flows can be added to get the desired representation whether or not subgraphs for $i$ and $j$ are the same. Those are details that might be added to the implementation with no bearing on the relation between pricing-out and change of basis step.

## 2.4. Extreme-Point Tabu Search Heuristic for FCGT

Despite this extensive work, the problem of applying fixed charges to generalized network flow problems has not been addressed in the literature. In practice, fixed charges emerge in a wide range of applications and provide added realism to linear formulations. Such is the case for generalized transportation problems as in Figure 2.7, which this research addresses. The meta-heuristic techniques of tabu search [43] can be employed in concert with the mathematical structure of GT (the linear relaxation of FCGT) to great advantage in solving the fixed-charge problem. These techniques include specialized move evaluation, short-term memory for tabu conditions with aspiration criteria, long-term memory for diversification, and candidate-list strategies.

### 2.4.1. Extreme-Point Tabu Search Overview

Tabu search is a meta-heuristic for exploring a solution space beyond local optimality. The *neighborhood* of any solution is defined as the set of other solutions that can be reached by a single *move*. The process uses a variety of rules and memory structures to perform a sequence of moves and visit a series of solutions with the goal of finding the optimal one.

The neighborhood of a given solution can be quite large, many available moves are not promising, and some moves can lead the search back to a previously visited point. Short-term memory can help avoid such cycling by temporarily assigning some moves a tabu status so that they will be avoided, unless overridden by meeting an aspiration criterion. Longer-term memory can be employed for diversification to force exploration of neighborhoods far away from the current solution.

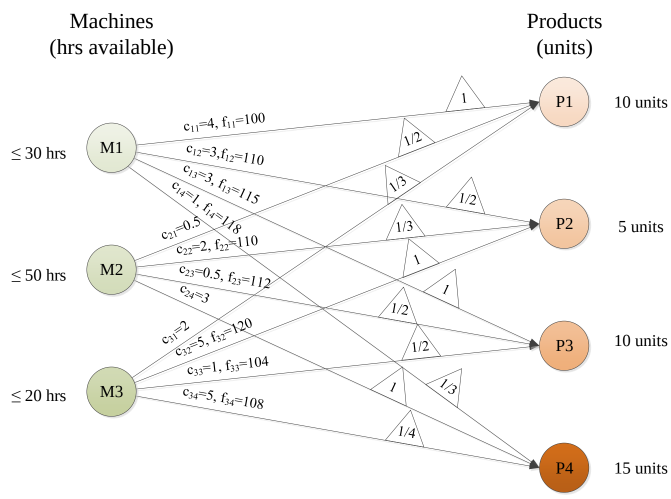



Figure 2.7. Addition of Fixed Charges on Generalized Arcs

For the EPTS heuristic for FCGT, a solution will be a basic feasible solution to GT, as in [104]. To transition from one solution to another will involve adding a new arc to the current solution and removing an existing arc from the basis, while maintaining feasibility. Such a move will be accomplished by executing a simplex pivot.

Potential moves are evaluated using operations on the current solution's forest of quasi-trees, short-term memory for moves, aspiration criteria, and candidate list structures. If no improving moves are available, a local optimum has been reached. Diversification is employed to move to new neighborhoods and, possibly, better local optima.

An outline of the approach, using objective functions from (2.1) and (2.7), is:

- *Phase 1, Initialize:* Tighten the arcs' upper bounds, solve the relaxation of FCGT as a linear program GT minimizing $LC(x)$, and save the final solution $x$ as $\mathbf{x}^\bullet$, the incumbent best found, with fixed-charge value $z^\bullet = FC(\mathbf{x}^\bullet)$.

- *Phase 2, Improve:* Starting at $x$, move through a series of adjacent extreme-point solutions, each of which improves $z = FC(x)$ until reaching $x'$, a local optimum with respect to FC(). If $FC(x') < z^{\bullet}$, update incumbent by setting $z^{\bullet} = FC(x')$ and $\mathbf{x}^{\bullet} = x'$

- *Phase 3, Diversify and Improve:* Apply the following $m_1$ times: starting at the most recent solution $x'$, make $m_2$ diversification moves to solution $x$; then reapply Phase 2 to move to a new local optimum and save any newly discovered incumbent.

- *Phase 4, Termination*: Exit the process, returning $\mathbf{x}^{\bullet}$ as the best solution found with value $z^{\bullet}$.

The EPTS algorithm is documented as Algorithm 2.2 and related procedures. The individual portions of the heuristic are detailed next.

**Algorithm 2.2** FCGT EPTS Algorithm

**Require:** $\mathcal{P}, m_1, m_2$ ▷

**Ensure:** $\mathbf{x}^{\bullet}, z^{\bullet}$

$z^{\bullet} \leftarrow \infty$, Iter $\leftarrow 0$ ▷ Initialize values

$u_{ij} \leftarrow \min\ \{u_{ij}, a_i, \frac{b_j}{\mu_{ij}}\}, \forall (i,j) \in A$ ▷ Tighten arc upper bounds

$\mathbf{x}' \leftarrow$ LPRSolve($\mathbf{u}$)

$\mathbf{x}'' \leftarrow$ LocSearch($\mathbf{x}'$), $\mathbf{z}'' \leftarrow$ ZF($\mathbf{x}''$)

IncumbentUpdate($\mathbf{x}^{\bullet}, z^{\bullet}, \mathbf{x}''$)

**for** $k = 1,\ m_1$ **do** ▷ Diversification inner loop

    $\mathbf{x}' \leftarrow$ Diversify($\mathbf{x}'', m_2$) ▷ Move $m_2$ diversification moves away

    $\mathbf{x}'' \leftarrow$ LocSearch($\mathbf{x}'$)

    IncumbentUpdate($\mathbf{x}^{\bullet}, z^{\bullet}, \mathbf{x}''$)

**end for**

Return $\mathbf{x}^{\bullet}, z^{\bullet}$

### 2.4.2. Phase 1: Initialization

The EPTS heuristic starts with finding an optimal solution, $x$, to GT, the linear relaxation of $FCGT$.

1. *Bound tightening.* Because of the structure of transportation problems, the flow on any arc $(i, j)$ cannot exceed the smaller of supply $a_i$ and demand $b_j/\mu_{ij}$. Since $f_{ij}$ is linearized as $f_{ij}/u_{ij}$, the smallest possible $u_{ij}$ will minimize the distance between FC() and LC() over its operable range. Hence the algorithm attempts to tighten the upper bounds used by adjusting $u_{ij} = \min\{u_{ij}, a_i, b_j/\mu_{ij}\}, \forall(i, j) \in A$.

2. *Initial solution.* GT is solved minimizing $LC(x)$ to produce optimal solution $x$, which is saved as the initial incumbent solution $\mathbf{x}^\bullet$ with value $z^\bullet = FC(\mathbf{x}^\bullet)$. Solution $x$ is the starting basis for Phase 2 and the duals are recomputed using the $c_{ij}$ variable costs only.

### 2.4.3. Phase 2: Improvement and Local FC Optimum

While Phase 1 of the EPTS heuristic involves the solution of the linear problem GT via the network simplex method, Phase 2 is concerned with finding solutions that minimize $FC(x)$, the true fixed-charge objective. As described previously, a potential incoming arc $(i, j)$ is evaluated based on its reduced cost, $\bar{c}_{ij} = c_{ij} - R_i - \mu_{ij}K_j$. Since the GT node duals only reflect variable costs and a portion of the fixed costs, the full effect of the flow changes is determined by using **both** pricing methods mentioned in Section 2.3.2, as described next.

#### *2.4.3.1. Move Evaluation*

The EPTS move evaluation process calculates the *total reduced cost*, $\kappa_{ij}$, which gives the effect on FCGT's $FC(x)$ objective if nonbasic arc $(i, j)$ is pivoted into the basis at its maximum allowed flow level, $\delta$. Steps are given in Algorithm 2.3.

---

**Algorithm 2.3** Total reduced cost calculation, $\kappa_{ij}$, for nonbasic $(i, j)$

1: Compute the representation and BEP for $(i, j)$ using Algorithm 2.1,

2: Apply the ratio test along the BEP to determine flow change $\delta$.

3: Retrace the BEP to determine the total reduced cost $\kappa_{ij}$ as:

$$\kappa_{ij} = \delta\bar{c}_{ij} + \sum_{(k,\ell)\in BEP(i,j)\cup(i,j)} \phi(k,\ell,\delta)$$

$$\text{where } \phi(k,\ell,\delta) = \begin{cases} F_{k\ell} & \text{if } x_{k\ell} = 0, \text{flow increases, and } \delta > 0 \\ -F_{k\ell} & \text{if } x_{k\ell} > 0, \text{flow decreases, and } \delta = x_{k\ell} \\ 0 & \text{otherwise} \end{cases}$$

and $\bar{c}_{ij} = c_{ij} - R_i - \mu_{ij} K_j$ using duals based on variable costs only.

---

Although the effort to compute $\kappa_{ij}$ is significantly higher than simple arc pricing with duals, it provides the exact value of the pivot's effect on the current objective $FC(x)$. For example, the ratio test might find that $\delta = 0$ and a pivot would result in a degenerate solution with no improvement in objective, despite an attractive $\bar{c}_{ij}$. Fortunately the second and third BEP traversals are expedited by knowing the previously computed BEP and representation values, so that $\phi(k, \ell, \delta)$ can be quickly determined.

*2.4.3.2. Candidate List*

Beginning with the earliest mathematical programming solution systems, a variety of heuristics have been used to select an incoming variable from all possible nonbasics. Insted of selecting the one with the best reduced cost, a more successful approach—

originally termed "multiple pricing" and "partial pricing," [99]—has been to gather a *candidate list* of attractive nonbasics to select from for a defined number of pivots and then replenishing the list with a new set of candidate variables [59, 93]. The motivation lies in the observation that is faster to search from a set of pre-selected candidates arcs than to evaluate all possibilities and choose the best at every iteration.

Our EPTS algorithm for FCGT employs a candidate list and other several nonbasic arc filters to select an incoming arc for a pivot. Since arc data is typically stored grouped by arcs leading into or out of a node, a convenient means of subdividing the search for attractive nonbasics is to evaluate those associated with a node and select the most attractive one, if any, to put on the list. Arc data is searched as a circular list of node-grouped arcs, adding the best eligible arc from a node group, until the candidate list is full or all arcs have been evaluated. If no eligible arcs can be found, a local optimum has been reached and this phase of the algorithm terminates.

Prior to each move or pivot, all list members are evaluated; the most attractive member (with largest $\kappa_{ij}$) is selected as the incoming arc and removed from the list, along with any other arcs that have become unattractive. After a given number of such selections are made, the list is discarded and a new one is created.

The algorithm's pivot strategy is controlled by two parameters, $(k_1, k_2)$, where $k_1$ is the maximum number of selections to be made from the current list and $k_2$ is the maximum length of the list. Hence a $(10, 20)$ strategy uses a candidate list of up to 20 arcs, from which up to 10 pivots can be made before rebuilding the list. While creating a new list, if $k_2$ attractive arcs cannot be found, the search is likely nearing the local optimum and a secondary pivot strategy, such as $(1, 1)$, can be deployed to reduce the pricing time when few eligible candidates are expected to be found.

#### 2.4.3.3. *Tabu Status and Aspiration Criterion*

To improve the search for a local optimum and avoid returning to a previously constructed extreme-point solution, the EPTS heuristic employs several techniques from the tabu search meta-heuristic pioneered by Glover [43, 57, 59]. Some of these mechanisms involve maintaining historical information about the status of each variable and adjusting move decisions based on this memory-based data.

The *tabu status* technique uses recency-based "short-term memory" to record when an arc last left an active status and ensure that it is not re-activated for a given number of moves. Hence, when a solution variable is the leaving arc in a pivot, it is assigned a temporary "tabu" status and blocked from re-entering the solution for a pre-specified number of iterations. At that time, the arc's tabu status is set as $TabuEnd = Iter + TabuTenure$, where $Iter$ is the current iteration number and $TabuTenure$ is the user-defined number of iterations the arc should not be considered for entry into a solution again.

This static tabu search method, as described in [108], can be readily incorporated into the candidate list move selection mechanism. Before applying Algorithm 2.3 to evaluate a nonbasic arc to add to the current list, the arc's *TabuEnd* status can be checked for eligibility. If currently $Iter < TabuEnd$, the arc should not be considered. The static tabu search technique is remarkably powerful and has been shown to significantly improve the quality of solutions discovered for integer programming and combinatorial problems.

However, strict application of the tabu classification rule has been found to be enhanced by the addition of *aspiration criteria*, which define situations when the rule can be overridden for moves. Criteria can include aspiration by: default, when all possible moves are tabu; objective, when a tabu move would reach a new incumbent best solution; search direction, when a search has stalled at an objective; and influence, if a tabu move would significantly change the structure of a solution [58].

Our EPTS algorithm uses the default and objective aspiration criteria, based on $\kappa_{ij}$ move evaluations. When building a candidate list or selecting an incoming arc from an existing list at a solution point $x$, an arc's tabu status can be overridden if $\kappa_{ij} + FC(x) < z^{\bullet}$, the value of the best solution found so far. So if the proposed move would produce a new incumbent solution, then the tabu status is overridden by the aspiration criterion and the nonbasic arc is brought to the quasi-tree forest basis.

#### *2.4.3.4. Stopping Criteria and Incumbent Update*

If at some point the candidate list cannot be fully replenished, then pricing shifts to the secondary strategy. If no improving nonbasic arcs exit, Phase 2 terminates with the current solution, $x$, as a local optimum. If $FC(x) < z^{\bullet}$, a new incumbent best solution has been found and $z^{\bullet} \leftarrow FC(x)$ and $\mathbf{x}^{\bullet} \leftarrow x$.

### 2.4.4. Phase 3: Diversification of Local Search

Since the objective function is concave, even when the heuristic's optimality criteria are met and no improving arcs can be found, it is not true that a global minimum has been reached [109]. It is still possible that there exists an improved solution outside of the neighborhood.

Numerous approaches have been proposed for moving the search out of local minimum to different regions of the solution landscape. One of such mechanisms is the long-term memory component of the tabu search that guides other search procedures to move from one solution to another to overcome local optimality.

Our EPTS heuristic adapts a variation of the approach described in [39, 40, 41, 104, 105]. The diversification component of FCGT is the long-term memory search process that brings into the basis non-basic variables that were non-basic for the longest time.

The diversification phase is called when a particular number of non-improving iterations or pivots were encountered. The goal of the diversify approach is to move away from the current local minimum to hopefully improve the solution by exploring a new region that was not reached before by local search.

A parameter for the maximum number of times the diversification function is performed is defined as $m_1$. If that parameter has been reached then the search heuristic stops and the best solution is reported. The diversification step consists of bringing into the basis several non-basic arcs that have been non-basic the longest time. The parameter $m_2$ defines the number of non-basic arcs or variables forced to enter the basis and determined beforehand. The overall objective function value more than likely will be worse after bringing several non-basic arcs into the basis forest of quasi-trees, but the goal is to induce the search to a new subregion of the solution space [105]. After pivoting the number of non-basics into basis forest of quasi-trees, the search returns to the local search trying to find another local optimum which hopefully is better than the previous local optimum found and therefore finding the global optimum.

#### 2.4.5. Phase 4: Termination

Parameter $m_1$ described in phase 3 provides a stopping criteria that makes the search finite. Meeting the optimality criteria, reaching the maximum number of iterations or moves, and reaching the maximum number of diversification steps determine the termination of the search process.

### 2.5. Computational Testing

This section describes the experimental design used to test the effectiveness of the proposed heuristic approach against a commercially available state-of-the-art solver. The experiment is designed to test the quality of solution and the solution time. The software used, the problems set, results, and statistical analysis are described in the following paragraphs.

### 2.5.1. Solvers Tested

#### *2.5.1.1. Commercial Software Description*

The commercial software used for comparison purposes is CPLEX version 12.6.0.0 from IBM at Southern Methodist University's Lyle School of Engineering with default settings, single thread mode, and a time limit of 3600 seconds. Only the time that CPLEX solver used to solve the problem is used for comparison. The time limit is altered to ensure a timely termination of testing.

#### *2.5.1.2. FIXNET Software Description*

The base for the implementation of the EPTS heuristic for the fixed-charge generalized transportation problem is a one-multiplier generalized network solver FIXNET, developed by the author and written in FORTRAN, which can solve uncapacitated and capacitated generalized and pure networks, including transportation and transshipment structures. The current implementation extends the capabilities of GN to solve the class of fixed-charge generalized transportation problems.

The data structure for nodes holds the node potential, requirements (supply/demand), and quasi-tree structure for each node. The quasi-tree data is maintained using the concept of predecessors and threads as defined by Barr et al. [11], which additionally stores the information about loop factors for the nodes that belong to one-trees. The data structure for arcs holds all the information for each arc including from and to nodes, upper bounds, multipliers, conditional lower bounds, a flag to determine if the arc is part of basis set $\mathcal{F}$, or non-basic sets $\mathcal{L}$ or $\mathcal{U}$, the reduced cost and total reduced cost as part of the entering arc selection process.

The FIXNET code captures multiple statistics as it solves each test problem including but not limited to the relaxed solution, total cost for the relaxed solution, local search solution, variable and fixed cost at the local search solution, number of degenerate pivots, number of arcs at upper bound for the local search solution with total flow at upper bound, number of times aspiration criteria was applied, number of rooted trees and number of nodes on cycles for the relaxed solution and for the local search solution to analyze the basis forest structure for the solution. Timing statistics are captured for several sections of the code and include time for relaxed solution and overall solution time which also includes reading from the data file and reinverting the network to eliminate round-off errors and recalculate node duals for different costs.

### 2.5.2. Test Environment

General use Linux machine Dell R730 with Intel Xeon@2.6 GHz, 320GB RAM was used as the computing environment to run all test problems with the FIXNET code written in FORTRAN and compiled using gfortran and CPLEX for comparison.

### 2.5.3. Test Problems

#### *2.5.3.1. Problem Generator*

The available network generator GNETGEN (NETGEN modification by F. Glover) is used as a basic tool to generate all random generalized networks flow problems. The generator was modified to include the parameters needed for FCGT problem such as fixed charges on arcs. The output file generated by GNETGEN with fixed charges is read by *createInput* program that outputs a network file in the FIXNET acceptable formats. Another routine written in AWK, creates a data file for AMPL front end for CPLEX. The test problem definitions for each type of nodes-arc levels and fixed-charge types (FCtypes) are shown in Table 2.1.

#### *2.5.3.2. Test Set Generated*

Due to the absence of published research for fixed-charge generalized networks, we did not find any standard problem set for testing performance of proposed heuristic for FCGT. For consistency, we use available testbed parameters as in Sun et al. [104] and Glover et al. [44] for FC transportation problems for pure networks with 100% fixed charge capacitated and 100% dense transportation network problems. The variable costs range over values from 3 to 8 and multipliers range over values from 0.5 to 1.5. Total supply for 30x100 size problems is 30,000 and for 50x100 size problems is 50,000.

The test set main factors and levels that later are used for the ANOVA analysis, shown in Table 2.2. Differences in types of fixed-charges on arcs for test problems are in their ranges of fixed costs (FC) and described in the Table 2.3.

### 2.5.4. Test Results

User specified parameters such as candidate list strategy $(15, 20)$ for its length, 20, and number of pivots between replenishment, 15, and Tabu parameter 25 are used for all test problems solved by FIXNET solver. Parameter calibration were run on $3,072$ problems to test four different candidate list strategies and six different tabu length parameters. Candidate list strategy $(15, 20)$ and tabu parameter of 25 were shown to be robust and therefore are used in all test runs. The following subsections describe

Table 2.1. Test Set Definitions

| Problem Set | UB minimum | Multiplier type |
|---|---|---|
| fcgnTest1-nodes-arcs-FCtype | 1000 | mixed or [0.5-1.5] |
| fcgnTest2-nodes-arcs-FCtype | 800 | mixed or [0.5-1.5] |
| fcgnTest3-nodes-arcs-FCtype | 1000 | lossy or [0.5-1.0] |
| fcgnTest4-nodes-arcs-FCtype | 800 | lossy or [0.5-1.0] |
| fcgnTest5-nodes-arcs-FCtype | 1000 | gainy or [1.0-1.5] |
| fcgnTest6-nodes-arcs-FCtype | 800 | gainy or [1.0-1.5] |
| fcgnTest7-nodes-arcs-FCtype | 800 | pure or [1.0-1.0] |
| fcgnTest8-nodes-arcs-FCtype | 1000 | pure or [1.0-1.0] |

Table 2.2. Test Set: Problem Factors and Levels

| Factor | Levels | |
|---|---|---|
| Number of nodes and arcs for TP network | (130,3000) | (150,5000) |
| Minimum value of upper bound on an arc | 800 | 1,000 |
| Minimum value of multiplier on an arc | 0.5 | 1.0 |
| Maximum value of multiplier on an arc | 1.0 | 1.5 |
| Fixed-charges on an arc | 8 levels: A–H | |

Table 2.3. Parameters for Fixed-charge Types

| Problem | FC Type | FC Range |
|---|---|---|
| 1 | A | [50, 200] |
| 2 | B | [100, 400] |
| 3 | C | [200, 800] |
| 4 | D | [400,1600] |
| 5 | E | [800, 3200] |
| 6 | F | [1600, 6400] |
| 7 | G | [3200, 12800] |
| 8 | H | [6400, 25600] |

the statistical results for the data of 128 problems run by FIXNET and CPLEX.

Tables 2.4 - 2.7 list the codes' performances for test problems. To compare the qualities of the integer solution values obtained by FIXNET and CPLEX, we define the metric, $R$, as: $R = z_1/z_2$, where $z_1$ and $z_2$ are the best integer upper-bound solution values obtained by FIXNET and CPLEX, respectively, per Sun et al. [104]. For example, if $R = 1.01$, then FIXNET's solution value was 1% larger than the best from CPLEX.

Also shown is the *time multiple*, the ratio of the CPLEX solution time to that of the FIXNET. A time multiple of 2, then, indicates that CPLEX required twice the solution time as FIXNET. The table shows the solution value and solution time for each code, $R$ metric, and time multiple.

There are 128 problems divided into four tables for 130 nodes and 3000 arcs, 150 nodes and 5000 arcs, and for a minimum value of upper bound limit of 800 and 1000. Each table shows 32 test problems results and provided overall average, standard deviation, minimum, median, and maximum values for each column.

### 2.5.5. Statistical Analysis

Two main hypothesis were tested: one for solution quality and another one for solution time. The first hypothesis states there is no statistical difference in solution quality between the FIXNET code and CPLEX results, where $\bar{z_1}$ is the average objective value for 128 test problems solved by CPLEX and $\bar{z_2}$ is the average objective value for the same 128 test problems solved by FIXNET code.

$$H_0 : \bar{z_1} = \bar{z_2} \tag{2.16}$$

$$H_1 : \bar{z_1} \neq \bar{z_2} \tag{2.17}$$

Table 2.4. Empirical results for 130 nodes and 3000 arcs network with 800 of minimum value for UB

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | R | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest2-130-3000-A | 95,749 | 96,571 | 1.009 | 12.54 | 0.02 | 535.2 |
| fcgnTest2-130-3000-B | 101,644 | 103,133 | 1.015 | 2.39 | 0.02 | 102.0 |
| fcgnTest2-130-3000-C | 123,007 | 126,032 | 1.025 | 62.80 | 0.02 | 4020.3 |
| fcgnTest2-130-3000-D | 152,589 | 156,552 | 1.026 | 16.69 | 0.02 | 1068.7 |
| fcgnTest2-130-3000-E | 211,423 | 220,774 | 1.044 | 50.78 | 0.02 | 3251.0 |
| fcgnTest2-130-3000-F | 309,944 | 330,598 | 1.067 | 4.18 | 0.02 | 178.5 |
| fcgnTest2-130-3000-G | 515,732 | 562,115 | 1.090 | 21.80 | 0.04 | 558.2 |
| fcgnTest2-130-3000-H | 927,249 | 1,008,104 | 1.087 | 238.57 | 0.02 | 10177.8 |
| fcgnTest4-130-3000-A | 186,597 | 186,753 | 1.001 | 54.12 | 0.02 | 3464.9 |
| fcgnTest4-130-3000-B | 211,636 | 211,781 | 1.001 | 3600.00 | 0.02 | 153583.6 |
| fcgnTest4-130-3000-C | 230,272 | 231,465 | 1.005 | 3600.00 | 0.02 | 230473.8 |
| fcgnTest4-130-3000-D | 280,474 | 282,164 | 1.006 | 1322.65 | 0.02 | 56427.0 |
| fcgnTest4-130-3000-E | 406,078 | 408,982 | 1.007 | 3600.00 | 0.02 | 230473.8 |
| fcgnTest4-130-3000-F | 671,863 | 678,027 | 1.009 | 3600.00 | 0.02 | 230473.8 |
| fcgnTest4-130-3000-G | 1,095,638 | 1,107,427 | 1.011 | 3600.00 | 0.04 | 92165.9 |
| fcgnTest4-130-3000-H | 2,033,811 | 2,049,132 | 1.008 | 3600.00 | 0.01 | 460947.5 |
| fcgnTest6-130-3000-A | 84,721 | 85,036 | 1.004 | 21.30 | 0.01 | 2727.6 |
| fcgnTest6-130-3000-B | 95,727 | 96,607 | 1.009 | 6.27 | 0.02 | 401.6 |
| fcgnTest6-130-3000-C | 112,186 | 113,110 | 1.008 | 12.13 | 0.02 | 776.2 |
| fcgnTest6-130-3000-D | 141,452 | 145,325 | 1.027 | 32.91 | 0.02 | 2107.0 |
| fcgnTest6-130-3000-E | 190,802 | 195,845 | 1.026 | 4.94 | 0.02 | 316.0 |
| fcgnTest6-130-3000-F | 294,321 | 307,546 | 1.045 | 2.12 | 0.02 | 136.0 |
| fcgnTest6-130-3000-G | 492,861 | 512,241 | 1.039 | 0.91 | 0.01 | 117.1 |
| fcgnTest6-130-3000-H | 885,265 | 953,115 | 1.077 | 1.43 | 0.02 | 91.7 |
| fcgnTest7-130-3000-A | 111,283 | 111,817 | 1.005 | 3600.00 | 0.02 | 230473.8 |
| fcgnTest7-130-3000-B | 125,433 | 126,267 | 1.007 | 3600.00 | 0.04 | 92165.9 |
| fcgnTest7-130-3000-C | 144,935 | 145,717 | 1.005 | 3600.00 | 0.06 | 57600.0 |
| fcgnTest7-130-3000-D | 179,764 | 182,204 | 1.014 | 3600.00 | 0.09 | 41889.7 |
| fcgnTest7-130-3000-E | 238,706 | 240,583 | 1.008 | 3600.00 | 0.05 | 65825.6 |
| fcgnTest7-130-3000-F | 370,447 | 373,127 | 1.007 | 3600.00 | 0.04 | 92165.9 |
| fcgnTest7-130-3000-G | 624,195 | 625,540 | 1.002 | 3600.00 | 0.04 | 92165.9 |
| fcgnTest7-130-3000-H | 1,096,563 | 1,108,508 | 1.011 | 3600.00 | 0.03 | 115200.0 |
| Overall average | 398,199 | 408,819 | 1.022 | 1633.39 | 0.03 | 71001.94 |
| Overall st.dev. | 420,489 | 429,626 | 0.025 | 1776.94 | 0.02 | 105831.27 |
| Minimum | 84,721 | 85,036 | 1.001 | 0.91 | 0.01 | 91.68 |
| Median | 220,954 | 226,119 | 1.009 | 150.68 | 0.02 | 7099.06 |
| Maximum | 2,033,811 | 2,049,132 | 1.090 | 3600.00 | 0.09 | 460947.50 |

Table 2.5. Empirical results for 130 nodes and 3000 arcs network with 1000 of minimum value for UB

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | R | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest1-130-3000-A | 92,417 | 92,877 | 1.005 | 37.50 | 0.02 | 2401.0 |
| fcgnTest1-130-3000-B | 102,556 | 103,715 | 1.011 | 25.21 | 0.04 | 645.4 |
| fcgnTest1-130-3000-C | 115,185 | 117,253 | 1.018 | 8.53 | 0.02 | 546.1 |
| fcgnTest1-130-3000-D | 146,472 | 148,290 | 1.012 | 2.24 | 0.01 | 286.8 |
| fcgnTest1-130-3000-E | 208,957 | 216,289 | 1.035 | 8.47 | 0.02 | 542.4 |
| fcgnTest1-130-3000-F | 329,878 | 350,065 | 1.061 | 44.87 | 0.04 | 1148.8 |
| fcgnTest1-130-3000-G | 525,997 | 561,945 | 1.068 | 478.91 | 0.02 | 20431.2 |
| fcgnTest1-130-3000-H | 909,331 | 999,018 | 1.099 | 80.55 | 0.02 | 5157.0 |
| fcgnTest3-130-3000-A | 188,884 | 188,995 | 1.001 | 253.84 | 0.03 | 8123.0 |
| fcgnTest3-130-3000-B | 205,646 | 205,825 | 1.001 | 297.65 | 0.02 | 19055.6 |
| fcgnTest3-130-3000-C | 243,962 | 244,551 | 1.002 | 9.01 | 0.02 | 576.8 |
| fcgnTest3-130-3000-D | 297,458 | 298,834 | 1.005 | 1781.74 | 0.01 | 228135.7 |
| fcgnTest3-130-3000-E | 407,548 | 411,877 | 1.011 | 457.70 | 0.02 | 29301.9 |
| fcgnTest3-130-3000-F | 613,303 | 622,369 | 1.015 | 3600.00 | 0.04 | 92165.9 |
| fcgnTest3-130-3000-G | 1,069,340 | 1,085,471 | 1.015 | 3600.00 | 0.04 | 92165.9 |
| fcgnTest3-130-3000-H | 1,925,228 | 1,977,657 | 1.027 | 3600.00 | 0.02 | 153583.6 |
| fcgnTest5-130-3000-A | 83,704 | 84,204 | 1.006 | 28.76 | 0.01 | 3682.4 |
| fcgnTest5-130-3000-B | 94,220 | 94,624 | 1.004 | 8.55 | 0.01 | 1095.1 |
| fcgnTest5-130-3000-C | 110,144 | 111,383 | 1.011 | 0.74 | 0.01 | 94.3 |
| fcgnTest5-130-3000-D | 138,347 | 142,631 | 1.031 | 46.28 | 0.02 | 2962.9 |
| fcgnTest5-130-3000-E | 192,723 | 200,963 | 1.043 | 4.19 | 0.02 | 268.6 |
| fcgnTest5-130-3000-F | 292,146 | 310,160 | 1.062 | 55.76 | 0.02 | 3569.5 |
| fcgnTest5-130-3000-G | 498,590 | 519,061 | 1.041 | 27.01 | 0.01 | 3458.1 |
| fcgnTest5-130-3000-H | 870,628 | 927,512 | 1.065 | 56.42 | 0.02 | 2406.8 |
| fcgnTest8-130-3000-A | 112,833 | 113,199 | 1.003 | 3600.00 | 0.04 | 92165.9 |
| fcgnTest8-130-3000-B | 122,542 | 123,898 | 1.011 | 3600.00 | 0.02 | 153583.6 |
| fcgnTest8-130-3000-C | 143,237 | 144,141 | 1.006 | 3600.00 | 0.05 | 76791.8 |
| fcgnTest8-130-3000-D | 179,973 | 179,984 | 1.000 | 3600.00 | 0.05 | 76791.8 |
| fcgnTest8-130-3000-E | 246,057 | 247,741 | 1.007 | 3600.00 | 0.04 | 92165.9 |
| fcgnTest8-130-3000-F | 378,477 | 380,110 | 1.004 | 3600.00 | 0.04 | 92165.9 |
| fcgnTest8-130-3000-G | 631,238 | 638,869 | 1.012 | 3600.00 | 0.05 | 65825.6 |
| fcgnTest8-130-3000-H | 1,116,227 | 1,122,982 | 1.006 | 3600.00 | 0.02 | 153583.6 |
| Overall average | 393,539 | 405,203 | 1.022 | 1353.56 | 0.02 | 46089.97 |
| Overall st.dev. | 404,697 | 418,495 | 0.025 | 1682.16 | 0.01 | 60967.11 |
| Minimum | 83,704 | 84,204 | 1.000 | 0.74 | 0.01 | 94.34 |
| Median | 226,460 | 230,420 | 1.011 | 167.20 | 0.02 | 6640.00 |
| Maximum | 1,925,228 | 1,977,657 | 1.099 | 3600.00 | 0.05 | 228135.72 |

Table 2.6. Empirical results for 150 nodes and 5000 arcs network with 800 of minimum value for UB

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | R | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest2-150-5000-A | 145,634 | 146,140 | 1.003 | 275.00 | 0.02 | 11732.08 |
| fcgnTest2-150-5000-B | 152,547 | 154,688 | 1.014 | 78.31 | 0.03 | 2505.99 |
| fcgnTest2-150-5000-C | 171,869 | 173,661 | 1.010 | 64.73 | 0.04 | 1657.12 |
| fcgnTest2-150-5000-D | 200,605 | 205,896 | 1.026 | 476.10 | 0.03 | 15235.17 |
| fcgnTest2-150-5000-E | 271,000 | 290,289 | 1.071 | 593.82 | 0.02 | 25333.70 |
| fcgnTest2-150-5000-F | 375,329 | 396,433 | 1.056 | 80.06 | 0.02 | 3415.38 |
| fcgnTest2-150-5000-G | 577,676 | 621,744 | 1.076 | 60.42 | 0.02 | 3867.90 |
| fcgnTest2-150-5000-H | 984,349 | 1,056,675 | 1.073 | 103.02 | 0.03 | 3296.54 |
| fcgnTest4-150-5000-A | 300,199 | 300,457 | 1.001 | 287.78 | 0.05 | 6138.67 |
| fcgnTest4-150-5000-B | 326,901 | 327,121 | 1.001 | 71.67 | 0.03 | 2293.43 |
| fcgnTest4-150-5000-C | 362,259 | 363,080 | 1.002 | 3600.00 | 0.02 | 153583.62 |
| fcgnTest4-150-5000-D | 395,566 | 398,868 | 1.008 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest4-150-5000-E | 533,002 | 539,940 | 1.013 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest4-150-5000-F | 809,654 | 816,569 | 1.009 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest4-150-5000-G | 1,268,954 | 1,295,758 | 1.021 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest4-150-5000-H | 2,318,198 | 2,351,353 | 1.014 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest6-150-5000-A | 133,556 | 134,384 | 1.006 | 33.48 | 0.02 | 2143.28 |
| fcgnTest6-150-5000-B | 144,183 | 145,750 | 1.011 | 1174.57 | 0.04 | 30070.92 |
| fcgnTest6-150-5000-C | 159,408 | 163,524 | 1.026 | 1276.01 | 0.02 | 54437.29 |
| fcgnTest6-150-5000-D | 191,292 | 196,155 | 1.025 | 6.72 | 0.02 | 430.29 |
| fcgnTest6-150-5000-E | 246,603 | 257,087 | 1.043 | 151.31 | 0.02 | 9687.00 |
| fcgnTest6-150-5000-F | 355,263 | 366,340 | 1.031 | 14.47 | 0.02 | 617.21 |
| fcgnTest6-150-5000-G | 551,199 | 593,668 | 1.077 | 124.65 | 0.02 | 7979.90 |
| fcgnTest6-150-5000-H | 956,078 | 992,928 | 1.039 | 92.58 | 0.02 | 5926.94 |
| fcgnTest7-150-5000-A | 174,885 | 175,606 | 1.004 | 3600.00 | 0.06 | 57600.00 |
| fcgnTest7-150-5000-B | 189,422 | 190,846 | 1.008 | 3600.00 | 0.03 | 115200.00 |
| fcgnTest7-150-5000-C | 219,006 | 220,431 | 1.007 | 3600.00 | 0.08 | 46082.95 |
| fcgnTest7-150-5000-D | 257,661 | 259,674 | 1.008 | 3600.00 | 0.09 | 41889.69 |
| fcgnTest7-150-5000-E | 340,455 | 351,475 | 1.032 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest7-150-5000-F | 492,446 | 499,206 | 1.014 | 3600.00 | 0.07 | 51201.82 |
| fcgnTest7-150-5000-G | 781,501 | 788,399 | 1.009 | 3600.00 | 0.07 | 51201.82 |
| fcgnTest7-150-5000-H | 1,310,903 | 1,324,079 | 1.010 | 3600.00 | 0.03 | 115200.00 |
| Overall average | 490,550 | 503,069 | 1.023 | 1730.15 | 0.03 | 60154.86 |
| Overall st.dev. | 465,361 | 474,926 | 0.024 | 1698.97 | 0.02 | 76496.29 |
| Minimum | 133,556 | 134,384 | 1.001 | 6.72 | 0.02 | 430.29 |
| Median | 333,678 | 339,298 | 1.013 | 884.20 | 0.03 | 27702.31 |
| Maximum | 2,318,198 | 2,351,353 | 1.077 | 3600.00 | 0.09 | 230473.75 |

Table 2.7. Empirical results for 150 nodes and 5000 arcs network with 1000 of minimum value for UB

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | R | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest1-150-5000-A | 141,444 | 141,911 | 1.003 | 64.98 | 0.02 | 2772.30 |
| fcgnTest1-150-5000-B | 153,395 | 154,173 | 1.005 | 39.44 | 0.02 | 2525.15 |
| fcgnTest1-150-5000-C | 165,237 | 167,270 | 1.012 | 56.56 | 0.02 | 2413.05 |
| fcgnTest1-150-5000-D | 202,641 | 207,865 | 1.026 | 10.65 | 0.02 | 454.19 |
| fcgnTest1-150-5000-E | 269,725 | 277,052 | 1.027 | 770.73 | 0.05 | 16440.51 |
| fcgnTest1-150-5000-F | 377,150 | 393,364 | 1.043 | 43.35 | 0.03 | 1387.18 |
| fcgnTest1-150-5000-G | 591,978 | 627,149 | 1.059 | 1795.29 | 0.04 | 45962.37 |
| fcgnTest1-150-5000-H | 987,862 | 1,084,330 | 1.098 | 2685.09 | 0.04 | 68742.70 |
| fcgnTest3-150-5000-A | 300,692 | 300,904 | 1.001 | 201.66 | 0.05 | 3687.35 |
| fcgnTest3-150-5000-B | 307,406 | 307,609 | 1.001 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest3-150-5000-C | 335,554 | 333,668 | 0.994 | 3600.00 | 0.03 | 115200.00 |
| fcgnTest3-150-5000-D | 404,886 | 406,103 | 1.003 | 3600.00 | 0.08 | 46082.95 |
| fcgnTest3-150-5000-E | 542,330 | 551,278 | 1.016 | 3600.00 | 0.05 | 76791.81 |
| fcgnTest3-150-5000-F | 793,989 | 808,525 | 1.018 | 23.48 | 0.03 | 751.34 |
| fcgnTest3-150-5000-G | 1,305,463 | 1,314,328 | 1.007 | 3600.00 | 0.03 | 115200.00 |
| fcgnTest3-150-5000-H | 2,465,266 | 2,487,311 | 1.009 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest5-150-5000-A | 133,532 | 133,972 | 1.003 | 357.31 | 0.02 | 15243.47 |
| fcgnTest5-150-5000-B | 150,938 | 152,665 | 1.011 | 4.09 | 0.03 | 131.04 |
| fcgnTest5-150-5000-C | 157,178 | 159,417 | 1.014 | 41.34 | 0.03 | 1322.91 |
| fcgnTest5-150-5000-D | 189,913 | 196,037 | 1.032 | 619.74 | 0.02 | 39676.06 |
| fcgnTest5-150-5000-E | 253,274 | 260,501 | 1.029 | 79.16 | 0.02 | 5067.60 |
| fcgnTest5-150-5000-F | 353,658 | 378,625 | 1.071 | 6.18 | 0.02 | 263.81 |
| fcgnTest5-150-5000-G | 569,891 | 603,294 | 1.059 | 136.60 | 0.02 | 5827.82 |
| fcgnTest5-150-5000-H | 925,685 | 978,339 | 1.057 | 11.02 | 0.01 | 1410.99 |
| fcgnTest8-150-5000-A | 178,830 | 179,353 | 1.003 | 3600.00 | 0.06 | 57600.00 |
| fcgnTest8-150-5000-B | 188,771 | 189,818 | 1.006 | 3600.00 | 0.05 | 65825.56 |
| fcgnTest8-150-5000-C | 216,078 | 218,218 | 1.010 | 3600.00 | 0.06 | 57600.00 |
| fcgnTest8-150-5000-D | 259,043 | 261,126 | 1.008 | 3600.00 | 0.06 | 57600.00 |
| fcgnTest8-150-5000-E | 340,987 | 351,006 | 1.029 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest8-150-5000-F | 482,548 | 488,722 | 1.013 | 3600.00 | 0.03 | 115200.00 |
| fcgnTest8-150-5000-G | 769,946 | 779,461 | 1.012 | 3600.00 | 0.05 | 65825.56 |
| fcgnTest8-150-5000-H | 1,342,828 | 1,347,275 | 1.003 | 3600.00 | 0.06 | 57600.00 |
| Overall average | 495,566 | 507,521 | 1.021 | 1792.08 | 0.04 | 49928.72 |
| Overall st.dev. | 487,291 | 495,548 | 0.024 | 1706.60 | 0.02 | 60573.64 |
| Minimum | 133,532 | 133,972 | 0.994 | 4.09 | 0.01 | 131.04 |
| Median | 321,480 | 320,638 | 1.012 | 1,283.01 | 0.03 | 42819.21 |
| Maximum | 2,465,266 | 2,487,311 | 1.098 | 3600.00 | 0.08 | 230473.75 |

The second hypothesis states there is a significant difference in solution time with the FIXNET code obtaining results faster than CPLEX with $\bar{T_1}$ and $\bar{T_2}$ being average solution time respectively for CPLEX and FIXNET solvers.

$$H_0 : \bar{T_1} = \bar{T_2} \tag{2.18}$$

$$H_1 : \bar{T_1} \neq \bar{T_2} \tag{2.19}$$

The level of significance of $\alpha = 5\%$ was used for all statistical tests. The following paragraphs describe the statistical results for the test data of 128 problems run separately by FIXNET and CPLEX.

#### *2.5.5.1. Analysis of Quality of Solution due to Code Type*

To support the first hypothesis, an analysis of variance was performed to determine factors affecting the solution quality. The factors considered were the code type, the total number of nodes and arcs, minimum value of upper bound on an arc, multiplier type (mixed, lossy, gainy, and pure networks), and fixed-charges type. It is expected that the solution quality may vary due to different problem sizes, multiplier types, and fixed-cost ranges. One important factor for this analysis is to determine if the solution quality due to the code type or any interactions with code type. Also it is important to analyze how close FIXNET objective values compared to CPLEX's. A boxplot of objective values found by CPLEX and FIXNET codes is shown in Figure 2.8. This figure includes all 256 test problems and shows that the solutions found by FIXNET are comparable to those found by CPLEX.

All 256 observations were used for analysis using the ANOVA procedure with Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] Jupyter notebook, server version 5.7.4. with significance level of 5%. The results show that

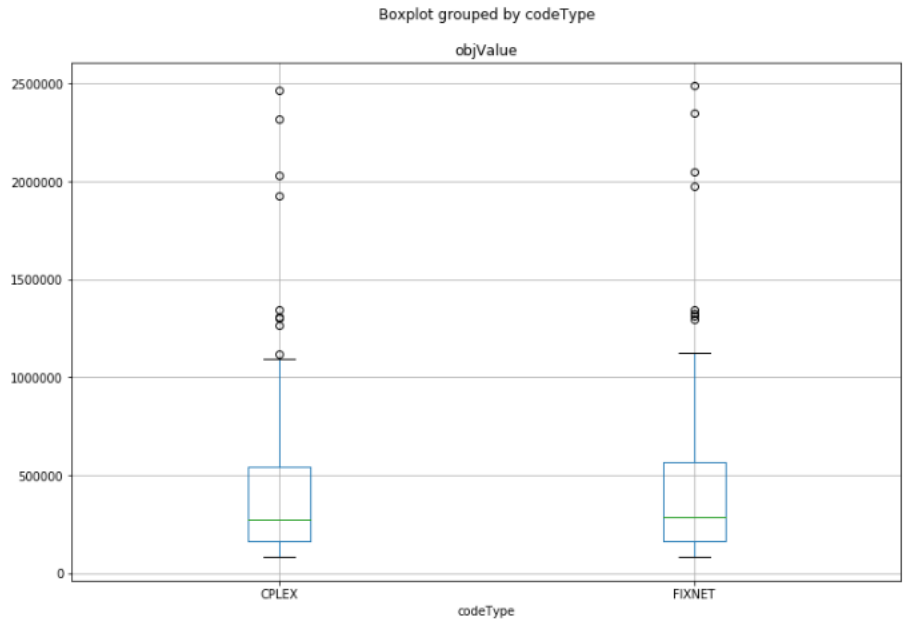
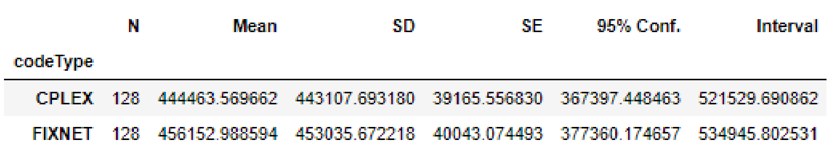



Figure 2.8. Box plot of objective value obtained by CPLEX and FIXNET codes

there is no statistically significant difference between CPLEX and FIXNET solution values with $F(1, 254) = 0.044$ and $p = 0.8349$ and descriptive statistics as shown in Figure 2.9.



| codeType | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|
| CPLEX | 128 | 444463.569662 | 443107.693180 | 39165.556830 | 367397.448463 | 521529.690862 |
| FIXNET | 128 | 456152.988594 | 453035.672218 | 40043.074493 | 377360.174657 | 534945.802531 |

Figure 2.9. Solution Value Descriptive Statistics by Code Type

A Tukey post-hoc testing shows both codes are in the same group. While code type and minimum level of upper bound are not statistically significant factors for the solution quality, the overall model shows statistical significance for the number of nodes and arcs factor. The analysis indicates that there is no statistically significant difference between solution values for problems with 130 nodes and 3000 arcs and problems with 150 nodes and 5000 arcs. The model determines there is a statistically significant difference between solution values for problems with different fixed-charges and also with different multipliers. The overall model of multiplier types, fixed-charge types and their interaction have $F(63, 192) = 2218.340$ with $p = 0.0000$. The model's overall coefficient of determination $R^2 = 0.99$ with fixed-charges type being the most influential factor on solution quality as shown in Figures 2.10 and 2.11.

| | sum_sq | mean_sq | df | F | PR(>F) | eta_sq | omega_sq |
|---|---|---|---|---|---|---|---|
| **numNA** | 6.113587e+11 | 6.113587e+11 | 1.0 | 1677.275589 | 7.930131e-97 | 0.011985 | 0.011978 |
| **multType** | 6.443407e+12 | 2.147802e+12 | 3.0 | 5892.541561 | 1.088420e-188 | 0.126316 | 0.126294 |
| **FCtype** | 3.793904e+13 | 5.419863e+12 | 7.0 | 14869.511034 | 7.978095e-259 | 0.743755 | 0.743700 |
| **numNA:multType** | 1.382206e+11 | 4.607355e+10 | 3.0 | 126.403765 | 3.175989e-45 | 0.002710 | 0.002688 |
| **numNA:FCtype** | 9.400484e+10 | 1.342926e+10 | 7.0 | 36.843470 | 2.266050e-32 | 0.001843 | 0.001793 |
| **multType:FCtype** | 5.618418e+12 | 2.675437e+11 | 21.0 | 734.011890 | 3.847089e-171 | 0.110143 | 0.109992 |
| **numNA:multType:FCtype** | 9.569923e+10 | 4.557106e+09 | 21.0 | 12.502518 | 6.787733e-26 | 0.001876 | 0.001726 |
| **Residual** | 6.998305e+10 | 3.644951e+08 | 192.0 | NaN | NaN | NaN | NaN |

Figure 2.10. ANOVA model for Solution Value

A Tukey post-hoc testing shows problems of fixed-charges types of A, B, C, D, and E are in the same group as well as F, G, and H in the other group with two groups being different groups. Similar distinction was observed for pure transportation networks with fixed-charges between test problems due to fixed-charge types. Descriptive statistics are shown in Figure 2.12.

As for multipliers types, gainy networks are not in the same group as lossy networks based on their solution values, while lossy networks are different from mixed and pure. Descriptive statistics are shown in Figure 2.13.

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| Intercept | 6.027e+04 | 1.52e+04 | 3.974 | 0.000 | 3.04e+04 | 9.02e+04 |
| numNA[T.150-5000] | 9.774e+04 | 5280.181 | 18.510 | 0.000 | 8.73e+04 | 1.08e+05 |
| multType[T.lossy] | 1.35e+05 | 2.11e+04 | 6.394 | 0.000 | 9.34e+04 | 1.77e+05 |
| multType[T.mix] | 9954.2610 | 2.11e+04 | 0.471 | 0.638 | -3.17e+04 | 5.16e+04 |
| multType[T.pure] | 3.559e+04 | 2.11e+04 | 1.685 | 0.093 | -6034.797 | 7.72e+04 |
| FCtype[T.B] | 1.27e+04 | 2.11e+04 | 0.601 | 0.548 | -2.89e+04 | 5.43e+04 |
| FCtype[T.C] | 2.666e+04 | 2.11e+04 | 1.262 | 0.208 | -1.5e+04 | 6.83e+04 |
| FCtype[T.D] | 5.851e+04 | 2.11e+04 | 2.770 | 0.006 | 1.69e+04 | 1e+05 |
| FCtype[T.E] | 1.156e+05 | 2.11e+04 | 5.473 | 0.000 | 7.4e+04 | 1.57e+05 |
| FCtype[T.F] | 2.231e+05 | 2.11e+04 | 10.564 | 0.000 | 1.81e+05 | 2.65e+05 |
| FCtype[T.G] | 4.335e+05 | 2.11e+04 | 20.523 | 0.000 | 3.92e+05 | 4.75e+05 |
| FCtype[T.H] | 8.271e+05 | 2.11e+04 | 39.158 | 0.000 | 7.85e+05 | 8.69e+05 |
| multType[T.lossy]:FCtype[T.B] | 6104.9383 | 2.99e+04 | 0.204 | 0.838 | -5.28e+04 | 6.5e+04 |
| multType[T.mix]:FCtype[T.B] | -3562.3191 | 2.99e+04 | -0.119 | 0.905 | -6.24e+04 | 5.53e+04 |
| multType[T.pure]:FCtype[T.B] | -301.6827 | 2.99e+04 | -0.010 | 0.992 | -5.92e+04 | 5.86e+04 |
| multType[T.lossy]:FCtype[T.C] | 2.226e+04 | 2.99e+04 | 0.745 | 0.457 | -3.66e+04 | 8.11e+04 |
| multType[T.mix]:FCtype[T.C] | -808.7300 | 2.99e+04 | -0.027 | 0.978 | -5.97e+04 | 5.81e+04 |
| multType[T.pure]:FCtype[T.C] | 1.009e+04 | 2.99e+04 | 0.338 | 0.736 | -4.88e+04 | 6.9e+04 |
| multType[T.lossy]:FCtype[T.D] | 4.285e+04 | 2.99e+04 | 1.435 | 0.153 | -1.6e+04 | 1.02e+05 |
| multType[T.mix]:FCtype[T.D] | 15.3601 | 2.99e+04 | 0.001 | 1.000 | -5.88e+04 | 5.89e+04 |
| multType[T.pure]:FCtype[T.D] | 1.67e+04 | 2.99e+04 | 0.559 | 0.577 | -4.22e+04 | 7.56e+04 |
| multType[T.lossy]:FCtype[T.E] | 1.154e+05 | 2.99e+04 | 3.862 | 0.000 | 5.65e+04 | 1.74e+05 |
| multType[T.mix]:FCtype[T.E] | 1.101e+04 | 2.99e+04 | 0.369 | 0.713 | -4.79e+04 | 6.99e+04 |
| multType[T.pure]:FCtype[T.E] | 3.431e+04 | 2.99e+04 | 1.149 | 0.252 | -2.45e+04 | 9.32e+04 |
| multType[T.lossy]:FCtype[T.F] | 2.595e+05 | 2.99e+04 | 8.687 | 0.000 | 2.01e+05 | 3.18e+05 |
| multType[T.mix]:FCtype[T.F] | 1.563e+04 | 2.99e+04 | 0.523 | 0.601 | -4.32e+04 | 7.45e+04 |
| multType[T.pure]:FCtype[T.F] | 6.529e+04 | 2.99e+04 | 2.186 | 0.030 | 6429.002 | 1.24e+05 |
| multType[T.lossy]:FCtype[T.G] | 5.152e+05 | 2.99e+04 | 17.247 | 0.000 | 4.56e+05 | 5.74e+05 |
| multType[T.mix]:FCtype[T.G] | 2.049e+04 | 2.99e+04 | 0.686 | 0.493 | -3.84e+04 | 7.93e+04 |
| multType[T.pure]:FCtype[T.G] | 1.267e+05 | 2.99e+04 | 4.242 | 0.000 | 6.78e+04 | 1.86e+05 |
| multType[T.lossy]:FCtype[T.H] | 1.13e+06 | 2.99e+04 | 37.823 | 0.000 | 1.07e+06 | 1.19e+06 |
| multType[T.mix]:FCtype[T.H] | 4.847e+04 | 2.99e+04 | 1.623 | 0.106 | -1.04e+04 | 1.07e+05 |

Figure 2.11. Regression Statistics for Solution Value

| codeType | FCtype | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|---|
| CPLEX | A | 16 | 1.540600e+05 | 67198.909243 | 16799.727311 | 1.200527e+05 | 1.880673e+05 |
| | B | 16 | 1.670604e+05 | 69963.234577 | 17490.808644 | 1.316541e+05 | 2.024667e+05 |
| | C | 16 | 1.880947e+05 | 75958.950774 | 18989.737693 | 1.496542e+05 | 2.265352e+05 |
| | D | 16 | 2.261335e+05 | 83832.904927 | 20958.226232 | 1.837082e+05 | 2.685588e+05 |
| | E | 16 | 3.062294e+05 | 113005.918559 | 28251.479640 | 2.490405e+05 | 3.634183e+05 |
| | F | 16 | 4.562761e+05 | 173124.985546 | 43281.246387 | 3.686627e+05 | 5.438894e+05 |
| | G | 16 | 7.418875e+05 | 281730.327557 | 70432.581889 | 5.993123e+05 | 8.844627e+05 |
| | H | 16 | 1.315967e+06 | 547746.048452 | 136936.512113 | 1.038769e+06 | 1.593165e+06 |
| FIXNET | A | 16 | 1.545112e+05 | 67098.270338 | 16774.567585 | 1.205547e+05 | 1.884676e+05 |
| | B | 16 | 1.680325e+05 | 69647.844108 | 17411.961027 | 1.327858e+05 | 2.032792e+05 |
| | C | 16 | 1.895576e+05 | 75264.913402 | 18816.228350 | 1.514683e+05 | 2.276469e+05 |
| | D | 16 | 2.292317e+05 | 83209.769581 | 20802.442395 | 1.871218e+05 | 2.713417e+05 |
| | E | 16 | 3.138551e+05 | 112992.735521 | 28248.183880 | 2.566729e+05 | 3.710373e+05 |
| | F | 16 | 4.687367e+05 | 170485.699627 | 42621.424907 | 3.824590e+05 | 5.550143e+05 |
| | G | 16 | 7.647793e+05 | 275841.985238 | 68960.496310 | 6.251840e+05 | 9.043746e+05 |
| | H | 16 | 1.360520e+06 | 534542.311392 | 133635.577848 | 1.090004e+06 | 1.631036e+06 |

Figure 2.12. Solution Value Descriptive Statistics by Code Type and Fixed-charge Type

| codeType | multType | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|---|
| CPLEX | gainy | 32 | 314046.489630 | 266193.674964 | 47056.838169 | 220339.290666 | 407753.688594 |
| | lossy | 32 | 713689.334177 | 649061.001226 | 114738.858843 | 485202.731745 | 942175.936609 |
| | mixed | 32 | 332392.861008 | 277266.570268 | 49014.268008 | 234787.710662 | 429998.011354 |
| | pure | 32 | 417725.593833 | 359557.865909 | 63561.451303 | 291151.735592 | 544299.452075 |
| FIXNET | gainy | 32 | 328501.548125 | 283792.653272 | 50167.927395 | 228599.044325 | 428404.051925 |
| | lossy | 32 | 721693.138437 | 658920.590567 | 116481.804463 | 489735.700344 | 953650.576531 |
| | mixed | 32 | 352874.133750 | 304923.533967 | 53903.374653 | 245533.002260 | 460215.265240 |
| | pure | 32 | 421543.134062 | 362326.666146 | 64050.910659 | 293994.585064 | 549091.683061 |

Figure 2.13. Solution Value Descriptive Statistics by Code Type and Multiplier Type

*2.5.5.2. Analysis of Solution Time due to Code Type*

To support the second hypothesis, an analysis of variance was performed to determine factors affecting the solution times. The factors considered were the code type, the total number of nodes and arcs, minimum value of upper bound on an arc, multiplier type (mixed, lossy, gainy, and pure networks), and fixed-charge type. A boxplot of solution times for CPLEX and FIXNET clearly shows that FIXNET performance was faster as in Figure 2.14. One important factor for this analysis is to determine if the solution time difference is due to the code type or any interactions with code type.

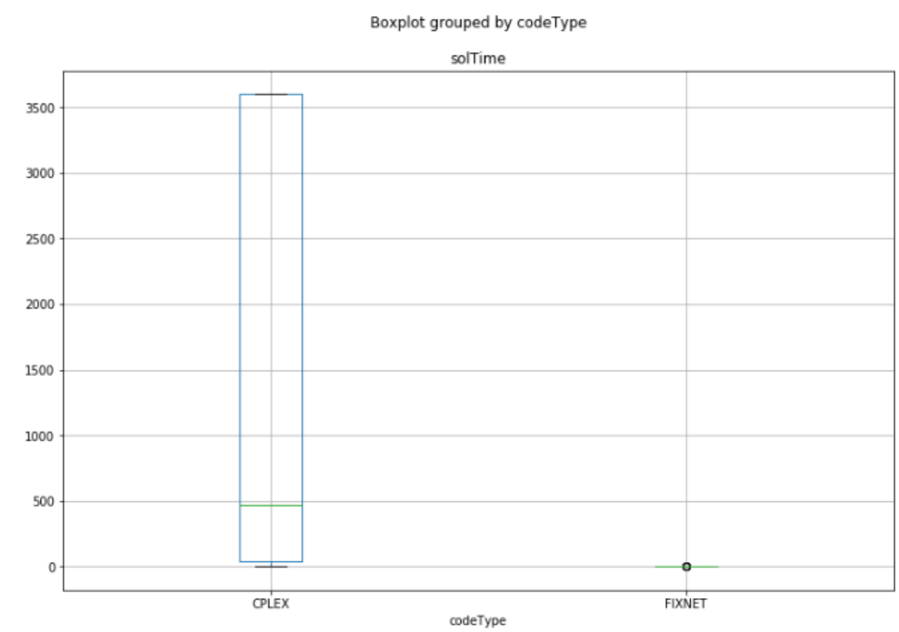



Figure 2.14. Box plot of solution times spent by CPLEX and FIXNET codes to find a solution

All 256 observations were used for analysis using the ANOVA procedure with Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] Jupyter notebook, server version 5.7.4. with significance level of 5%. The results show that there is a statistically significant difference in the time to obtain a solution by CPLEX and by the FIXNET code with overall model $F(1, 254) = 116.662$, $p = 0.0000$ and descriptive statistics as shown in Figure 2.15. The Tukey test shows that the FIXNET code performs faster than CPLEX to find a solution. On average for all tested problems FIXNET was more than $56,000$ times faster than CPLEX with optimality gap on average 2.215%.

| codeType | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|
| CPLEX | 128 | 1627.295669 | 1704.504727 | 150.658356 | 1330.845010 | 1923.746329 |
| FIXNET | 128 | 0.029601 | 0.017856 | 0.001578 | 0.026495 | 0.032706 |

Figure 2.15. Solution Time Descriptive Statistics by Code Type

There was no statistically significant difference in the solution time due to minimum upper bound value. All other factors including number of nodes and arcs, multiplier types, and fixed charge types showed statistically significant effect on solution time including second order interactions of code type with number of nodes and arcs, with fixed-charges types, and with multiplier types as shown in Figure 2.16. Model's overall coefficient of determination $R^2 = 0.904$. Descriptive statistics for solution time by number of nodes and arcs and by fixed-charge type are shown in Figures 2.17 and 2.18.

While code type is the factor that affects solution time the most, the next important factor is multiplier type. The Tukey post-hoc testing showed there are distinctive groups for multipliers types with descriptive statistics as shown on Figure 2.19. It also showed that $130 - 3000$ and $150 - 5000$ size networks are in the same group for solution time as well as problems with different fixed-charge types. The last testing shows that generated problems with larger range of fixed charges, which were con-

| | sum_sq | mean_sq | df | F | PR(>F) | eta_sq | omega_sq |
|---|---|---|---|---|---|---|---|
| **numNA** | 1.146171e+06 | 1.146171e+06 | 1.0 | 4.231746 | 4.103362e-02 | 0.002129 | 0.001625 |
| **codeType** | 1.694717e+08 | 1.694717e+08 | 1.0 | 625.701678 | 3.586296e-62 | 0.314740 | 0.314079 |
| **multType** | 1.401720e+08 | 4.672399e+07 | 3.0 | 172.508350 | 4.041444e-54 | 0.260325 | 0.258686 |
| **FCtype** | 5.491180e+06 | 7.844543e+05 | 7.0 | 2.896262 | 6.711806e-03 | 0.010198 | 0.006674 |
| **codeType:multType** | 1.401682e+08 | 4.672273e+07 | 3.0 | 172.503699 | 4.049036e-54 | 0.260318 | 0.258679 |
| **codeType:FCtype** | 5.491176e+06 | 7.844537e+05 | 7.0 | 2.896260 | 6.711845e-03 | 0.010198 | 0.006674 |
| **multType:FCtype** | 1.238824e+07 | 5.899162e+05 | 21.0 | 2.178013 | 3.154783e-03 | 0.023007 | 0.012438 |
| **codeType:multType:FCtype** | 1.238833e+07 | 5.899206e+05 | 21.0 | 2.178029 | 3.154520e-03 | 0.023007 | 0.012438 |
| **Residual** | 5.173246e+07 | 2.708506e+05 | 191.0 | NaN | NaN | NaN | NaN |

Figure 2.16. ANOVA model for Solution Time

| | | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|---|
| **codeType** | **numNA** | | | | | | |
| **CPLEX** | **130-3000** | 64 | 1493.476274 | 1722.196886 | 215.274611 | 1068.202505 | 1918.750043 |
| | **150-5000** | 64 | 1761.115065 | 1689.509782 | 211.188723 | 1343.912944 | 2178.317186 |
| **FIXNET** | **130-3000** | 64 | 0.024778 | 0.015474 | 0.001934 | 0.020957 | 0.028599 |
| | **150-5000** | 64 | 0.034423 | 0.018868 | 0.002359 | 0.029764 | 0.039082 |

Figure 2.17. Solution Time Descriptive Statistics by Code Type and Number of Nodes/Arcs

| codeType | FCtype | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|---|
| CPLEX | A | 16 | 1001.767875 | 1553.184046 | 388.296011 | 215.748298 | 1787.787452 |
| | B | 16 | 1456.760286 | 1737.572946 | 434.393236 | 577.427052 | 2336.093521 |
| | C | 16 | 1670.739845 | 1783.247941 | 445.811985 | 768.291874 | 2573.187817 |
| | D | 16 | 1619.732544 | 1659.756595 | 414.939149 | 779.779816 | 2459.685272 |
| | E | 16 | 1707.568691 | 1736.957804 | 434.239451 | 828.546761 | 2586.590621 |
| | F | 16 | 1592.154549 | 1828.932904 | 457.233226 | 666.586797 | 2517.722302 |
| | G | 16 | 1965.349394 | 1739.154972 | 434.788743 | 1085.215544 | 2845.483245 |
| | H | 16 | 2004.292169 | 1764.475518 | 441.118879 | 1111.344354 | 2897.239983 |
| FIXNET | A | 16 | 0.029296 | 0.018379 | 0.004595 | 0.019995 | 0.038597 |
| | B | 16 | 0.027342 | 0.012104 | 0.003026 | 0.021217 | 0.033468 |
| | C | 16 | 0.031737 | 0.020665 | 0.005166 | 0.021279 | 0.042195 |
| | D | 16 | 0.034178 | 0.028224 | 0.007056 | 0.019895 | 0.048462 |
| | E | 16 | 0.025388 | 0.014409 | 0.003602 | 0.018096 | 0.032680 |
| | F | 16 | 0.029784 | 0.014039 | 0.003510 | 0.022679 | 0.036889 |
| | G | 16 | 0.032225 | 0.018015 | 0.004504 | 0.023108 | 0.041342 |
| | H | 16 | 0.026854 | 0.013967 | 0.003492 | 0.019786 | 0.033923 |

Figure 2.18. Solution Time Descriptive Statistics by Code Type and Fixed Charges Type

sidered difficult problems to solve for pure fixed-charge transportation problems in [44, 104], can be solved on average to 3.7% of optimality compared to the commercial solver CPLEX in 0.09 seconds in the worst case.

| codeType | multType | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|---|
| CPLEX | gainy | 32 | 138.717228 | 310.609645 | 54.908546 | 29.374436 | 248.060021 |
| | lossy | 32 | 2511.290614 | 1563.372310 | 276.367790 | 1960.942301 | 3061.638928 |
| | mixed | 32 | 259.174834 | 564.504125 | 99.791174 | 60.454484 | 457.895185 |
| | pure | 32 | 3600.000000 | 0.000000 | 0.000000 | 3600.000000 | 3600.000000 |
| FIXNET | gainy | 32 | 0.017087 | 0.007541 | 0.001333 | 0.014432 | 0.019742 |
| | lossy | 32 | 0.028074 | 0.015354 | 0.002714 | 0.022669 | 0.033479 |
| | mixed | 32 | 0.025390 | 0.009723 | 0.001719 | 0.021967 | 0.028812 |
| | pure | 32 | 0.047851 | 0.019820 | 0.003504 | 0.040874 | 0.054828 |

Figure 2.19. Solution Time Descriptive Statistics by Code Type and Multiplier Type

Overall for both codes, gainy types of networks ran faster than lossy and pure, mixed types ran faster than pure, and lossy ran slower than mixed types of networks. The analysis also shows that the FIXNET code dominated CPLEX when solving pure networks and lossy networks.

#### *2.5.5.3. Analysis of Solution Quality Variation*

Based on obtained solution values from all test problems solved by CPLEX and by FIXNET code, the $R$ metric is calculated as defined earlier. The analysis is used to determine if there are any factors which affect the value of $R$. The factors considered were the total number of nodes and arcs, minimum value of upper bound on an arc, multiplier type (mixed, lossy, gainy, and pure networks), and fixed-charge type. The analysis of solution quality showed there is no statistically significant difference between the two codes, but indicated multipliers types and fixed-charge types affected the solution quality. Descriptive statistics for $R$ are shown in Figure 2.20.

| | Variable | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|---|
| 0 | R | 256.0 | 1.02215 | 0.024172 | 0.001511 | 1.019175 | 1.025125 |

Figure 2.20. Descriptive Statistics for $R$

The analysis showed that there were several factors affecting the variability of the solutions with overall model $F(63, 192) = 49.235$ and $p = 0.0000$ with the coefficient of determination $R^2 = 0.942$, factors and their interactions as shown in Figure 2.21. Descriptive statistics for $R$ by number of nodes and arcs, multiplier type, and fixed-charge type are shown in Figures 2.22, 2.23, and 2.24 respectively.

| | sum_sq | df | F | PR(>F) |
|---|---|---|---|---|
| numNA | 0.000013 | 1.0 | 0.295380 | 5.874236e-01 |
| multType | 0.051632 | 3.0 | 380.474972 | 1.628667e-80 |
| FCtype | 0.049747 | 7.0 | 157.107524 | 6.377253e-76 |
| numNA:multType | 0.000616 | 3.0 | 4.541176 | 4.231532e-03 |
| numNA:FCtype | 0.001801 | 7.0 | 5.688115 | 5.476313e-06 |
| multType:FCtype | 0.033287 | 21.0 | 35.041515 | 2.334442e-54 |
| numNA:multType:FCtype | 0.003212 | 21.0 | 3.381406 | 3.987347e-06 |
| Residual | 0.008685 | 192.0 | NaN | NaN |

Figure 2.21. ANOVA model for $R$ metric

| numNA | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|
| 130-3000 | 128 | 1.021921 | 0.024885 | 0.00220 | 1.017593 | 1.026249 |
| 150-5000 | 128 | 1.022378 | 0.023533 | 0.00208 | 1.018285 | 1.026471 |

Figure 2.22. Descriptive Statistics for $R$ by Number of Nodes/Arcs

As in the pure fixed-charge transportation problems computational study by Sun et al. [104], problems with bigger fixed charges can be considered more difficult to

| multType | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|
| gainy | 64 | 1.032269 | 0.022525 | 0.002816 | 1.026706 | 1.037831 |
| lossy | 64 | 1.007562 | 0.007161 | 0.000895 | 1.005794 | 1.009330 |
| mixed | 64 | 1.039902 | 0.030693 | 0.003837 | 1.032322 | 1.047481 |
| pure | 64 | 1.008867 | 0.006686 | 0.000836 | 1.007216 | 1.010518 |

Figure 2.23. Descriptive Statistics for $R$ by Multiplier Type

| FCtype | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|
| A | 32 | 1.003600 | 0.002189 | 0.000387 | 1.002829 | 1.004370 |
| B | 32 | 1.007157 | 0.004775 | 0.000844 | 1.005477 | 1.008838 |
| C | 32 | 1.009825 | 0.007976 | 0.001410 | 1.007017 | 1.012633 |
| D | 32 | 1.016128 | 0.010978 | 0.001941 | 1.012263 | 1.019992 |
| E | 32 | 1.027602 | 0.017179 | 0.003037 | 1.021555 | 1.033650 |
| F | 32 | 1.032769 | 0.023997 | 0.004242 | 1.024321 | 1.041216 |
| G | 32 | 1.037450 | 0.029500 | 0.005215 | 1.027065 | 1.047835 |
| H | 32 | 1.042668 | 0.035335 | 0.006246 | 1.030229 | 1.055106 |

Figure 2.24. Descriptive Statistics for $R$ by Fixed-charge Type

solve to optimality. Descriptive statistics for $R$ by problem difficulty group is shown in Figure 2.25.

| | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|
| FCgroup | | | | | | |
| difficult | 96 | 1.037629 | 0.029931 | 0.003055 | 1.031610 | 1.043648 |
| easy | 160 | 1.012862 | 0.013052 | 0.001032 | 1.010834 | 1.014891 |

Figure 2.25. Descriptive Statistics for $R$ by Problem Difficulty Group

### 2.5.6. Conclusions

Computational testing shows the effectiveness of the proposed extreme-point tabu search heuristic by testing the results of 128 generated problems and conducting statistical analysis. As expected and shown with analysis, there is no statistically significant difference in solution quality between FIXNET and CPLEX results while there is a statistically significant difference in the solution time with FCGT heuristics performing more than 50,000 times faster than CPLEX providing 2.2% optimality gap on average for 128 tested problems.

## 2.6. Conclusions

The addition of fixed charges to generalized arcs of transportation network problems expands modeling capabilities of wide range in finance, production, distribution, transportation, and scheduling applications. In the absence of published solution approaches for fixed-charge generalized transportation problems, this chapter proposes an efficient heuristic to solve FCGT and provides computational testing that fills a research literature gap for such a problem type. The extreme-point tabu-search heuristic builds on the primal generalized network simplex method and uses special quasi-tree basis forest structure in its core. The proposed heuristic solution algorithm is implemented with the short, intermediate, and long-term memory processes such as candidate list, tabu, aspiration criteria, modified reduced cost that takes into account the effect of the fixed charges, and diversification phase to overcome local optimality.

Computationally it has been tested on 128 generated instances with newly proposed parameters for fixed-charge generalized networks. The analysis showed statistical significance of multipliers and fixed-charge ranges types on both solution quality and solution time while post-hoc testing revealed no differences in solution time for problems with different fixed-charge ranges. The proposed EPTS heuristic was able to find a better solution for one problem than commercial solver CPLEX and on average for all test problems it is more than 56,000 times faster than the state-of-the-art commercial solver CPLEX with an average optimality gap of 2.2%. CPLEX reached time limit of 3600 seconds for 53 out of 128 problems with average solution time of 1627.30 seconds compared to 0.03 seconds on average for the proposed extreme-point tabu-search heuristic for fixed-charge generalized transportation network problems.

# Chapter 3

## Dynamic Linearization Meta-Heuristics for Solving Fixed-Charge Generalized Transshipment Problems

The extreme-point tabu search meta-heuristic proposed in the previous chapter was based on a tabu search, candidate list, aspiration criteria, and diversify phase designed to overcome local optimality. It proved to be effective for the fixed-charge generalized transportation problems. In the absence of published algorithms for fixed-charge generalized transshipment networks, we developed the solution approach to solve transshipment problems of larger sizes, with number of nodes 5,000 or 10,000 and number of arcs 50,000 or 100,000 with incorporation of dynamic linearization of the objective function. A parametric approach for solving fixed-charge problems first was sketched by Glover in [42] and proposed as the parametric ghost image processes for fixed-charge transportation networks in [44]. The approach involves a parameterization of the objective function that is progressively modified by the meta-heuristic procedures that in turn use basic tabu search strategies. The inclusion of such approach variation to the heuristic proposed in this chapter incorporates a dynamically modified objective function with linear parameters in order to improve the solution quality for generated fixed-charge generalized transshipment problems.

The basic approach and a background for the parametric ghost image process is provided in [44] as well as an algorithmic approach to solve fixed-charge transportation problems. The author mentions that the implementation developed applies to fixed-charge problems using generalized networks with a two-multiplier generalized network solver GN2. However, no generalized networks parameters/factors were provided and no generalized problems were generated or analyzed in the paper, no computational results were available, and no discussion was provided regarding implementation of the special basis forest structure for generalized network models. In addition, to the best of our knowledge, there is no published research for fixed-charge generalized transshipment problems currently available.

The fixed-charge generalized transshipment network model can be defined mathematically as follows:

$$
\begin{aligned}
FCGN: \quad \text{Minimize} \sum_{(i,j)\in\mathcal{A}} (c_{ij}x_{ij} + f_{ij}y_{ij}) = FC(x) & & & \quad (3.1)\\
\text{subject to:} \quad \sum_{j:(i,j)\in\mathcal{A}} x_{ij} - \sum_{j:(j,i)\in\mathcal{A}} \mu_{ji}x_{ji} &= & b_i, \forall i \in \mathcal{N} & \quad (3.2)\\
0 \leq x_{ij} &\leq & u_{ij}y_{ij}, \forall (i,j) \in \mathcal{A} & \quad (3.3)\\
0 \leq y_{ij} &\leq & 1, \forall (i,j) \in \mathcal{A} & \quad (3.4)\\
y_{ij} & & integer, \forall (i,j) \in \mathcal{A} & \quad (3.5)
\end{aligned}
$$

### 3.1. Algorithmic Approach for FCGN Heuristic

The variant of the parametric ghost image approach, called *dynamic linearization* (DL) of the objective function, is proposed for use in the modified extreme-point tabu search meta-heuristic, discussed in the previous chapter, with the intention to improve the solution quality. It is expected that the solution time will be increased due to the fact that several relaxations and local optimum procedures will be repeated. However, the goal is to improve the quality of the solution. The dynamic linearization of the objective function heuristic starts with the solution of the relaxation problems as in the FCGT heuristic to determine the lower and upper bounds for the objective value:

$$GN\text{:} \quad \text{Minimize} \sum_{(i,j)\in\mathcal{A}} (c_{ij} + f_{ij}/u_{ij})x_{ij} = LC(x) \tag{3.6}$$

$$\text{subject to:} \quad \sum_{j:(i,j)\in\mathcal{A}} x_{ij} - \sum_{j:(j,i)\in\mathcal{A}} \mu_{ji}x_{ji} = b_i, \forall i \in \mathcal{N} \tag{3.7}$$

$$0 \leq x_{ij} \leq u_{ij}, \forall (i,j) \in \mathcal{A} \tag{3.8}$$

where $(c_{ij} + f_{ij}/u_{ij})x_{ij}$ is the approximation of total cost that take into consideration the proportion of fixed cost on arc $(i,j)$.

Similar to the concepts described in [44], a non-negative parameter vector $\mathbf{v} = (v_{ij} : (i,j) \in \mathcal{A})$ is introduced to parameterize the proportion of fixed-charge to the total cost of the objective function by calculating $1/v_{ij}$. For the original relaxation problem, the "parameterized penalty," or proportion of the fixed charges to the total cost, is calculated as $f_{ij}/u_{ij}$. In this case we can set $v_{ij} = u_{ij}$ to initialize $\mathbf{v}$. After finding the first local optimum, the flow $x_{ij}$ is determined, and then the total cost can be linearized and approximated by solving the modified relaxation problem with the objective $\sum_{(i,j)\in\mathcal{A}}(c_{ij} + f_{ij}/v_{ij})x_{ij}$, with $v_{ij}$ being updated as a function of its current value and the solution $x$. Graphically, it can be viewed as in Figure 3.1.

The succession of LPs with dynamically updated linearized objective function in alternation with tabu search, intensify and diversify phases, is solved to produce an improved solution. Let $m_1$ be the maximum number for the objective function linearizations, $m_2$ be the maximum number of the diversify phase performed, $m_3$ be the number of nonbasic arcs brought to the basis to diversify the solution region search, and $m_4$ be the maximum number of moves or iterations. An outline of the method FCGN can be described as following:

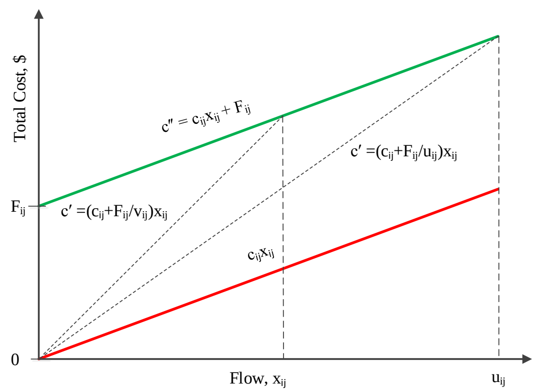



Figure 3.1. Total Cost vs Variable Cost

---

**Algorithm 3.1** FCGN Dynamic Linearization Heuristic for the Fixed-charge Generalized Transshipment Network

---

**Require:** $\mathcal{P}, m_1, m_2, m_3, m_4$

**Ensure:** $\mathbf{x}^{\bullet}, z^{\bullet}$

$z^{\bullet} \leftarrow \infty$, Iter $\leftarrow 0$ ▷ Initialize values

$\mathbf{x}' \leftarrow$ LPRSolve($\mathbf{v}$)

$\mathbf{x}'' \leftarrow$ LocSearch($\mathbf{x}'$), $\mathbf{z}'' \leftarrow$ ZF($\mathbf{x}''$)

IncumbentUpdate($\mathbf{x}^{\bullet}, z^{\bullet}, \mathbf{x}''$)

**while** primary iter LE $m_1$ and global Iter LE $m_4$ **do**

  $\mathbf{v} \leftarrow$ Vupdate($\mathbf{v}, \mathbf{x}'$) ▷ Update $\mathbf{v}$ using $x'$

  $\mathbf{x}' \leftarrow$ LPRSolve($\mathbf{v}$)

  $\mathbf{x}'' \leftarrow$ LocSearch($\mathbf{x}'$)

  IncumbentUpdate($\mathbf{x}^{\bullet}, z^{\bullet}, \mathbf{x}''$)

  **for** $k = 1,\ m_2$ **do** ▷ Diversification inner loop

    $\mathbf{x}' \leftarrow$ Diversify($\mathbf{x}'', m_3$)

    $\mathbf{x}'' \leftarrow$ LocSearch($\mathbf{x}'$)

    IncumbentUpdate($\mathbf{x}^{\bullet}, z^{\bullet}, \mathbf{x}''$)

  **end for**

**end while**

Return $\mathbf{x}^{\bullet}, z^{\bullet}$

---

**Algorithm 3.2** LPRSolve($\mathbf{v}$) Procedure

**Require:** $\mathbf{v}, \mathbf{c}, \mathbf{f}, \mu, \mathbf{b}, \mathbf{u}$

**Ensure:** $\mathbf{x}^{\bullet}, z^{\bullet}$

1: Minimize $\sum_{(i,j)\in A}(c_{ij} + \frac{f_{ij}}{v_{ij}})x_{ij}$, subject to:

2: $\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} \mu_{ji}x_{ji} = b_i, \forall i \in N$

3: $0 \le x_{ij} \le u_{ij}, \forall (i,j) \in A$

4: Return $\mathbf{x}^*$ ▷ Optimal solution to relaxation using $\mathbf{v}$

**Algorithm 3.3** Vupdate ($\mathbf{v}$) Procedure

**Require:** $\mathbf{v}, \mathbf{x}, \mathbf{maxFlow}, MaxSol, \beta, \alpha_1, \alpha_2$

**Ensure:** $\mathbf{v}$

1: $NumSol \leftarrow NumSol + 1$

2: $Y = 1/min\{NumSol, MaxSol\}$

3: **for** $(i,j) \in \mathcal{A}$ **do**

4: $\quad Mean_{ij} = Yx_{ij} + (1-Y)Mean_{ij}$

5: $\quad UMean = \beta Mean_{ij} + (1-\beta)MaxFlow_{ij}$

6: $\quad v_{ij} = \alpha_1 x_{ij} + \alpha_2 v_{ij} + (1-\alpha_1-\alpha_2)UMean$

7: **end for**

8: Return $\mathbf{v}$ ▷ adjusted value of $\mathbf{v}$

The move evaluation process calculates the *total reduced cost*, $\kappa_{ij}$, which gives the effect on FCGN's $FC(x)$ objective if nonbasic arc $(i,j)$ is pivoted into the basis at its maximum allowed flow level, $\delta$. The steps are given in Algorithm 3.4.

**Algorithm 3.4** Total reduced cost calculation, $\kappa_{ij}$, for nonbasic $(i, j)$

1: Compute the representation and BEP for $(i, j)$ using Algorithm 2.1,

2: Apply the ratio test along the BEP to determine flow change $\delta$.

3: Retrace the BEP to determine the total reduced cost $\kappa_{ij}$ as:

$$\kappa_{ij} = \delta \overline{c}_{ij} + \sum_{(k,l) \in BEP(i,j) \cup (i,j)} \phi(k, \ell, \delta)$$

$$\text{where } \phi(k, \ell, \delta) = \begin{cases} F_{k\ell} & \text{if } x_{k\ell} = 0, \text{flow increases, and } \delta > 0 \\ -F_{k\ell} & \text{if } x_{k\ell} > 0, \text{flow decreases, and } \delta = x_{k\ell} \\ 0 & \text{otherwise} \end{cases}$$

and $\overline{c}_{ij} = c_{ij} - \pi_i + \mu_{ij}\pi_j$ using duals based on variable costs only.

Although the effort to compute $\kappa_{ij}$ is significantly higher than simple arc pricing with duals, it provides the exact value of the pivot's effect on the current objective $FC(x)$. For example, the ratio test might find that $\delta = 0$ and a pivot would result in a degenerate solution with no improvement in objective, despite an attractive $\overline{c}_{ij}$. Fortunately the second and third BEP traversals are expedited by knowing the previously computed BEP and representation values, so that $\phi(k, \ell, \delta)$ can be quickly determined.

## 3.2. Computational Testing

This section describes the experimental design used to test the effectiveness of the proposed heuristic approach against a commercially available state-of-the-art solver. The experiment is designed to test the quality of solution and the solution time. The software used and the problems set are described in the following paragraphs.

### 3.2.1. Commercial Software Description

The commercial software used for comparison purposes is CPLEX version 12.6.0.0 from IBM at Southern Methodist University's Lyle School of Engineering with default settings, single thread mode, and a time limit of 3600 seconds. Only the time that CPLEX solver used to solve the problem is used for comparison. The time limit is altered to ensure a timely termination of testing.

### 3.2.2. FIXNET Software Description

The base for the implementation of the EPTS heuristic for the fixed-charge generalized transportation problem, a one-multiplier generalized network solver FIXNET, developed by the author and written in FORTRAN, can solve uncapacitated and capacitated generalized and pure networks, including transportation and transshipment structures. The current implementation extends the capabilities of GN to solve the class of fixed-charge generalized transportation problems.

The data structure for nodes holds the node potential, requirements (supply/demand), and quasi-tree structure for each node. The tree data is maintained using the concept of threads as defined by Barr et al. in [11] additionally storing the information about loop factors for the nodes that belong to one-tree roots or cycles. The data structure for arcs holds all the information for each arc including from and to nodes, upper bounds, conditional lower bounds, a flag to determine if the arc is part of basic set $F$, the reduced cost and total reduced cost as part of the arc selection process.

The FIXNET code captures multiple statistics as it solves each test problem including but not limited to the relaxed solution, total cost for the relaxed solution, local search solution, variable and fixed cost at the local search solution, number of degenerate pivots, number of arcs at upper bound for the local search solution with total flow at upper bound, number of times aspiration criteria was applied, number of rooted trees and number of nodes on cycles for the relaxed solution and for the local search solution to analyze the basis forest structure for the solution. Timing statistics are captured for several sections of the code and include time for relaxed solution and overall solution time which also includes reading from the data file and reinverting the network to eliminate round-off errors and recalculate node duals for different costs.

### 3.2.3. Test Environment

General use Linux machine Dell R730 with Intel Xeon@2.6 GHz, 320GB RAM was used as the computing environment to run all test problems with the FIXNET code written in FORTRAN and compiled using gfortran and CPLEX for comparison.

### 3.2.4. Problem Set Definition and Generation

The test set used for testing the solution quality and the solution time was designed to compare FIXNET performance with the state-of-the-art solver CPLEX. The results are compared using statistical analysis to determine if there are differences in quality of the solution and in solution time. The test set main factors and levels shown in Table 3.1.

Table 3.1. Transshipment Networks Test Set: Problem Factors and Levels

| Factors | Levels | |
|---|---|---|
| Number of nodes | 5,000 | 10,000 |
| Number of arcs (nested within nodes) | 50,000 | 100,000 |
| Percent source nodes | 2% | 5% |
| Percent demand nodes | 2% | 5% |
| Percent arcs with fixed charges | 25% | 50% |
| Range of fixed-charges on an arc | 1,000 to 2,000 | 10,000 to 20,000 |

The test problem definitions are shown in Tables 3.2 – 3.5. The total supply for each test problem was fixed at 100,000 with the variable cost for each arc ranged from 1 to 50 and the arc multipliers being in the interval $[0.5, 1.5]$. All test problems are 100% capacitated with the upper bound being in the range $[1000, 2000]$.

Table 3.2. Test Set Transshipment Problems with 5,000 nodes and 50,000 arcs

| Problem ID | Number of Nodes | Number of Arcs | Source Nodes | Sink Nodes | % FC Arcs | Fixed-Charge Range |
|---|---|---|---|---|---|---|
| 1-5000-50000 | 5,000 | 50,000 | 100 | 100 | 25 | [1000,2000] |
| 2-5000-50000 | 5,000 | 50,000 | 100 | 250 | 25 | [1000,2000] |
| 3-5000-50000 | 5,000 | 50,000 | 250 | 100 | 25 | [1000,2000] |
| 4-5000-50000 | 5,000 | 50,000 | 250 | 250 | 25 | [1000,2000] |
| 5-5000-50000 | 5,000 | 50,000 | 100 | 100 | 25 | [10000,20000] |
| 6-5000-50000 | 5,000 | 50,000 | 100 | 250 | 25 | [10000,20000] |
| 7-5000-50000 | 5,000 | 50,000 | 250 | 100 | 25 | [10000,20000] |
| 8-5000-50000 | 5,000 | 50,000 | 250 | 250 | 25 | [10000,20000] |
| 9-5000-50000 | 5,000 | 50,000 | 100 | 100 | 50 | [1000,2000] |
| 10-5000-50000 | 5,000 | 50,000 | 100 | 250 | 50 | [1000,2000] |
| 11-5000-50000 | 5,000 | 50,000 | 250 | 100 | 50 | [1000,2000] |
| 12-5000-50000 | 5,000 | 50,000 | 250 | 250 | 50 | [1000,2000] |
| 13-5000-50000 | 5,000 | 50,000 | 100 | 100 | 50 | [10000,20000] |
| 14-5000-50000 | 5,000 | 50,000 | 100 | 250 | 50 | [10000,20000] |
| 15-5000-50000 | 5,000 | 50,000 | 250 | 100 | 50 | [10000,20000] |
| 16-5000-50000 | 5,000 | 50,000 | 250 | 250 | 50 | [10000,20000] |

Table 3.3. Test Set Transshipment Problems with 10,000 nodes and 50,000 arcs

| Problem ID | Number of Nodes | Number of Arcs | Source Nodes | Sink Nodes | % FC Arcs | Fixed-Charge Range |
|---|---|---|---|---|---|---|
| 1-10000-50000 | 10,000 | 50,000 | 200 | 200 | 25 | [1000,2000] |
| 2-10000-50000 | 10,000 | 50,000 | 200 | 500 | 25 | [1000,2000] |
| 3-10000-50000 | 10,000 | 50,000 | 500 | 200 | 25 | [1000,2000] |
| 4-10000-50000 | 10,000 | 50,000 | 500 | 500 | 25 | [1000,2000] |
| 5-10000-50000 | 10,000 | 50,000 | 200 | 200 | 25 | [10000,20000] |
| 6-10000-50000 | 10,000 | 50,000 | 200 | 500 | 25 | [10000,20000] |
| 7-10000-50000 | 10,000 | 50,000 | 500 | 200 | 25 | [10000,20000] |
| 8-10000-50000 | 10,000 | 50,000 | 500 | 500 | 25 | [10000,20000] |
| 9-10000-50000 | 10,000 | 50,000 | 200 | 200 | 50 | [1000,2000] |
| 10-10000-50000 | 10,000 | 50,000 | 200 | 500 | 50 | [1000,2000] |
| 11-10000-50000 | 10,000 | 50,000 | 500 | 200 | 50 | [1000,2000] |
| 12-10000-50000 | 10,000 | 50,000 | 500 | 500 | 50 | [1000,2000] |
| 13-10000-50000 | 10,000 | 50,000 | 200 | 200 | 50 | [10000,20000] |
| 14-10000-50000 | 10,000 | 50,000 | 200 | 500 | 50 | [10000,20000] |
| 15-10000-50000 | 10,000 | 50,000 | 500 | 200 | 50 | [10000,20000] |
| 16-10000-50000 | 10,000 | 50,000 | 500 | 500 | 50 | [10000,20000] |

### 3.2.5. Test Results

User specified parameters such as candidate list strategy $(15, 20)$ for its length, 20, and number of pivots between replenishment, 15, Tabu parameter 25, number the linearizations of the objective funtion, 20, and number of nonbasic arcs forced to enter basis, 5, are used for all test problems solved by FIXNET solver. Parameter calibration were run on 555 problems to test three different values for the number of the linearizations and five different values for the number of nonbasic arcs being nonbasic the longest to enter the basis. Candidate list strategy $(15, 20)$, tabu parameter 25, number of the objective function linearizations 20, and the number of nonbasic

Table 3.4. Test Set Transshipment Problems with 5,000 nodes and 100,000 arcs

| Problem ID | Number of Nodes | Number of Arcs | Source Nodes | Sink Nodes | % FC Arcs | Fixed-Charge Range |
|---|---|---|---|---|---|---|
| 1-5000-100000 | 5,000 | 100,000 | 100 | 100 | 25 | [1000,2000] |
| 2-5000-100000 | 5,000 | 100,000 | 100 | 250 | 25 | [1000,2000] |
| 3-5000-100000 | 5,000 | 100,000 | 250 | 100 | 25 | [1000,2000] |
| 4-5000-100000 | 5,000 | 100,000 | 250 | 250 | 25 | [1000,2000] |
| 5-5000-100000 | 5,000 | 100,000 | 100 | 100 | 25 | [10000,20000] |
| 6-5000-100000 | 5,000 | 100,000 | 100 | 250 | 25 | [10000,20000] |
| 7-5000-100000 | 5,000 | 100,000 | 250 | 100 | 25 | [10000,20000] |
| 8-5000-100000 | 5,000 | 100,000 | 250 | 250 | 25 | [10000,20000] |
| 9-5000-100000 | 5,000 | 100,000 | 100 | 100 | 50 | [1000,2000] |
| 10-5000-100000 | 5,000 | 100,000 | 100 | 250 | 50 | [1000,2000] |
| 11-5000-100000 | 5,000 | 100,000 | 250 | 100 | 50 | [1000,2000] |
| 12-5000-100000 | 5,000 | 100,000 | 250 | 250 | 50 | [1000,2000] |
| 13-5000-100000 | 5,000 | 100,000 | 100 | 100 | 50 | [10000,20000] |
| 14-5000-100000 | 5,000 | 100,000 | 100 | 250 | 50 | [10000,20000] |
| 15-5000-100000 | 5,000 | 100,000 | 250 | 100 | 50 | [10000,20000] |
| 16-5000-100000 | 5,000 | 100,000 | 250 | 250 | 50 | [10000,20000] |

arcs to enter the basis 5 were shown to be robust and therefore are used in all test runs. The following subsections describe the statistical results for the data of 128 problems run by the FIXNET code compared to CPLEX.

Tables 3.8 - 3.11 list the codes' performances for test problems. To compare the qualities of the integer solution values obtained by FIXNET and CPLEX, we use the earlier defined metric, $R$, as: $R = z_1/z_2$, where $z_1$ and $z_2$ are the best integer upper-bound solution values obtained by FIXNET and CPLEX, respectively, per Sun et al. [104]. For example, if $R = 1.01$, then FIXNET's solution value was 1% larger than the best from CPLEX.

Table 3.5. Test Set Transshipment Problems with 10,000 nodes and 100,000 arcs

| Problem ID | Number of Nodes | Number of Arcs | Source Nodes | Sink Nodes | % FC Arcs | Fixed-Charge Range |
|---|---|---|---|---|---|---|
| 1-10000-100000 | 10,000 | 100,000 | 200 | 200 | 25 | [1000,2000] |
| 2-10000-100000 | 10,000 | 100,000 | 200 | 500 | 25 | [1000,2000] |
| 3-10000-100000 | 10,000 | 100,000 | 500 | 200 | 25 | [1000,2000] |
| 4-10000-100000 | 10,000 | 100,000 | 500 | 500 | 25 | [1000,2000] |
| 5-10000-100000 | 10,000 | 100,000 | 200 | 200 | 25 | [10000,20000] |
| 6-10000-100000 | 10,000 | 100,000 | 200 | 500 | 25 | [10000,20000] |
| 7-10000-100000 | 10,000 | 100,000 | 500 | 200 | 25 | [10000,20000] |
| 8-10000-100000 | 10,000 | 100,000 | 500 | 500 | 25 | [10000,20000] |
| 9-10000-100000 | 10,000 | 100,000 | 200 | 200 | 50 | [1000,2000] |
| 10-10000-100000 | 10,000 | 100,000 | 200 | 500 | 50 | [1000,2000] |
| 11-10000-100000 | 10,000 | 100,000 | 500 | 200 | 50 | [1000,2000] |
| 12-10000-100000 | 10,000 | 100,000 | 500 | 500 | 50 | [1000,2000] |
| 13-10000-100000 | 10,000 | 100,000 | 200 | 200 | 50 | [10000,20000] |
| 14-10000-100000 | 10,000 | 100,000 | 200 | 500 | 50 | [10000,20000] |
| 15-10000-100000 | 10,000 | 100,000 | 500 | 200 | 50 | [10000,20000] |
| 16-10000-100000 | 10,000 | 100,000 | 500 | 500 | 50 | [10000,20000] |

Also shown is the *time multiple*, the ratio of the CPLEX solution time to that of the FIXNET. A time multiple of 2, then, indicates that CPLEX required twice the solution time as FIXNET.

The table shows the best solution for each code, $R$ metric, solution times for both codes and time multiple that shows how much faster FIXNET code is compared to CPLEX. There are 64 problems divided into four tables for 5,000–50,000, 10,000–50,000, 5,000–100,000, and 10,000–100,000 nodes and arcs networks. Each table shows 16 test problems results and provides the overall average, standard deviation, minimum, median, and maximum values for each column. The generated 64 problems were first run with the heuristic without the dynamic linearization of the objective function to be able to see the improvements of the latter approach. The summary results are provided in the Table 3.6.

The dynamic linearization of the objective function heuristic improved 41 out of 64 tested problems with $R$ decreased from 1.1478 to 1.0210 while sacrificing time on average from 0.74 seconds to 2.86 seconds, which is still faster then CPLEX solution time by a factor of 753. One major improvement was for problems $7, 8, 13, 14, 15,$ and 16 with fixed-charge range of 10,000–20,000: the heuristic without the dynamic linearization for 24 of those problems could only find solutions within a 21.06% optimality gap on average while the addition of the dynamic linearization of the objective function has reduced this optimality gap on average to 2.95%. The summary of results for the EPTS FCGN heuristic with the dynamic linearization for 64 generated transshipment problems is provided in the Table 3.7.

### 3.2.6. Statistical Analysis

Two main hypothesis were tested: one for solution quality and another one for solution time. The first hypothesis states there is no statistical difference in solution quality between the FIXNET code and CPLEX results, where $\bar{z_1}$ is the average ob-

Table 3.6. Summary Average Results for FCGN heuristic without the Dynamic Linearization for 64 transshipment networks

| Problem Size | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|
| 5,000-50,000, average for 16 problems | 1.071 | 2319.22 | 0.55 | 4563 |
| 10,000-50,000, average for 16 problems | 1.122 | 2787.51 | 0.63 | 4521 |
| 5,000-100,000, average for 16 problems | 1.067 | 2402.36 | 0.82 | 2991 |
| 10,000-100,000, average for 16 problems | 1.165 | 2711.52 | 1.03 | 2600 |
| Overall Average, 64 problems | 1.106 | 2555.15 | 0.76 | 3669 |

Table 3.7. Summary Average Results for FCGN heuristic with the Dynamic Linearization for 64 transshipment networks

| Problem Size | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|
| 5,000-50,000, average for 16 problems | 1.016 | 2319.22 | 1.78 | 1360 |
| 10,000-50,000, average for 16 problems | 1.035 | 2787.52 | 2.37 | 1236 |
| 5,000-100,000, average for 16 problems | 1.013 | 2402.36 | 2.78 | 933 |
| 10,000-100,000, average for 16 problems | 1.036 | 2711.53 | 3.83 | 742 |
| Overall Average, 64 problems | 1.025 | 2555.15 | 2.66 | 1068 |

Table 3.8. Empirical results for 5,000 nodes and 50,000 arcs transshipment network

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| 1-5000-50000 | 2,402,862 | 2,417,902 | 1.006 | 3600.00 | 1.50 | 2406.3 |
| 2-5000-50000 | 2,286,945 | 2,306,877 | 1.009 | 3600.00 | 1.62 | 2220.7 |
| 3-5000-50000 | 2,074,403 | 2,100,240 | 1.012 | 3600.00 | 1.94 | 1854.3 |
| 4-5000-50000 | 2,084,807 | 2,091,955 | 1.003 | 3600.00 | 2.49 | 1444.5 |
| 5-5000-50000 | 2,835,780 | 2,857,281 | 1.008 | 421.16 | 1.45 | 289.8 |
| 6-5000-50000 | 2,751,068 | 2,762,061 | 1.004 | 63.81 | 1.75 | 36.4 |
| 7-5000-50000 | 2,493,693 | 2,505,173 | 1.005 | 24.83 | 1.91 | 13.0 |
| 8-5000-50000 | 2,569,595 | 2,585,979 | 1.006 | 11.29 | 1.93 | 5.9 |
| 9-5000-50000 | 2,627,148 | 2,653,551 | 1.010 | 3600.00 | 1.54 | 2339.1 |
| 10-5000-50000 | 2,573,229 | 2,638,675 | 1.025 | 3600.00 | 1.53 | 2351.0 |
| 11-5000-50000 | 2,288,626 | 2,308,734 | 1.009 | 3600.00 | 1.52 | 2375.3 |
| 12-5000-50000 | 2,353,455 | 2,387,345 | 1.014 | 3600.00 | 1.94 | 1858.1 |
| 13-5000-50000 | 3,663,600 | 3,809,449 | 1.040 | 3600.00 | 1.55 | 2315.6 |
| 14-5000-50000 | 3,731,012 | 3,904,555 | 1.047 | 3600.00 | 1.84 | 1956.7 |
| 15-5000-50000 | 3,375,812 | 3,502,397 | 1.037 | 212.09 | 1.82 | 116.8 |
| 16-5000-50000 | 3,415,538 | 3,508,242 | 1.027 | 374.30 | 2.11 | 177.4 |
| Overall Average | 2,720,473 | 2,771,276 | 1.016 | 2319.22 | 1.78 | 1360.05 |
| Overall st.dev. | 539,974 | 588,643 | 0.014 | 1710.89 | 0.28 | 1034.90 |
| Minimum | 2,074,403 | 2,091,955 | 1.003 | 11.29 | 1.45 | 5.86 |
| Median | 2,571,412 | 2,612,327 | 1.009 | 3600.00 | 1.79 | 1856.19 |
| Maximum | 3,731,012 | 3,904,555 | 1.047 | 3600.00 | 2.49 | 2406.27 |

Table 3.9. Empirical results for 10,000 nodes and 50,000 arcs transshipment network

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| 1-10000-50000 | 4,349,744 | 4,378,779 | 1.007 | 3600.00 | 1.98 | 1821.3 |
| 2-10000-50000 | 4,490,927 | 4,595,162 | 1.023 | 3600.00 | 1.93 | 1869.4 |
| 3-10000-50000 | 3,976,488 | 4,037,253 | 1.015 | 3600.00 | 1.89 | 1904.1 |
| 4-10000-50000 | 3,984,397 | 4,047,308 | 1.016 | 3600.00 | 2.22 | 1622.5 |
| 5-10000-50000 | 5,180,831 | 5,235,711 | 1.011 | 107.82 | 1.97 | 54.7 |
| 6-10000-50000 | 5,015,553 | 5,106,374 | 1.018 | 1187.09 | 2.56 | 464.0 |
| 7-10000-50000 | 4,513,842 | 4,561,029 | 1.010 | 33.66 | 2.75 | 12.2 |
| 8-10000-50000 | 4,797,182 | 4,870,013 | 1.015 | 71.70 | 2.84 | 25.2 |
| 9-10000-50000 | 4,833,268 | 4,916,706 | 1.017 | 3600.00 | 2.07 | 1735.6 |
| 10-10000-50000 | 5,000,547 | 5,182,411 | 1.036 | 3600.00 | 2.06 | 1745.5 |
| 11-10000-50000 | 4,157,376 | 4,301,084 | 1.035 | 3600.00 | 2.16 | 1669.6 |
| 12-10000-50000 | 4,536,323 | 4,822,248 | 1.063 | 3600.00 | 2.13 | 1694.1 |
| 13-10000-50000 | 6,995,251 | 7,318,128 | 1.046 | 3600.00 | 2.40 | 1501.0 |
| 14-10000-50000 | 7,758,786 | 8,377,551 | 1.080 | 3600.00 | 2.75 | 1311.0 |
| 15-10000-50000 | 6,671,524 | 7,163,171 | 1.074 | 3600.00 | 2.94 | 1223.9 |
| 16-10000-50000 | 7,432,854 | 8,077,346 | 1.087 | 3600.00 | 3.21 | 1121.2 |
| Overall Average | 5,230,931 | 5,436,892 | 1.035 | 2787.52 | 2.37 | 1235.95 |
| Overall st.dev. | 1,251,018 | 1,439,861 | 0.027 | 1474.74 | 0.42 | 697.18 |
| Minimum | 3,976,488 | 4,037,253 | 1.007 | 33.66 | 1.89 | 12.24 |
| Median | 4,815,225 | 4,893,360 | 1.021 | 3600.00 | 2.19 | 1561.76 |
| Maximum | 7,758,786 | 8,377,551 | 1.087 | 3600.00 | 3.21 | 1904.14 |

Table 3.10. Empirical results for 5,000 nodes and 100,000 arcs transshipment network

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| 1-5000-100000 | 1,341,449 | 1,351,240 | 1.007 | 3600.00 | 2.11 | 1706.7 |
| 2-5000-100000 | 1,403,777 | 1,409,230 | 1.004 | 3600.00 | 2.33 | 1543.7 |
| 3-5000-100000 | 1,286,840 | 1,294,613 | 1.006 | 3600.00 | 2.28 | 1578.1 |
| 4-5000-100000 | 1,271,657 | 1,281,088 | 1.007 | 3600.00 | 2.87 | 1255.6 |
| 5-5000-100000 | 1,661,398 | 1,668,129 | 1.004 | 32.85 | 2.64 | 12.4 |
| 6-5000-100000 | 1,595,888 | 1,599,188 | 1.002 | 15.61 | 2.73 | 5.7 |
| 7-5000-100000 | 1,425,020 | 1,429,909 | 1.003 | 13.46 | 2.67 | 5.0 |
| 8-5000-100000 | 1,342,179 | 1,345,508 | 1.002 | 13.25 | 3.96 | 3.3 |
| 9-5000-100000 | 1,481,727 | 1,521,095 | 1.027 | 3600.00 | 2.87 | 1253.9 |
| 10-5000-100000 | 1,553,536 | 1,595,726 | 1.027 | 3600.00 | 2.41 | 1491.3 |
| 11-5000-100000 | 1,380,995 | 1,425,067 | 1.032 | 3600.00 | 2.43 | 1479.3 |
| 12-5000-100000 | 1,444,148 | 1,477,136 | 1.023 | 3600.00 | 2.87 | 1253.9 |
| 13-5000-100000 | 2,231,972 | 2,278,722 | 1.021 | 3600.00 | 2.56 | 1404.9 |
| 14-5000-100000 | 2,082,997 | 2,100,067 | 1.008 | 3600.00 | 2.95 | 1220.7 |
| 15-5000-100000 | 1,867,499 | 1,908,741 | 1.022 | 1679.39 | 3.28 | 511.8 |
| 16-5000-100000 | 1,974,116 | 2,003,720 | 1.015 | 683.13 | 3.51 | 194.7 |
| Overall Average | 1,584,075 | 1,605,574 | 1.013 | 2402.36 | 2.78 | 932.56 |
| Overall st.dev. | 298,959 | 306,819 | 0.010 | 1644.14 | 0.48 | 671.73 |
| Minimum | 1,271,657 | 1,281,088 | 1.002 | 13.25 | 2.11 | 3.34 |
| Median | 1,462,937 | 1,499,115 | 1.008 | 3600.00 | 2.70 | 1253.88 |
| Maximum | 2,231,972 | 2,278,722 | 1.032 | 3600.00 | 3.96 | 1706.66 |

Table 3.11. Empirical results for 10,000 nodes and 100,000 arcs transshipment network

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| 1-10000-100000 | 2,499,089 | 2,509,067 | 1.004 | 3600.00 | 3.12 | 1153.4 |
| 2-10000-100000 | 2,419,839 | 2,552,253 | 1.055 | 3600.00 | 2.83 | 1271.2 |
| 3-10000-100000 | 2,208,953 | 2,262,214 | 1.024 | 3600.00 | 3.29 | 1093.2 |
| 4-10000-100000 | 2,183,920 | 2,297,404 | 1.052 | 3600.00 | 3.38 | 1066.7 |
| 5-10000-100000 | 2,827,207 | 2,857,851 | 1.011 | 59.10 | 3.71 | 15.9 |
| 6-10000-100000 | 2,798,864 | 2,801,327 | 1.001 | 35.17 | 3.84 | 9.2 |
| 7-10000-100000 | 2,497,241 | 2,499,202 | 1.001 | 50.50 | 3.62 | 14.0 |
| 8-10000-100000 | 2,448,628 | 2,456,658 | 1.003 | 39.66 | 4.30 | 9.2 |
| 9-10000-100000 | 2,899,632 | 2,989,534 | 1.031 | 3600.00 | 3.23 | 1113.0 |
| 10-10000-100000 | 2,802,874 | 3,034,923 | 1.083 | 3600.00 | 3.30 | 1089.4 |
| 11-10000-100000 | 2,521,131 | 2,641,567 | 1.048 | 3600.00 | 3.27 | 1102.4 |
| 12-10000-100000 | 2,666,660 | 2,950,758 | 1.107 | 3600.00 | 3.50 | 1027.4 |
| 13-10000-100000 | 3,863,185 | 4,019,042 | 1.040 | 3600.00 | 4.44 | 810.6 |
| 14-10000-100000 | 3,851,366 | 3,956,674 | 1.027 | 3600.00 | 5.37 | 670.7 |
| 15-10000-100000 | 3,604,918 | 3,740,119 | 1.038 | 3600.00 | 5.20 | 692.4 |
| 16-10000-100000 | 3,668,229 | 3,857,794 | 1.052 | 3600.00 | 4.88 | 737.3 |
| Overall Average | 2,860,109 | 2,964,149 | 1.036 | 2711.53 | 3.83 | 742.25 |
| Overall st.dev. | 569,500 | 601,891 | 0.030 | 1589.36 | 0.78 | 468.42 |
| Minimum | 2,183,920 | 2,262,214 | 1.001 | 35.17 | 2.83 | 9.17 |
| Median | 2,732,762 | 2,829,589 | 1.034 | 3600.00 | 3.56 | 918.99 |
| Maximum | 3,863,185 | 4,019,042 | 1.107 | 3600.00 | 5.37 | 1271.17 |

jective value for 64 test problems solved by CPLEX and $\bar{z}_2$ is the average objective value for the same 64 test problems solved by the FIXNET code.

$$H_0 : \bar{z}_1 = \bar{z}_2 \tag{3.9}$$

$$H_1 : \bar{z}_1 \neq \bar{z}_2 \tag{3.10}$$

The second hypothesis states there is a significant difference in solution time with the FIXNET code obtaining results faster than CPLEX with $\bar{T}_1$ and $\bar{T}_2$ being average solution times respectively for CPLEX and FIXNET solvers.

$$H_0 : \bar{T}_1 = \bar{T}_2 \tag{3.11}$$

$$H_1 : \bar{T}_1 \neq \bar{T}_2 \tag{3.12}$$

The level of significance of $\alpha = 5\%$ was used for all statistical tests. The following paragraphs describe the statistical results for the test data of 64 transshipment problems run separately by the FIXNET and CPLEX code types.

#### *3.2.6.1. Analysis of Quality of Solution due to Code Type*

To support the first hypothesis, an analysis of variance was performed to determine factors affecting the solution quality. The factors considered were the code type, the total number of nodes and arcs, number of sources, number of sinks, percent of fixed-charge arcs, and fixed-charge type. It is expected that solution quality may vary due to the different problem sizes, nodes and arcs, percent of fixed-charge arcs, and fixed-charge types. One important factor for this analysis is to determine if the solution quality difference is due to the code type or any interactions with code type. A boxplot of objective values found by CPLEX and FIXNET codes is shown in Figure 3.2. This figure includes all 128 test problems and shows that the solutions found by the FIXNET are comparable to those found by CPLEX.

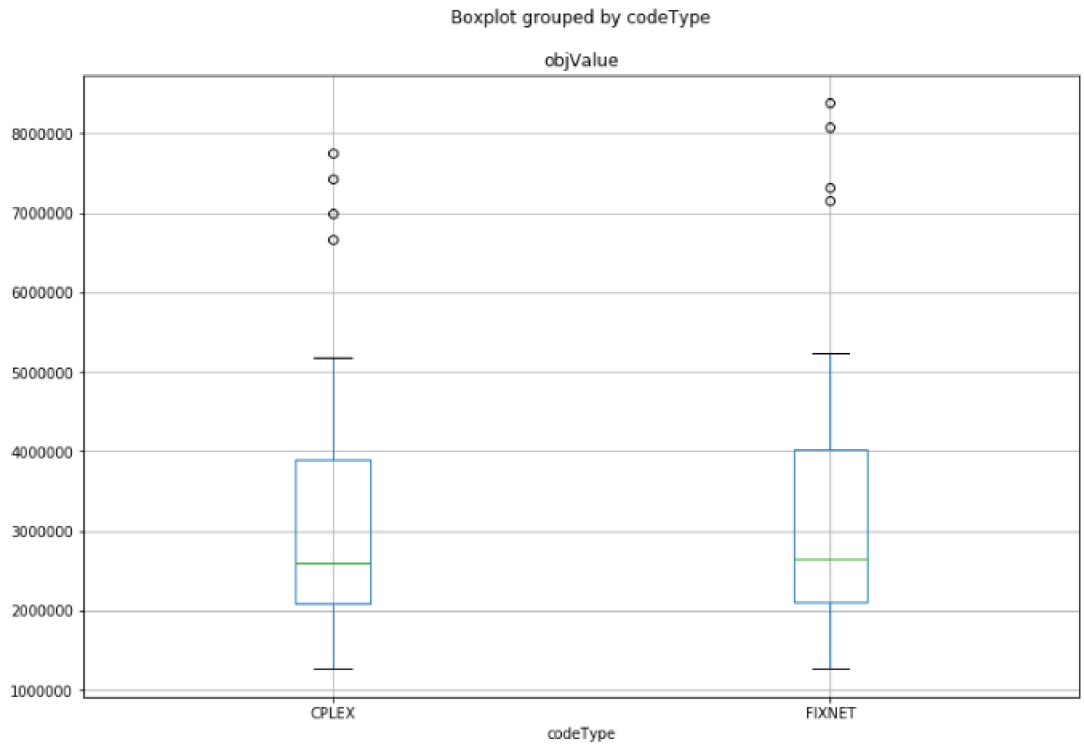



Figure 3.2. Box plot of objective value obtained by CPLEX and FIXNET codes

All 128 observations were used for analysis using the ANOVA procedure with Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] Jupyter notebook, server version 5.7.4. with significance level of 5%. The results show that there is no statistically significant difference between CPLEX and FIXNET solution values with $F(1, 126) = 0.117$ and $p = 0.7327$ and descriptive statistics as shown in Figure 3.3.

| codeType | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|
| CPLEX | 64 | 3.098897e+06 | 1.525979e+06 | 190747.341877 | 2.722077e+06 | 3.475717e+06 |
| FIXNET | 64 | 3.194473e+06 | 1.631557e+06 | 203944.680582 | 2.791581e+06 | 3.597364e+06 |

Figure 3.3. Solution Value Descriptive Statistics by Code Type

A Tukey post-hoc testing shows both codes are in the same group. While code type and number of sinks are not statistically significant factors for the solution quality, the overall model shows there is a statistically significant difference between solution values for problems with different number of nodes, arcs, sources, percent of fixed-charge arcs, and fixed-charge types as overall model have $F(7, 120) = 175.091$ with $p = 0.0000$. The model's overall coefficient determination $R^2 = 0.91$ with number of nodes and number of arcs being the most influential factors on solution quality as shown in Figure 3.4.

| | sum_sq | mean_sq | df | F | PR(>F) | eta_sq | omega_sq |
|---|---|---|---|---|---|---|---|
| Fctype | 2.488989e+13 | 2.488989e+13 | 1.0 | 106.427819 | 2.996009e-18 | 0.096084 | 0.095095 |
| Nodes | 6.635698e+13 | 6.635698e+13 | 1.0 | 283.738867 | 2.095964e-33 | 0.256162 | 0.255029 |
| Arcs | 1.021211e+14 | 1.021211e+14 | 1.0 | 436.664291 | 8.481400e-42 | 0.394225 | 0.392967 |
| Nodes:Arcs | 1.291809e+13 | 1.291809e+13 | 1.0 | 55.237043 | 1.729979e-11 | 0.049869 | 0.048922 |
| sinks | 4.195051e+11 | 4.195051e+11 | 1.0 | 1.793781 | 1.829974e-01 | 0.001619 | 0.000716 |
| PerFCarcs | 2.160510e+13 | 2.160510e+13 | 1.0 | 92.382232 | 1.450389e-16 | 0.083404 | 0.082426 |
| sources | 2.668361e+12 | 2.668361e+12 | 1.0 | 11.409766 | 9.855778e-04 | 0.010301 | 0.009390 |
| Residual | 2.806397e+13 | 2.338664e+11 | 120.0 | NaN | NaN | NaN | NaN |

Figure 3.4. ANOVA model for Solution Value

A Tukey post-hoc testing shows problems with different number of nodes, arcs, sources, percent of fixed-charge arcs, and fixed-charge types are in different groups.

Descriptive statistics are shown in Figure 3.5, 3.6, 3.7, and 3.8 by code type and by number of nodes and arcs, number of sources, percent of fixed-charge arcs, and fixed-charge types respectively.

| codeType | Nodes | Arcs | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|---|---|
| CPLEX | 5000 | 50000 | 16 | 2.720473e+06 | 5.399737e+05 | 134993.433945 | 2.447209e+06 | 2.993738e+06 |
| | | 100000 | 16 | 1.584075e+06 | 2.989586e+05 | 74739.646435 | 1.432781e+06 | 1.735369e+06 |
| | 10000 | 50000 | 16 | 5.230931e+06 | 1.251018e+06 | 312754.440455 | 4.597828e+06 | 5.864033e+06 |
| | | 100000 | 16 | 2.860109e+06 | 5.694997e+05 | 142374.928229 | 2.571902e+06 | 3.148315e+06 |
| FIXNET | 5000 | 50000 | 16 | 2.771276e+06 | 5.886428e+05 | 147160.692931 | 2.473382e+06 | 3.069170e+06 |
| | | 100000 | 16 | 1.605574e+06 | 3.068187e+05 | 76704.664089 | 1.450302e+06 | 1.760845e+06 |
| | 10000 | 50000 | 16 | 5.436892e+06 | 1.439861e+06 | 359965.353744 | 4.708222e+06 | 6.165563e+06 |
| | | 100000 | 16 | 2.964149e+06 | 6.018905e+05 | 150472.636368 | 2.659550e+06 | 3.268748e+06 |

Figure 3.5. Solution Value Descriptive Statistics by Code Type, Number of Nodes, and Number of Arcs

| codeType | sources | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|---|
| CPLEX | 100 | 16 | 2.264024e+06 | 7.516012e+05 | 187900.307932 | 1.883662e+06 | 2.644387e+06 |
| | 200 | 16 | 4.224185e+06 | 1.571260e+06 | 392815.001993 | 3.429018e+06 | 5.019352e+06 |
| | 250 | 16 | 2.040524e+06 | 6.914836e+05 | 172870.890126 | 1.690585e+06 | 2.390463e+06 |
| | 500 | 16 | 3.866854e+06 | 1.533029e+06 | 383257.213395 | 3.091034e+06 | 4.642674e+06 |
| FIXNET | 100 | 16 | 2.304609e+06 | 7.898330e+05 | 197458.243016 | 1.904898e+06 | 2.704320e+06 |
| | 200 | 16 | 4.364468e+06 | 1.684232e+06 | 421057.946059 | 3.512130e+06 | 5.216807e+06 |
| | 250 | 16 | 2.072240e+06 | 7.160523e+05 | 179013.073434 | 1.709868e+06 | 2.434613e+06 |
| | 500 | 16 | 4.036573e+06 | 1.674079e+06 | 418519.701883 | 3.189372e+06 | 4.883774e+06 |

Figure 3.6. Solution Value Descriptive Statistics by Code Type and Number of Sources

| codeType | PerFCarcs | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|---|
| CPLEX | 25 | 32 | 2.719377e+06 | 1.185893e+06 | 209638.311296 | 2.301911e+06 | 3.136843e+06 |
| | 50 | 32 | 3.478417e+06 | 1.740297e+06 | 307643.888031 | 2.865786e+06 | 4.091047e+06 |
| FIXNET | 25 | 32 | 2.752312e+06 | 1.206255e+06 | 213237.694835 | 2.327678e+06 | 3.176945e+06 |
| | 50 | 32 | 3.636634e+06 | 1.884451e+06 | 333126.933726 | 2.973257e+06 | 4.300010e+06 |

Figure 3.7. Solution Value Descriptive Statistics by Code Type and Percent of Fixed-charge Arcs

| codeType | Fctype | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|---|
| CPLEX | [1000,2000] | 32 | 2.668337e+06 | 1.139373e+06 | 201414.654336 | 2.267247e+06 | 3.069426e+06 |
| | [10000,20000] | 32 | 3.529457e+06 | 1.746838e+06 | 308800.229232 | 2.914524e+06 | 4.144390e+06 |
| FIXNET | [1000,2000] | 32 | 2.743098e+06 | 1.180431e+06 | 208672.697211 | 2.327555e+06 | 3.158641e+06 |
| | [10000,20000] | 32 | 3.645847e+06 | 1.896257e+06 | 335214.012961 | 2.978315e+06 | 4.313380e+06 |

Figure 3.8. Solution Value Descriptive Statistics by Code Type and Fixed-charge Types

#### *3.2.6.2. Analysis of Solution Time due to Code Type*

To support the second hypothesis, an analysis of variance was performed to determine factors affecting the solution times. The factors considered were the code type, the total number of nodes and arcs, number of sources, number of sinks, percent of fixed-charge arcs, and fixed-charge type. A boxplot of solution times for CPLEX and FIXNET clearly shows that FIXNET performance was faster as in Figure 3.9. One important factor for this analysis is to determine if the solution time difference is due to the code type or any interactions with code type.

All 128 observations were used for analysis using the ANOVA procedure with Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] Jupyter notebook, server version 5.7.4. with significance level of 5%. The results show that there is a statistically significant difference in the time to obtain a solution by CPLEX

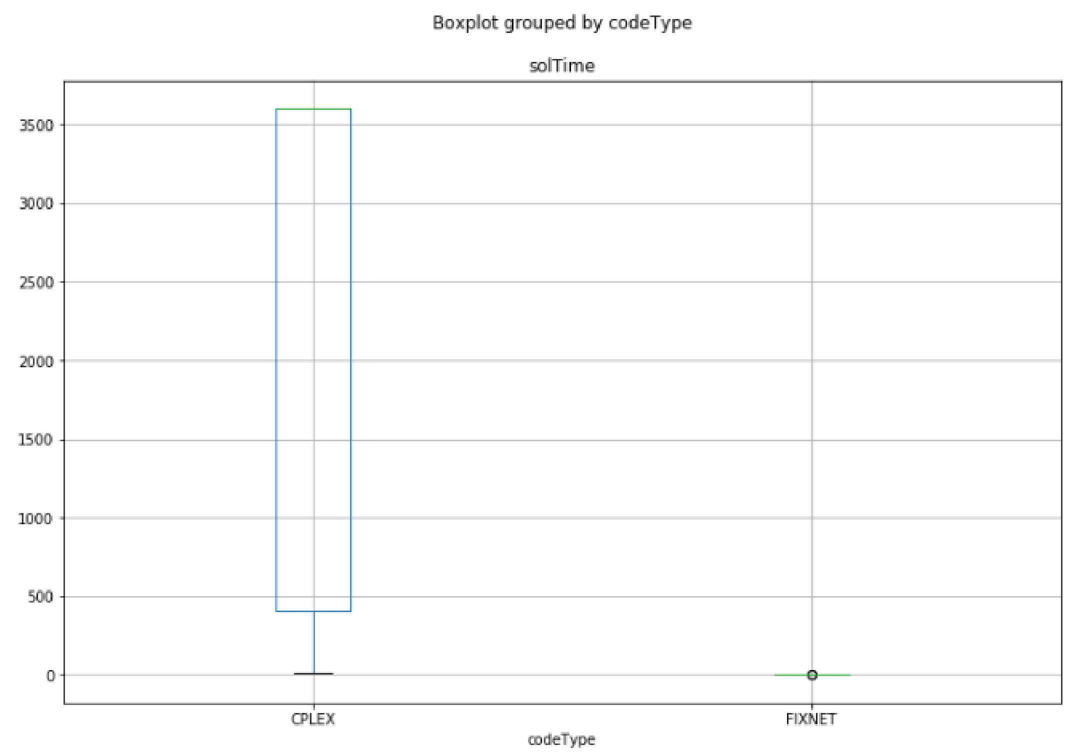



Figure 3.9. Box plot of solution times spent by CPLEX and FIXNET codes to find a solution

and by the FIXNET code with overall model $F(1, 126) = 166.799$, $p = 0.0000$ and descriptive statistics as shown in Figure 3.10. A Tukey test shows that the FIXNET code performs faster than CPLEX in finding a solution.

| codeType | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|
| CPLEX | 64 | 2555.153763 | 1581.078614 | 197.634827 | 2164.727286 | 2945.580239 |
| FIXNET | 64 | 2.688477 | 0.910898 | 0.113862 | 2.463542 | 2.913411 |

Figure 3.10. Solution Time Descriptive Statistics by Code Type

There is no statistically significant difference in the solution time due to number of nodes, number of arcs, number of sources, and number of sinks. Other factors, including percent of fixed-charge arcs and fixed-charge types and their interaction with code type, showed statistically significant effect on solution time including second order interactions of code type with percent of fixed-charge arcs and fixed-charge types as shown in Figure 3.11. The model's overall coefficient of determination $R^2 = 0.926$ with $F(7, 120) = 213.451$, $p = 0.0000$.

| | sum_sq | mean_sq | df | F | PR(>F) | eta_sq | omega_sq |
|---|---|---|---|---|---|---|---|
| codeType | 2.084825e+08 | 2.084825e+08 | 1.0 | 919.536184 | 4.247318e-58 | 0.569670 | 0.568698 |
| Fctype | 3.491484e+07 | 3.491484e+07 | 1.0 | 153.995916 | 2.951615e-23 | 0.095403 | 0.094725 |
| codeType:Fctype | 3.495420e+07 | 3.495420e+07 | 1.0 | 154.169524 | 2.840863e-23 | 0.095511 | 0.094833 |
| PerFCarcs | 1.510795e+07 | 1.510795e+07 | 1.0 | 66.635360 | 3.716506e-13 | 0.041282 | 0.040637 |
| codeType:PerFCarcs | 1.509801e+07 | 1.509801e+07 | 1.0 | 66.591518 | 3.770028e-13 | 0.041255 | 0.040610 |
| PerFCarcs:Fctype | 1.510649e+07 | 1.510649e+07 | 1.0 | 66.628896 | 3.724348e-13 | 0.041278 | 0.040633 |
| codeType:PerFCarcs:Fctype | 1.509948e+07 | 1.509948e+07 | 1.0 | 66.597980 | 3.762091e-13 | 0.041259 | 0.040614 |
| Residual | 2.720709e+07 | 2.267257e+05 | 120.0 | NaN | NaN | NaN | NaN |

Figure 3.11. ANOVA model for Solution Time

While code type is the factor that affects solution time the most, the next important factor is fixed-charge type. A Tukey post-hoc testing showed that there are distinctive groups for percent of fixed-charge arcs and fixed-charge types with descriptive statistics as shown on Figure 3.12.

| codeType | PerFCarcs | Fctype | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|---|---|
| CPLEX | 25 | [1000,2000] | 16 | 3600.000000 | 0.000000 | 0.000000 | 3600.000000 | 3600.000000 |
| | | [10000,20000] | 16 | 136.308737 | 296.930844 | 74.232711 | -13.959002 | 286.576477 |
| | 50 | [1000,2000] | 16 | 3600.000000 | 0.000000 | 0.000000 | 3600.000000 | 3600.000000 |
| | | [10000,20000] | 16 | 2884.306312 | 1313.634281 | 328.408570 | 2219.515643 | 3549.096982 |
| FIXNET | 25 | [1000,2000] | 16 | 2.360839 | 0.582558 | 0.145640 | 2.066024 | 2.655655 |
| | | [10000,20000] | 16 | 2.790039 | 0.873609 | 0.218402 | 2.347932 | 3.232147 |
| | 50 | [1000,2000] | 16 | 2.427489 | 0.671452 | 0.167863 | 2.087688 | 2.767291 |
| | | [10000,20000] | 16 | 3.175538 | 1.216662 | 0.304165 | 2.559822 | 3.791254 |

Figure 3.12. Solution Time Descriptive Statistics by Code Type, Percent Fixed-charge Arcs, and Fixed-charge Type

### *3.2.6.3. Analysis of Solution Quality Variation*

Based on obtained solution values from all test problems solved by CPLEX and the FIXNET code, the $R$ metric is calculated as defined earlier. The analysis is used to determine if there are any factors which affect the value of $R$. The factors considered were the total number of nodes and arcs, number of sources and sinks, percent of fixed-charge arcs, and fixed-charge range type. The analysis of solution quality showed there is no statistically significant difference between the two codes, but indicated percent of fixed-charge arcs and number of sinks affected the solution quality. The analysis of quality variation showed that there were several factors affecting the variability of the solutions with overall model $F(5, 122) = 43.245$ and $p = 0.0000$ with the coefficient of determination $R^2 = 0.642$, factors and their interactions as shown in Figure 3.13.

A Tukey post-hoc testing of solution quality variation showed that there are distinctive groups for number of nodes and percent of fixed-charge arcs with descriptive

| | sum_sq | mean_sq | df | F | PR(>F) | eta_sq | omega_sq |
|---|---|---|---|---|---|---|---|
| **Nodes** | 0.000947 | 0.000947 | 1.0 | 4.448343 | 3.697966e-02 | 0.015893 | 0.012276 |
| **sinks** | 0.004326 | 0.004326 | 1.0 | 20.322340 | 1.513014e-05 | 0.072608 | 0.068790 |
| **PerFCarcs** | 0.024803 | 0.024803 | 1.0 | 116.523179 | 1.771818e-19 | 0.416318 | 0.411276 |
| **sinks:PerFCarcs** | 0.002594 | 0.002594 | 1.0 | 12.188569 | 6.701315e-04 | 0.043548 | 0.039833 |
| **sources** | 0.000938 | 0.000938 | 1.0 | 4.407393 | 3.784460e-02 | 0.015747 | 0.012131 |
| **Residual** | 0.025969 | 0.000213 | 122.0 | NaN | NaN | NaN | NaN |

Figure 3.13. ANOVA model for $R$

statistics as shown on Figures 3.14 and 3.18. There is a statistically significant difference due to the number of nodes with the larger node problems and larger percent of fixed-charge arcs showing higher variation. Also, larger number of both sources and sinks cause higher variation. There is no group difference for fixed-charge range types. Descriptive statistics are shown in Figures 3.15, 3.16, 3.17, 3.19 respectively for number of arcs, sources, sinks, and fixed-charge types.

| Nodes | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|
| **5000** | 64 | 1.014826 | 0.012225 | 0.001528 | 1.011807 | 1.017844 |
| **10000** | 64 | 1.035262 | 0.027950 | 0.003494 | 1.028360 | 1.042163 |

Figure 3.14. Descriptive Statistics for $R$ by Number of Nodes

| Arcs | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|
| **50000** | 64 | 1.025496 | 0.022965 | 0.002871 | 1.019825 | 1.031167 |
| **100000** | 64 | 1.024591 | 0.024798 | 0.003100 | 1.018467 | 1.030715 |

Figure 3.15. Descriptive Statistics for $R$ by Number of Arcs

| sources | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|
| 100 | 32 | 1.015533 | 0.013590 | 0.002402 | 1.010749 | 1.020317 |
| 200 | 32 | 1.030627 | 0.024685 | 0.004364 | 1.021937 | 1.039317 |
| 250 | 32 | 1.014118 | 0.010863 | 0.001920 | 1.010294 | 1.017942 |
| 500 | 32 | 1.039896 | 0.030559 | 0.005402 | 1.029138 | 1.050653 |

Figure 3.16. Descriptive Statistics for $R$ by Number of Sources

| sinks | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|
| 100 | 32 | 1.015586 | 0.012245 | 0.002165 | 1.011276 | 1.019897 |
| 200 | 32 | 1.025690 | 0.019654 | 0.003474 | 1.018771 | 1.032608 |
| 250 | 32 | 1.014065 | 0.012353 | 0.002184 | 1.009717 | 1.018414 |
| 500 | 32 | 1.044834 | 0.031814 | 0.005624 | 1.033634 | 1.056033 |

Figure 3.17. Descriptive Statistics for $R$ by Number of Sinks

| PerFCarcs | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|
| 25 | 64 | 1.011123 | 0.012502 | 0.001563 | 1.008036 | 1.014211 |
| 50 | 64 | 1.038964 | 0.024347 | 0.003043 | 1.032952 | 1.044976 |

Figure 3.18. Descriptive Statistics for $R$ by Percent of Fixed-charge Arcs

| Fctype | N | Mean | SD | SE | 95% Conf. | Interval |
|---|---|---|---|---|---|---|
| [1000,2000] | 64 | 1.026179 | 0.023985 | 0.002998 | 1.020256 | 1.032102 |
| [10000,20000] | 64 | 1.023908 | 0.023767 | 0.002971 | 1.018039 | 1.029777 |

Figure 3.19. Descriptive Statistics for $R$ by Fixed-charge Type

### 3.3. Conclusions

This chapter extends the approach applied to fixed-charge generalized transportation networks to fixed-charge generalized transshipment problems. Transshipment problems of large sizes with 5,000 and 10,000 nodes, 50,000 and 100,000 arcs were generated and the FCGT extreme-point tabu search heuristic that incorporates a diversification phase was applied first. To improve the solution quality, the addition of the dynamic linearization of the objective function to the meta-heuristic EPTS FCGN is developed. The computational results showed the overall improvement in the quality of the solution compared to the heuristic without the dynamic linearization of the objective function for transshipment network problems. The meta-heuristic builds on the extreme-point tabu search approach and uses the special basis structure of the forest of quasi-trees for generalized networks. The quality of the solution obtained by the meta-heuristic with the dynamic linearization for 64 transshipment networks on average is within 2.5% of optimality reported by CPLEX. While the solution time has increased slightly compared to smaller transportation problems to 2.66 seconds on average, it was 1,000 times faster on average than the commercial state-of-the-art solver CPLEX. The proposed heuristic, testbed parameters and generated problems, and computational results fill the gap in the published research for fixed-charge generalized transshipment network problems.

## Chapter 4

## Summary of Findings and Conclusions

The fixed-charge generalized network problem has a number of real world applications that receive scant attention compared to pure classical fixed-charge transportation and transshipment problems. This dissertation presented meta-heuristics to solve the problems and provided computational comparisons with commercial solver CPLEX.

Chapter 1 is an overview of generalized networks that presented the mathematical model formulations including models with fixed charge component and interval-flow networks. Example real-world applications are also presented showing the diverse areas where generalized networks with fixed charges are useful.

Chapter 2 presents the initial heuristics for solving the capacitated generalized fixed charge transportation network problem. The heuristic employs the quasi-forest structure of a generalized network to perform a pivot in an attempt to find a basic feasible solution. Once found, the heuristic improves the solution through a tabu search approach using recency based memory and a network-based implementation of the primal simplex method as a local search method. Different candidate list strategies and tabu parameters are calibrated for the FIXNET implementation. The meta-heuristic is extended by the inclusion of the diversification phase that showed improved performance for both solution quality and solution time compared to the local search approach. An extensive computational comparison is performed to evaluate the performance on randomly generated problems of 130 nodes and 3000 arcs and 150 nodes and 5000 arcs networks and different ranges of magnitude of fixed costs relative to variable costs. Objective function values and solution time metrics are used as criteria to compare the performance of the heuristic with the state-of-the-art solver CPLEX.

Chapter 3 investigates the application of the proposed meta-heuristic to the class of fixed-charge generalized transshipment problems of larger size with number of nodes 50,000 or 10,000 and number of arcs 50,000 or 100,000. To improve the solution quality the meta-heuristic is expanded to investigate the performance of the dynamic linearization of the objective function process for solving FCGN. The robust parameters for candidate list strategies and tabu are used for testbed generalized problems. Experimental design is conducted and best practices for FCGN is developed.

In summary, the main contributions of this research are (1) the development of a heuristic approach for the fixed-charge generalized transportation problems (FCGT), (2) the development of a heuristic solution method with dynamic linearization of objective function for the fixed-charge generalized transshipment problems (FCGN) of larger sizes with up to 10,000 nodes and 100,000 arcs, (3) presenting new test problem sets for both FCGT and FCGN, and (4) providing computational testing of codes for FCGT and FCGN heuristics to demonstrate the effectiveness of each in terms of speed and quality of solutions.

Chapter 5

# APPENDIX A

## 5.1. FCGT: Empirical Results Summary by Groups

### 5.1.1. Summary by Fixed-charge Range Types

### 5.1.2. Summary by Multiplier Range Types

### 5.1.3. Summary by Number of Nodes and Arcs

### 5.1.4. Summary by Difficulty of the Problem based on FC Ranges

Table 5.1. Empirical results for [50,200] fixed-charge range

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest1-130-3000-A | 92,417 | 92,877 | 1.005 | 37.50 | 0.02 | 2401.0 |
| fcgnTest2-130-3000-A | 95,749 | 96,571 | 1.009 | 12.54 | 0.02 | 535.2 |
| fcgnTest3-130-3000-A | 188,884 | 188,995 | 1.001 | 253.84 | 0.03 | 8123.0 |
| fcgnTest4-130-3000-A | 186,597 | 186,753 | 1.001 | 54.12 | 0.02 | 3464.9 |
| fcgnTest5-130-3000-A | 83,704 | 84,204 | 1.006 | 28.76 | 0.01 | 3682.4 |
| fcgnTest6-130-3000-A | 84,721 | 85,036 | 1.004 | 21.30 | 0.01 | 2727.6 |
| fcgnTest7-130-3000-A | 111,283 | 111,817 | 1.005 | 3600.00 | 0.02 | 230473.8 |
| fcgnTest8-130-3000-A | 112,833 | 113,199 | 1.003 | 3600.00 | 0.04 | 92165.9 |
| fcgnTest1-150-5000-A | 141,444 | 141,911 | 1.003 | 64.98 | 0.02 | 2772.3 |
| fcgnTest2-150-5000-A | 145,634 | 146,140 | 1.003 | 275.00 | 0.02 | 11732.1 |
| fcgnTest3-150-5000-A | 300,692 | 300,904 | 1.001 | 201.66 | 0.05 | 3687.3 |
| fcgnTest4-150-5000-A | 300,199 | 300,457 | 1.001 | 287.78 | 0.05 | 6138.7 |
| fcgnTest5-150-5000-A | 133,532 | 133,972 | 1.003 | 357.31 | 0.02 | 15243.5 |
| fcgnTest6-150-5000-A | 133,556 | 134,384 | 1.006 | 33.48 | 0.02 | 2143.3 |
| fcgnTest7-150-5000-A | 174,885 | 175,606 | 1.004 | 3600.00 | 0.06 | 57600.0 |
| fcgnTest8-150-5000-A | 178,830 | 179,353 | 1.003 | 3600.00 | 0.06 | 57600.0 |
| Overall average | 154,060 | 154,511 | 1.004 | 1001.77 | 0.03 | 31280.7 |
| Overall st.dev. | 67,199 | 67,098 | 0.002 | 1553.18 | 0.02 | 59515.3 |
| Minimum | 83,704 | 84,204 | 1.001 | 12.54 | 0.01 | 535.2 |
| Median | 137,500 | 138,148 | 1.003 | 227.75 | 0.02 | 4913.0 |
| Maximum | 300,692 | 300,904 | 1.009 | 3600.00 | 0.06 | 230473.8 |

Table 5.2. Empirical results for [100,400] fixed-charge range

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest1-130-3000-B | 102,556 | 103,715 | 1.011 | 25.21 | 0.04 | 645.4 |
| fcgnTest2-130-3000-B | 101,644 | 103,133 | 1.015 | 2.39 | 0.02 | 102.0 |
| fcgnTest3-130-3000-B | 205,646 | 205,825 | 1.001 | 297.65 | 0.02 | 19055.6 |
| fcgnTest4-130-3000-B | 211,636 | 211,781 | 1.001 | 3600.00 | 0.02 | 153583.6 |
| fcgnTest5-130-3000-B | 94,220 | 94,624 | 1.004 | 8.55 | 0.01 | 1095.1 |
| fcgnTest6-130-3000-B | 95,727 | 96,607 | 1.009 | 6.27 | 0.02 | 401.6 |
| fcgnTest7-130-3000-B | 125,433 | 126,267 | 1.007 | 3600.00 | 0.04 | 92165.9 |
| fcgnTest8-130-3000-B | 122,542 | 123,898 | 1.011 | 3600.00 | 0.02 | 153583.6 |
| fcgnTest1-150-5000-B | 153,395 | 154,173 | 1.005 | 39.44 | 0.02 | 2525.2 |
| fcgnTest2-150-5000-B | 152,547 | 154,688 | 1.014 | 78.31 | 0.03 | 2506.0 |
| fcgnTest3-150-5000-B | 307,406 | 307,609 | 1.001 | 3600.00 | 0.02 | 230473.8 |
| fcgnTest4-150-5000-B | 326,901 | 327,121 | 1.001 | 71.67 | 0.03 | 2293.4 |
| fcgnTest5-150-5000-B | 150,938 | 152,665 | 1.011 | 4.09 | 0.03 | 131.0 |
| fcgnTest6-150-5000-B | 144,183 | 145,750 | 1.011 | 1174.57 | 0.04 | 30070.9 |
| fcgnTest7-150-5000-B | 189,422 | 190,846 | 1.008 | 3600.00 | 0.03 | 115200.0 |
| fcgnTest8-150-5000-B | 188,771 | 189,818 | 1.006 | 3600.00 | 0.05 | 65825.6 |
| Overall average | 167,060 | 168,032 | 1.007 | 1456.76 | 0.03 | 54353.7 |
| Overall st.dev. | 69,963 | 69,648 | 0.005 | 1737.57 | 0.01 | 73216.8 |
| Minimum | 94,220 | 94,624 | 1.001 | 2.39 | 0.01 | 102.0 |
| Median | 151,743 | 153,419 | 1.007 | 187.98 | 0.03 | 10790.4 |
| Maximum | 326,901 | 327,121 | 1.015 | 3600.00 | 0.05 | 230473.8 |

Table 5.3. Empirical results for [200,800] fixed-charge range

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest1-130-3000-C | 115,185 | 117,253 | 1.018 | 8.53 | 0.02 | 546.06 |
| fcgnTest2-130-3000-C | 123,007 | 126,032 | 1.025 | 62.80 | 0.02 | 4020.30 |
| fcgnTest3-130-3000-C | 243,962 | 244,551 | 1.002 | 9.01 | 0.02 | 576.79 |
| fcgnTest4-130-3000-C | 230,272 | 231,465 | 1.005 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest5-130-3000-C | 110,144 | 111,383 | 1.011 | 0.74 | 0.01 | 94.34 |
| fcgnTest6-130-3000-C | 112,186 | 113,110 | 1.008 | 12.13 | 0.02 | 776.25 |
| fcgnTest7-130-3000-C | 144,935 | 145,717 | 1.005 | 3600.00 | 0.06 | 57600.00 |
| fcgnTest8-130-3000-C | 143,237 | 144,141 | 1.006 | 3600.00 | 0.05 | 76791.81 |
| fcgnTest1-150-5000-C | 165,237 | 167,270 | 1.012 | 56.56 | 0.02 | 2413.05 |
| fcgnTest2-150-5000-C | 171,869 | 173,661 | 1.010 | 64.73 | 0.04 | 1657.12 |
| fcgnTest3-150-5000-C | 335,554 | 333,668 | 0.994 | 3600.00 | 0.03 | 115200.00 |
| fcgnTest4-150-5000-C | 362,259 | 363,080 | 1.002 | 3600.00 | 0.02 | 153583.62 |
| fcgnTest5-150-5000-C | 157,178 | 159,417 | 1.014 | 41.34 | 0.03 | 1322.91 |
| fcgnTest6-150-5000-C | 159,408 | 163,524 | 1.026 | 1276.01 | 0.02 | 54437.29 |
| fcgnTest7-150-5000-C | 219,006 | 220,431 | 1.007 | 3600.00 | 0.08 | 46082.95 |
| fcgnTest8-150-5000-C | 216,078 | 218,218 | 1.010 | 3600.00 | 0.06 | 57600.00 |
| Overall average | 188,095 | 189,558 | 1.010 | 1670.74 | 0.03 | 50198.51 |
| Overall st.dev. | 75,959 | 75,265 | 0.008 | 1783.25 | 0.02 | 66985.64 |
| Minimum | 110,144 | 111,383 | 0.994 | 0.74 | 0.01 | 94.34 |
| Median | 162,322 | 165,397 | 1.009 | 670.37 | 0.02 | 25051.63 |
| Maximum | 362,259 | 363,080 | 1.026 | 3600.00 | 0.08 | 230473.75 |

Table 5.4. Empirical results for [400,1600] fixed-charge range

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest1-130-3000-D | 146,472 | 148,290 | 1.012 | 2.24 | 0.01 | 286.85 |
| fcgnTest2-130-3000-D | 152,589 | 156,552 | 1.026 | 16.69 | 0.02 | 1068.69 |
| fcgnTest3-130-3000-D | 297,458 | 298,834 | 1.005 | 1781.74 | 0.01 | 228135.72 |
| fcgnTest4-130-3000-D | 280,474 | 282,164 | 1.006 | 1322.65 | 0.02 | 56427.05 |
| fcgnTest5-130-3000-D | 138,347 | 142,631 | 1.031 | 46.28 | 0.02 | 2962.86 |
| fcgnTest6-130-3000-D | 141,452 | 145,325 | 1.027 | 32.91 | 0.02 | 2107.00 |
| fcgnTest7-130-3000-D | 179,764 | 182,204 | 1.014 | 3600.00 | 0.09 | 41889.69 |
| fcgnTest8-130-3000-D | 179,973 | 179,984 | 1.000 | 3600.00 | 0.05 | 76791.81 |
| fcgnTest1-150-5000-D | 202,641 | 207,865 | 1.026 | 10.65 | 0.02 | 454.19 |
| fcgnTest2-150-5000-D | 200,605 | 205,896 | 1.026 | 476.10 | 0.03 | 15235.17 |
| fcgnTest3-150-5000-D | 404,886 | 406,103 | 1.003 | 3600.00 | 0.08 | 46082.95 |
| fcgnTest4-150-5000-D | 395,566 | 398,868 | 1.008 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest5-150-5000-D | 189,913 | 196,037 | 1.032 | 619.74 | 0.02 | 39676.06 |
| fcgnTest6-150-5000-D | 191,292 | 196,155 | 1.025 | 6.72 | 0.02 | 430.29 |
| fcgnTest7-150-5000-D | 257,661 | 259,674 | 1.008 | 3600.00 | 0.09 | 41889.69 |
| fcgnTest8-150-5000-D | 259,043 | 261,126 | 1.008 | 3600.00 | 0.06 | 57600.00 |
| Overall average | 226,134 | 229,232 | 1.016 | 1619.73 | 0.03 | 52594.49 |
| Overall st.dev. | 83,833 | 83,210 | 0.011 | 1659.76 | 0.03 | 73374.93 |
| Minimum | 138,347 | 142,631 | 1.000 | 2.24 | 0.01 | 286.85 |
| Median | 195,949 | 201,025 | 1.013 | 971.20 | 0.02 | 40782.87 |
| Maximum | 404,886 | 406,103 | 1.032 | 3600.00 | 0.09 | 230473.75 |

Table 5.5. Empirical results for [800,3200] fixed-charge range

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest1-130-3000-E | 208,957 | 216,289 | 1.035 | 8.47 | 0.02 | 542.38 |
| fcgnTest2-130-3000-E | 211,423 | 220,774 | 1.044 | 50.78 | 0.02 | 3250.98 |
| fcgnTest3-130-3000-E | 407,548 | 411,877 | 1.011 | 457.70 | 0.02 | 29301.92 |
| fcgnTest4-130-3000-E | 406,078 | 408,982 | 1.007 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest5-130-3000-E | 192,723 | 200,963 | 1.043 | 4.19 | 0.02 | 268.56 |
| fcgnTest6-130-3000-E | 190,802 | 195,845 | 1.026 | 4.94 | 0.02 | 316.00 |
| fcgnTest7-130-3000-E | 238,706 | 240,583 | 1.008 | 3600.00 | 0.05 | 65825.56 |
| fcgnTest8-130-3000-E | 246,057 | 247,741 | 1.007 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest1-150-5000-E | 269,725 | 277,052 | 1.027 | 770.73 | 0.05 | 16440.51 |
| fcgnTest2-150-5000-E | 271,000 | 290,289 | 1.071 | 593.82 | 0.02 | 25333.70 |
| fcgnTest3-150-5000-E | 542,330 | 551,278 | 1.016 | 3600.00 | 0.05 | 76791.81 |
| fcgnTest4-150-5000-E | 533,002 | 539,940 | 1.013 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest5-150-5000-E | 253,274 | 260,501 | 1.029 | 79.16 | 0.02 | 5067.60 |
| fcgnTest6-150-5000-E | 246,603 | 257,087 | 1.043 | 151.31 | 0.02 | 9687.00 |
| fcgnTest7-150-5000-E | 340,455 | 351,475 | 1.032 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest8-150-5000-E | 340,987 | 351,006 | 1.029 | 3600.00 | 0.02 | 230473.75 |
| Overall average | 306,229 | 313,855 | 1.028 | 1707.57 | 0.03 | 69286.19 |
| Overall st.dev. | 113,006 | 112,993 | 0.017 | 1736.96 | 0.01 | 86280.77 |
| Minimum | 190,802 | 195,845 | 1.007 | 4.19 | 0.02 | 268.56 |
| Median | 261,500 | 268,777 | 1.028 | 682.28 | 0.02 | 27317.81 |
| Maximum | 542,330 | 551,278 | 1.071 | 3600.00 | 0.05 | 230473.75 |

Table 5.6. Empirical results for [1600,6400] fixed-charge range

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest1-130-3000-F | 329,878 | 350,065 | 1.061 | 44.87 | 0.04 | 1148.80 |
| fcgnTest2-130-3000-F | 309,944 | 330,598 | 1.067 | 4.18 | 0.02 | 178.51 |
| fcgnTest3-130-3000-F | 613,303 | 622,369 | 1.015 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest4-130-3000-F | 671,863 | 678,027 | 1.009 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest5-130-3000-F | 292,146 | 310,160 | 1.062 | 55.76 | 0.02 | 3569.48 |
| fcgnTest6-130-3000-F | 294,321 | 307,546 | 1.045 | 2.12 | 0.02 | 136.02 |
| fcgnTest7-130-3000-F | 370,447 | 373,127 | 1.007 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest8-130-3000-F | 378,477 | 380,110 | 1.004 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest1-150-5000-F | 377,150 | 393,364 | 1.043 | 43.35 | 0.03 | 1387.18 |
| fcgnTest2-150-5000-F | 375,329 | 396,433 | 1.056 | 80.06 | 0.02 | 3415.38 |
| fcgnTest3-150-5000-F | 793,989 | 808,525 | 1.018 | 23.48 | 0.03 | 751.34 |
| fcgnTest4-150-5000-F | 809,654 | 816,569 | 1.009 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest5-150-5000-F | 353,658 | 378,625 | 1.071 | 6.18 | 0.02 | 263.81 |
| fcgnTest6-150-5000-F | 355,263 | 366,340 | 1.031 | 14.47 | 0.02 | 617.21 |
| fcgnTest7-150-5000-F | 492,446 | 499,206 | 1.014 | 3600.00 | 0.07 | 51201.82 |
| fcgnTest8-150-5000-F | 482,548 | 488,722 | 1.013 | 3600.00 | 0.03 | 115200.00 |
| Overall average | 456,276 | 468,737 | 1.033 | 1592.15 | 0.03 | 57207.17 |
| Overall st.dev. | 173,125 | 170,486 | 0.024 | 1828.93 | 0.01 | 79698.92 |
| Minimum | 292,146 | 307,546 | 1.004 | 2.12 | 0.02 | 136.02 |
| Median | 376,239 | 386,737 | 1.025 | 67.91 | 0.03 | 3492.43 |
| Maximum | 809,654 | 816,569 | 1.071 | 3600.00 | 0.07 | 230473.75 |

Table 5.7. Empirical results for [3200,12800] fixed-charge range

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest1-130-3000-G | 525,997 | 561,945 | 1.068 | 478.91 | 0.02 | 20431.23 |
| fcgnTest2-130-3000-G | 515,732 | 562,115 | 1.090 | 21.80 | 0.04 | 558.19 |
| fcgnTest3-130-3000-G | 1,069,340 | 1,085,471 | 1.015 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest4-130-3000-G | 1,095,638 | 1,107,427 | 1.011 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest5-130-3000-G | 498,590 | 519,061 | 1.041 | 27.01 | 0.01 | 3458.14 |
| fcgnTest6-130-3000-G | 492,861 | 512,241 | 1.039 | 0.91 | 0.01 | 117.12 |
| fcgnTest7-130-3000-G | 624,195 | 625,540 | 1.002 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest8-130-3000-G | 631,238 | 638,869 | 1.012 | 3600.00 | 0.05 | 65825.56 |
| fcgnTest1-150-5000-G | 591,978 | 627,149 | 1.059 | 1795.29 | 0.04 | 45962.37 |
| fcgnTest2-150-5000-G | 577,676 | 621,744 | 1.076 | 60.42 | 0.02 | 3867.90 |
| fcgnTest3-150-5000-G | 1,305,463 | 1,314,328 | 1.007 | 3600.00 | 0.03 | 115200.00 |
| fcgnTest4-150-5000-G | 1,268,954 | 1,295,758 | 1.021 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest5-150-5000-G | 569,891 | 603,294 | 1.059 | 136.60 | 0.02 | 5827.82 |
| fcgnTest6-150-5000-G | 551,199 | 593,668 | 1.077 | 124.65 | 0.02 | 7979.90 |
| fcgnTest7-150-5000-G | 781,501 | 788,399 | 1.009 | 3600.00 | 0.07 | 51201.82 |
| fcgnTest8-150-5000-G | 769,946 | 779,461 | 1.012 | 3600.00 | 0.05 | 65825.56 |
| Overall average | 741,888 | 764,779 | 1.037 | 1965.35 | 0.03 | 55826.69 |
| Overall st.dev. | 281,730 | 275,842 | 0.030 | 1739.15 | 0.02 | 61073.42 |
| Minimum | 492,861 | 512,241 | 1.002 | 0.91 | 0.01 | 117.12 |
| Median | 608,087 | 626,344 | 1.030 | 2697.65 | 0.04 | 48582.09 |
| Maximum | 1,305,463 | 1,314,328 | 1.090 | 3600.00 | 0.07 | 230473.75 |

Table 5.8. Empirical results for [6400,25600] fixed-charge range

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest1-130-3000-H | 909,331 | 999,018 | 1.099 | 80.55 | 0.02 | 5157.03 |
| fcgnTest2-130-3000-H | 927,249 | 1,008,104 | 1.087 | 238.57 | 0.02 | 10177.82 |
| fcgnTest3-130-3000-H | 1,925,228 | 1,977,657 | 1.027 | 3600.00 | 0.02 | 153583.62 |
| fcgnTest4-130-3000-H | 2,033,811 | 2,049,132 | 1.008 | 3600.00 | 0.01 | 460947.50 |
| fcgnTest5-130-3000-H | 870,628 | 927,512 | 1.065 | 56.42 | 0.02 | 2406.83 |
| fcgnTest6-130-3000-H | 885,265 | 953,115 | 1.077 | 1.43 | 0.02 | 91.68 |
| fcgnTest7-130-3000-H | 1,096,563 | 1,108,508 | 1.011 | 3600.00 | 0.03 | 115200.00 |
| fcgnTest8-130-3000-H | 1,116,227 | 1,122,982 | 1.006 | 3600.00 | 0.02 | 153583.62 |
| fcgnTest1-150-5000-H | 987,862 | 1,084,330 | 1.098 | 2685.09 | 0.04 | 68742.70 |
| fcgnTest2-150-5000-H | 984,349 | 1,056,675 | 1.073 | 103.02 | 0.03 | 3296.54 |
| fcgnTest3-150-5000-H | 2,465,266 | 2,487,311 | 1.009 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest4-150-5000-H | 2,318,198 | 2,351,353 | 1.014 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest5-150-5000-H | 925,685 | 978,339 | 1.057 | 11.02 | 0.01 | 1410.99 |
| fcgnTest6-150-5000-H | 956,078 | 992,928 | 1.039 | 92.58 | 0.02 | 5926.94 |
| fcgnTest7-150-5000-H | 1,310,903 | 1,324,079 | 1.010 | 3600.00 | 0.03 | 115200.00 |
| fcgnTest8-150-5000-H | 1,342,828 | 1,347,275 | 1.003 | 3600.00 | 0.06 | 57600.00 |
| Overall average | 1,315,967 | 1,360,520 | 1.043 | 2004.29 | 0.03 | 83603.57 |
| Overall st.dev. | 547,746 | 534,542 | 0.036 | 1764.48 | 0.01 | 115347.02 |
| Minimum | 870,628 | 927,512 | 1.003 | 1.43 | 0.01 | 91.68 |
| Median | 1,042,212 | 1,096,419 | 1.033 | 3142.55 | 0.02 | 63171.35 |
| Maximum | 2,465,266 | 2,487,311 | 1.099 | 3600.00 | 0.06 | 460947.50 |

Table 5.9. Empirical results for [1.0 − 1.5] multipliers range

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest5-130-3000-A | 83,704 | 84,204 | 1.006 | 28.76 | 0.01 | 3682.36 |
| fcgnTest5-130-3000-B | 94,220 | 94,624 | 1.004 | 8.55 | 0.01 | 1095.11 |
| fcgnTest5-130-3000-C | 110,144 | 111,383 | 1.011 | 0.74 | 0.01 | 94.34 |
| fcgnTest5-130-3000-D | 138,347 | 142,631 | 1.031 | 46.28 | 0.02 | 2962.86 |
| fcgnTest5-130-3000-E | 192,723 | 200,963 | 1.043 | 4.19 | 0.02 | 268.56 |
| fcgnTest5-130-3000-F | 292,146 | 310,160 | 1.062 | 55.76 | 0.02 | 3569.48 |
| fcgnTest5-130-3000-G | 498,590 | 519,061 | 1.041 | 27.01 | 0.01 | 3458.14 |
| fcgnTest5-130-3000-H | 870,628 | 927,512 | 1.065 | 56.42 | 0.02 | 2406.83 |
| fcgnTest6-130-3000-A | 84,721 | 85,036 | 1.004 | 21.30 | 0.01 | 2727.63 |
| fcgnTest6-130-3000-B | 95,727 | 96,607 | 1.009 | 6.27 | 0.02 | 401.60 |
| fcgnTest6-130-3000-C | 112,186 | 113,110 | 1.008 | 12.13 | 0.02 | 776.25 |
| fcgnTest6-130-3000-D | 141,452 | 145,325 | 1.027 | 32.91 | 0.02 | 2107.00 |
| fcgnTest6-130-3000-E | 190,802 | 195,845 | 1.026 | 4.94 | 0.02 | 316.00 |
| fcgnTest6-130-3000-F | 294,321 | 307,546 | 1.045 | 2.12 | 0.02 | 136.02 |
| fcgnTest6-130-3000-G | 492,861 | 512,241 | 1.039 | 0.91 | 0.01 | 117.12 |
| fcgnTest6-130-3000-H | 885,265 | 953,115 | 1.077 | 1.43 | 0.02 | 91.68 |
| fcgnTest5-150-5000-A | 133,532 | 133,972 | 1.003 | 357.31 | 0.02 | 15243.47 |
| fcgnTest5-150-5000-B | 150,938 | 152,665 | 1.011 | 4.09 | 0.03 | 131.04 |
| fcgnTest5-150-5000-C | 157,178 | 159,417 | 1.014 | 41.34 | 0.03 | 1322.91 |
| fcgnTest5-150-5000-D | 189,913 | 196,037 | 1.032 | 619.74 | 0.02 | 39676.06 |
| fcgnTest5-150-5000-E | 253,274 | 260,501 | 1.029 | 79.16 | 0.02 | 5067.60 |
| fcgnTest5-150-5000-F | 353,658 | 378,625 | 1.071 | 6.18 | 0.02 | 263.81 |
| fcgnTest5-150-5000-G | 569,891 | 603,294 | 1.059 | 136.60 | 0.02 | 5827.82 |
| fcgnTest5-150-5000-H | 925,685 | 978,339 | 1.057 | 11.02 | 0.01 | 1410.99 |
| fcgnTest6-150-5000-A | 133,556 | 134,384 | 1.006 | 33.48 | 0.02 | 2143.28 |
| fcgnTest6-150-5000-B | 144,183 | 145,750 | 1.011 | 1174.57 | 0.04 | 30070.92 |
| fcgnTest6-150-5000-C | 159,408 | 163,524 | 1.026 | 1276.01 | 0.02 | 54437.29 |
| fcgnTest6-150-5000-D | 191,292 | 196,155 | 1.025 | 6.72 | 0.02 | 430.29 |
| fcgnTest6-150-5000-E | 246,603 | 257,087 | 1.043 | 151.31 | 0.02 | 9687.00 |
| fcgnTest6-150-5000-F | 355,263 | 366,340 | 1.031 | 14.47 | 0.02 | 617.21 |
| fcgnTest6-150-5000-G | 551,199 | 593,668 | 1.077 | 124.65 | 0.02 | 7979.90 |
| fcgnTest6-150-5000-H | 956,078 | 992,928 | 1.039 | 92.58 | 0.02 | 5926.94 |
| Overall average | 314,046 | 328,502 | 1.032 | 138.72 | 0.02 | 6388.98 |
| Overall st.dev. | 266,194 | 283,793 | 0.023 | 310.61 | 0.01 | 12308.13 |
| Minimum | 83,704 | 84,204 | 1.003 | 0.74 | 0.01 | 91.68 |
| Median | 191,047 | 196,096 | 1.030 | 27.88 | 0.02 | 2125.14 |
| Maximum | 956,078 | 992,928 | 1.077 | 1276.01 | 0.04 | 54437.29 |

Table 5.10. Empirical results for $[0.5 - 1.0]$ multipliers range

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest3-130-3000-A | 188,884 | 188,995 | 1.001 | 253.84 | 0.03 | 8122.98 |
| fcgnTest3-130-3000-B | 205,646 | 205,825 | 1.001 | 297.65 | 0.02 | 19055.57 |
| fcgnTest3-130-3000-C | 243,962 | 244,551 | 1.002 | 9.01 | 0.02 | 576.79 |
| fcgnTest3-130-3000-D | 297,458 | 298,834 | 1.005 | 1781.74 | 0.01 | 228135.72 |
| fcgnTest3-130-3000-E | 407,548 | 411,877 | 1.011 | 457.70 | 0.02 | 29301.92 |
| fcgnTest3-130-3000-F | 613,303 | 622,369 | 1.015 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest3-130-3000-G | 1,069,340 | 1,085,471 | 1.015 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest3-130-3000-H | 1,925,228 | 1,977,657 | 1.027 | 3600.00 | 0.02 | 153583.62 |
| fcgnTest4-130-3000-A | 186,597 | 186,753 | 1.001 | 54.12 | 0.02 | 3464.92 |
| fcgnTest4-130-3000-B | 211,636 | 211,781 | 1.001 | 3600.00 | 0.02 | 153583.62 |
| fcgnTest4-130-3000-C | 230,272 | 231,465 | 1.005 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest4-130-3000-D | 280,474 | 282,164 | 1.006 | 1322.65 | 0.02 | 56427.05 |
| fcgnTest4-130-3000-E | 406,078 | 408,982 | 1.007 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest4-130-3000-F | 671,863 | 678,027 | 1.009 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest4-130-3000-G | 1,095,638 | 1,107,427 | 1.011 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest4-130-3000-H | 2,033,811 | 2,049,132 | 1.008 | 3600.00 | 0.01 | 460947.50 |
| fcgnTest3-150-5000-A | 300,692 | 300,904 | 1.001 | 201.66 | 0.05 | 3687.35 |
| fcgnTest3-150-5000-B | 307,406 | 307,609 | 1.001 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest3-150-5000-C | 335,554 | 333,668 | 0.994 | 3600.00 | 0.03 | 115200.00 |
| fcgnTest3-150-5000-D | 404,886 | 406,103 | 1.003 | 3600.00 | 0.08 | 46082.95 |
| fcgnTest3-150-5000-E | 542,330 | 551,278 | 1.016 | 3600.00 | 0.05 | 76791.81 |
| fcgnTest3-150-5000-F | 793,989 | 808,525 | 1.018 | 23.48 | 0.03 | 751.34 |
| fcgnTest3-150-5000-G | 1,305,463 | 1,314,328 | 1.007 | 3600.00 | 0.03 | 115200.00 |
| fcgnTest3-150-5000-H | 2,465,266 | 2,487,311 | 1.009 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest4-150-5000-A | 300,199 | 300,457 | 1.001 | 287.78 | 0.05 | 6138.67 |
| fcgnTest4-150-5000-B | 326,901 | 327,121 | 1.001 | 71.67 | 0.03 | 2293.43 |
| fcgnTest4-150-5000-C | 362,259 | 363,080 | 1.002 | 3600.00 | 0.02 | 153583.62 |
| fcgnTest4-150-5000-D | 395,566 | 398,868 | 1.008 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest4-150-5000-E | 533,002 | 539,940 | 1.013 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest4-150-5000-F | 809,654 | 816,569 | 1.009 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest4-150-5000-G | 1,268,954 | 1,295,758 | 1.021 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest4-150-5000-H | 2,318,198 | 2,351,353 | 1.014 | 3600.00 | 0.04 | 92165.90 |
| Overall average | 713,689 | 721,693 | 1.008 | 2511.29 | 0.03 | 118726.27 |
| Overall st.dev. | 649,061 | 658,921 | 0.007 | 1563.37 | 0.02 | 105663.93 |
| Minimum | 186,597 | 186,753 | 0.994 | 9.01 | 0.01 | 576.79 |
| Median | 405,482 | 407,543 | 1.007 | 3600.00 | 0.02 | 92165.90 |
| Maximum | 2,465,266 | 2,487,311 | 1.027 | 3600.00 | 0.08 | 460947.50 |

Table 5.11. Empirical results for $[0.5 - 1.5]$ multipliers range

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest1-130-3000-A | 92,417 | 92,877 | 1.005 | 37.50 | 0.02 | 2401.05 |
| fcgnTest1-130-3000-B | 102,556 | 103,715 | 1.011 | 25.21 | 0.04 | 645.40 |
| fcgnTest1-130-3000-C | 115,185 | 117,253 | 1.018 | 8.53 | 0.02 | 546.06 |
| fcgnTest1-130-3000-D | 146,472 | 148,290 | 1.012 | 2.24 | 0.01 | 286.85 |
| fcgnTest1-130-3000-E | 208,957 | 216,289 | 1.035 | 8.47 | 0.02 | 542.38 |
| fcgnTest1-130-3000-F | 329,878 | 350,065 | 1.061 | 44.87 | 0.04 | 1148.80 |
| fcgnTest1-130-3000-G | 525,997 | 561,945 | 1.068 | 478.91 | 0.02 | 20431.23 |
| fcgnTest1-130-3000-H | 909,331 | 999,018 | 1.099 | 80.55 | 0.02 | 5157.03 |
| fcgnTest2-130-3000-A | 95,749 | 96,571 | 1.009 | 12.54 | 0.02 | 535.18 |
| fcgnTest2-130-3000-B | 101,644 | 103,133 | 1.015 | 2.39 | 0.02 | 102.03 |
| fcgnTest2-130-3000-C | 123,007 | 126,032 | 1.025 | 62.80 | 0.02 | 4020.30 |
| fcgnTest2-130-3000-D | 152,589 | 156,552 | 1.026 | 16.69 | 0.02 | 1068.69 |
| fcgnTest2-130-3000-E | 211,423 | 220,774 | 1.044 | 50.78 | 0.02 | 3250.98 |
| fcgnTest2-130-3000-F | 309,944 | 330,598 | 1.067 | 4.18 | 0.02 | 178.51 |
| fcgnTest2-130-3000-G | 515,732 | 562,115 | 1.090 | 21.80 | 0.04 | 558.19 |
| fcgnTest2-130-3000-H | 927,249 | 1,008,104 | 1.087 | 238.57 | 0.02 | 10177.82 |
| fcgnTest1-150-5000-A | 141,444 | 141,911 | 1.003 | 64.98 | 0.02 | 2772.30 |
| fcgnTest1-150-5000-B | 153,395 | 154,173 | 1.005 | 39.44 | 0.02 | 2525.15 |
| fcgnTest1-150-5000-C | 165,237 | 167,270 | 1.012 | 56.56 | 0.02 | 2413.05 |
| fcgnTest1-150-5000-D | 202,641 | 207,865 | 1.026 | 10.65 | 0.02 | 454.19 |
| fcgnTest1-150-5000-E | 269,725 | 277,052 | 1.027 | 770.73 | 0.05 | 16440.51 |
| fcgnTest1-150-5000-F | 377,150 | 393,364 | 1.043 | 43.35 | 0.03 | 1387.18 |
| fcgnTest1-150-5000-G | 591,978 | 627,149 | 1.059 | 1795.29 | 0.04 | 45962.37 |
| fcgnTest1-150-5000-H | 987,862 | 1,084,330 | 1.098 | 2685.09 | 0.04 | 68742.70 |
| fcgnTest2-150-5000-A | 145,634 | 146,140 | 1.003 | 275.00 | 0.02 | 11732.08 |
| fcgnTest2-150-5000-B | 152,547 | 154,688 | 1.014 | 78.31 | 0.03 | 2505.99 |
| fcgnTest2-150-5000-C | 171,869 | 173,661 | 1.010 | 64.73 | 0.04 | 1657.12 |
| fcgnTest2-150-5000-D | 200,605 | 205,896 | 1.026 | 476.10 | 0.03 | 15235.17 |
| fcgnTest2-150-5000-E | 271,000 | 290,289 | 1.071 | 593.82 | 0.02 | 25333.70 |
| fcgnTest2-150-5000-F | 375,329 | 396,433 | 1.056 | 80.06 | 0.02 | 3415.38 |
| fcgnTest2-150-5000-G | 577,676 | 621,744 | 1.076 | 60.42 | 0.02 | 3867.90 |
| fcgnTest2-150-5000-H | 984,349 | 1,056,675 | 1.073 | 103.02 | 0.03 | 3296.54 |
| Overall average | 332,393 | 352,874 | 1.040 | 259.17 | 0.03 | 8087.24 |
| Overall st.dev. | 277,267 | 304,924 | 0.031 | 564.50 | 0.01 | 14679.22 |
| Minimum | 92,417 | 92,877 | 1.003 | 2.24 | 0.01 | 102.03 |
| Median | 205,799 | 212,077 | 1.027 | 58.49 | 0.02 | 2515.57 |
| Maximum | 987,862 | 1,084,330 | 1.099 | 2685.09 | 0.05 | 68742.70 |

Table 5.12. Empirical results for $[1.0 - 1.0]$ multipliers range

| Problem ID | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| fcgnTest7-130-3000-A | 111,283 | 111,817 | 1.005 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest7-130-3000-B | 125,433 | 126,267 | 1.007 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest7-130-3000-C | 144,935 | 145,717 | 1.005 | 3600.00 | 0.06 | 57600.00 |
| fcgnTest7-130-3000-D | 179,764 | 182,204 | 1.014 | 3600.00 | 0.09 | 41889.69 |
| fcgnTest7-130-3000-E | 238,706 | 240,583 | 1.008 | 3600.00 | 0.05 | 65825.56 |
| fcgnTest7-130-3000-F | 370,447 | 373,127 | 1.007 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest7-130-3000-G | 624,195 | 625,540 | 1.002 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest7-130-3000-H | 1,096,563 | 1,108,508 | 1.011 | 3600.00 | 0.03 | 115200.00 |
| fcgnTest8-130-3000-A | 112,833 | 113,199 | 1.003 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest8-130-3000-B | 122,542 | 123,898 | 1.011 | 3600.00 | 0.02 | 153583.62 |
| fcgnTest8-130-3000-C | 143,237 | 144,141 | 1.006 | 3600.00 | 0.05 | 76791.81 |
| fcgnTest8-130-3000-D | 179,973 | 179,984 | 1.000 | 3600.00 | 0.05 | 76791.81 |
| fcgnTest8-130-3000-E | 246,057 | 247,741 | 1.007 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest8-130-3000-F | 378,477 | 380,110 | 1.004 | 3600.00 | 0.04 | 92165.90 |
| fcgnTest8-130-3000-G | 631,238 | 638,869 | 1.012 | 3600.00 | 0.05 | 65825.56 |
| fcgnTest8-130-3000-H | 1,116,227 | 1,122,982 | 1.006 | 3600.00 | 0.02 | 153583.62 |
| fcgnTest7-150-5000-A | 174,885 | 175,606 | 1.004 | 3600.00 | 0.06 | 57600.00 |
| fcgnTest7-150-5000-B | 189,422 | 190,846 | 1.008 | 3600.00 | 0.03 | 115200.00 |
| fcgnTest7-150-5000-C | 219,006 | 220,431 | 1.007 | 3600.00 | 0.08 | 46082.95 |
| fcgnTest7-150-5000-D | 257,661 | 259,674 | 1.008 | 3600.00 | 0.09 | 41889.69 |
| fcgnTest7-150-5000-E | 340,455 | 351,475 | 1.032 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest7-150-5000-F | 492,446 | 499,206 | 1.014 | 3600.00 | 0.07 | 51201.82 |
| fcgnTest7-150-5000-G | 781,501 | 788,399 | 1.009 | 3600.00 | 0.07 | 51201.82 |
| fcgnTest7-150-5000-H | 1,310,903 | 1,324,079 | 1.010 | 3600.00 | 0.03 | 115200.00 |
| fcgnTest8-150-5000-A | 178,830 | 179,353 | 1.003 | 3600.00 | 0.06 | 57600.00 |
| fcgnTest8-150-5000-B | 188,771 | 189,818 | 1.006 | 3600.00 | 0.05 | 65825.56 |
| fcgnTest8-150-5000-C | 216,078 | 218,218 | 1.010 | 3600.00 | 0.06 | 57600.00 |
| fcgnTest8-150-5000-D | 259,043 | 261,126 | 1.008 | 3600.00 | 0.06 | 57600.00 |
| fcgnTest8-150-5000-E | 340,987 | 351,006 | 1.029 | 3600.00 | 0.02 | 230473.75 |
| fcgnTest8-150-5000-F | 482,548 | 488,722 | 1.013 | 3600.00 | 0.03 | 115200.00 |
| fcgnTest8-150-5000-G | 769,946 | 779,461 | 1.012 | 3600.00 | 0.05 | 65825.56 |
| fcgnTest8-150-5000-H | 1,342,828 | 1,347,275 | 1.003 | 3600.00 | 0.06 | 57600.00 |
| Overall average | 417,726 | 421,543 | 1.009 | 3600.00 | 0.05 | 93972.99 |
| Overall st.dev. | 359,558 | 362,327 | 0.007 | 0.00 | 0.02 | 53249.62 |
| Minimum | 111,283 | 111,817 | 1.000 | 3600.00 | 0.02 | 41889.69 |
| Median | 251,859 | 253,708 | 1.007 | 3600.00 | 0.05 | 76791.81 |
| Maximum | 1,342,828 | 1,347,275 | 1.032 | 3600.00 | 0.09 | 230473.75 |

Table 5.13. Empirical results summary for EPTS FCGT with 130 nodes and 3000 arcs

| | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| Overall average | 395,869 | 407,011 | 1.022 | 1493.48 | 0.02 | 58545.95 |
| Overall st.dev. | 409,387 | 420,722 | 0.025 | 1722.20 | 0.02 | 86590.11 |
| Minimum | 83,704 | 84,204 | 1.000 | 0.74 | 0.01 | 91.68 |
| Median | 220,954 | 226,119 | 1.011 | 159.56 | 0.02 | 6640.00 |
| Maximum | 2,033,811 | 2,049,132 | 1.099 | 3600.00 | 0.09 | 460947.50 |

Table 5.14. Empirical results summary for EPTS FCGT with 150 nodes and 5000 arcs

| | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| Overall average | 493,058 | 505,295 | 1.022 | 1761.12 | 0.03 | 55041.79 |
| Overall st.dev. | 472,662 | 481,484 | 0.024 | 1689.51 | 0.02 | 68639.79 |
| Minimum | 133,532 | 133,972 | 0.994 | 4.09 | 0.01 | 131.04 |
| Median | 331,227 | 330,394 | 1.013 | 972.65 | 0.03 | 34873.49 |
| Maximum | 2,465,266 | 2,487,311 | 1.098 | 3600.00 | 0.09 | 230473.75 |

Table 5.15. Empirical results summary for EPTS FCGT “difficult” problems with F, G, H fixed-charge ranges

| | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| Overall average | 838,044 | 864,679 | 1.038 | 1853.93 | 0.03 | 65545.81 |
| Overall st.dev. | 511,048 | 514,990 | 0.030 | 1749.79 | 0.02 | 87353.95 |
| Minimum | 292,146 | 307,546 | 1.002 | 0.91 | 0.01 | 91.68 |
| Median | 720,904 | 728,744 | 1.029 | 2240.19 | 0.03 | 48582.09 |
| Maximum | 2,465,266 | 2,487,311 | 1.099 | 3600.00 | 0.07 | 460947.50 |

Table 5.16. Empirical results summary for EPTS FCGT "easy" problems with A-E fixed-charge ranges

| | CPLEX 12.6.0.0 Solution Value | FIXNET Solution Value | $R$ | CPLEX 12.6.0.0 Time in sec | FIXNET Time in sec | Time Multiple |
|---|---|---|---|---|---|---|
| Overall average | 208,316 | 211,038 | 1.013 | 1491.31 | 0.03 | 51542.71 |
| Overall st.dev. | 98,342 | 99,567 | 0.013 | 1673.06 | 0.02 | 71604.64 |
| Minimum | 83,704 | 84,204 | 0.994 | 0.74 | 0.01 | 94.34 |
| Median | 188,827 | 189,406 | 1.008 | 327.48 | 0.02 | 15239.32 |
| Maximum | 542,330 | 551,278 | 1.071 | 3600.00 | 0.09 | 230473.75 |

## Bibliography


[1] Mathematical Programming Glossary. http://glossary.computing.society.informs.org/ver2/mpgwiki/index.php/Generalized_network.

[2] Adlakha, V., and Kowalski, K. A simple heuristic for solving small fixed-charge transportation problems. *International Journal of Management Science 33* (2003), 205–211.

[3] Adlakha, V., and Kowalski, K. A heuristic algorithm for the fixed charge problem. *Opsearch 47*, 2 (2010), 166–175.

[4] Adlakha, V., Kowalski, K., Wang, S., Lev, B., and Shen, W. On approximation of the fixed charge transportation problem. *Omega 43* (2014), 64–70.

[5] Ahuja, R., Magnanti, T., and Orlin, J. *Network flows*, vol. 91. Prentice-Hall Upper Saddle River, NJ, 1993.

[6] Ali, I., Charnes, A., and Song, T. Design and implementation of data structures for generalized networks. *Journal of Information and Optimization Sciences 7*, 2 (1986), 81–104.

[7] Balas, E. The dual method for the generalized transportation problem. *Management Science 12*, 7 (1966), 555–568.

[8] Balas, E., and Ivanescu, P. On the generalized transportation problem. *Management Science 11*, 1 (1964), 188–202.

[9] Balinski, M. L. Fixed-cost transportation problems. *Naval Research Logistics Quarterly 8*, 1 (1961), 41–54.

[10] Barr, R., Glover, F., and Klingman, D. Enhancements of spanning tree labeling procedures for network optimization. Tech. rep., DTIC Document, 1976.

[11] Barr, R., Glover, F., and Klingman, D. Enhancements of spanning tree labeling procedures for network optimization. *INFOR. Information systems and operational research 17*, 1 (1979), 16–34.

[12] Barr, R. S. Process and applications for the mathematical modeling of data center systems. Technical report 15-EMIS-02, EMIS Department, Lyle School of Engineering, Southern Methodist University, Dallas, Texas, July 2015.

[13] Barr, R. S., Glover, F., and Klingman, D. A new optimization method for large scale fixed charge transportation problems. *Operations Research 29*, 3 (1981), 448–463.

[14] Bazaraa, M., Jarvis, J., and Sherali, H. *Linear programming and network flows*, 4 ed. Wiley Online Library, 2010.

[15] Bazlamacci, C. F., and Hindi, K. S. Enhanced adjacent extreme-point search and tabu search for the minimum concave-cost uncapacitated transshipment problem. *Journal of the Operational Research Society* (1996), 1150–1165.

[16] Bertsekas, D., Castañon, D., Eckstein, J., and Zenios, S. Parallel computing in network optimization. *Handbooks in Operations Research and Management Science 7* (1995), 331–399.

[17] Bertsekas, D. P., and Tseng, P. Relaxation methods for minimum cost ordinary and generalized network flow problems. *Operations Research 36*, 1 (1988), 93–114.

[18] Bixby, R. E. Implementing the simplex method: The initial basis. *ORSA Journal on Computing 4*, 3 (1992), 267–284.

[19] Blue, J. A., and Bennett, K. P. Hybrid extreme point tabu search. *European journal of operational research 106*, 2 (1998), 676–688.

[20] Brown, G., and McBride, R. Solving generalized networks. *Management Science 30*, 12 (1984), 1497–1523.

[21] Buson, E., Roberti, R., and Toth, P. A reduced-cost iterated local search heuristic for the fixed-charge transportation problem. *Operations Research 62*, 5 (2014), 1095–1106.

[22] Charnes, A. Optimality and degeneracy in linear programming. *Econometrica: Journal of the Econometric Society* (1952), 160–170.

[23] Charnes, A., and Cooper, W. W. The stepping stone method of explaining linear programming calculations in transportation problems. *Management Science 1*, 1 (1954), 49–69.

[24] Charnes, A., and Cooper, W. W. Management models and industrial applications of linear programming. *Management Science 4*, 1 (1957), 38–91.

[25] Charnes, A., and Raike, W. M. One-pass algorithms for some generalized network problems. *Operations Research 14*, 5 (1966), 914–924.

[26] Clark, R., Kennington, J., Meyer, R., and Ramamurti, M. Generalized networks: Parallel algorithms and an empirical analysis. *INFORMS Journal on Computing 4*, 2 (1992), 132.

[27] Clark, R., and Meyer, R. Parallel arc-allocation algorithms for optimizing generalized networks. *Annals of Operations Research 22*, 1 (1990), 129–160.

[28] Cunningham, W. H. Theoretical properties of the network simplex method. *Mathematics of Operations Research 4*, 2 (1979), 196–208.

[29] Curet, N. D. An incremental primal-dual method for generalized networks. *Computers & operations research 21*, 10 (1994), 1051–1059.

[30] Dantzig, G. *Linear programming and extensions*. Princeton Univ Press, 1963.

[31] Dantzig, G. B. Upper bounds, secondary constraints, and block triangularity in linear programming. *Econometrica: Journal of the Econometric Society* (1955), 174–183.

[32] Dantzig, G. B., Ford Jr, L. R., and Fulkerson, D. R. A primal–dual algorithm. Tech. rep., DTIC Document, 1956.

[33] Diaby, M. Successive linear approximation procedure for generalized fixed-charge transportation problems. *Journal of the Operational Research Society* (1991), 991–1001.

[34] Eisemann, K. The generalized stepping stone method for the machine loading model. *Management Science 11*, 1 (1964), 154–176.

[35] Elam, J., Glover, F., and Klingman, D. A strongly convergent primal simplex algorithm for generalized networks. *Mathematics of Operations Research* (1979), 39–59.

[36] Engquist, M., and Chang, M. New labeling procedures for the basis graph in generalized networks. *Operations research letters 4*, 4 (1985), 151–155.

[37] Gabriel, S., Manik, J., and Vikas, S. Computational experience with a large-scale, multi-period, spatial equilibrium model of the north american natural gas system. *Networks and Spatial Economics 3*, 2 (2003), 97–122.

[38] Geunes, J., and Pardalos, P. Network optimization in supply chain management and financial engineering: An annotated bibliography. *Networks 42*, 2 (2003), 66–84.

[39] Glover, F. Tabu search, part i. *ORSA Journal on computing 1*, 3 (1989), 190–206.

[40] Glover, F. Tabu search: A tutorial. *Interfaces 20*, 4 (1990), 74–94.

[41] Glover, F. Tabu search, part ii. *ORSA Journal on computing 2*, 1 (1990), 4–32.

[42] Glover, F. Optimization by ghost image processes in neural networks. *Computers & Operations Research 21*, 8 (1994), 801–822.

[43] Glover, F. *Tabu search fundamentals and uses*. Graduate School of Business, University of Colorado Boulder, 1995.

[44] Glover, F. Parametric ghost image processes for fixed-charge problems - a study of transportation networks. *Journal of Heuristics 11*, 4 (2005), 307–336.

[45] Glover, F., Hultz, J., and Klingman, D. Improved computer-based planning techniques, part 1. *Interfaces 8*, 4 (1978), 16–25.

[46] Glover, F., Hultz, J., and Klingman, D. Improved computer-based planning techniques, part 2. *Interfaces 9*, 4 (1979), 12–20.

[47] Glover, F., Hultz, J., Klingman, D., and Stutz, J. A new computer-based planning tool. Tech. rep., DTIC Document, 1976.

[48] Glover, F., Hultz, J., Klingman, D., and Stutz, J. Generalized networks: A fundamental computer-based planning tool. *Management Science 24*, 12 (1978), 1209–1220.

[49] Glover, F., Karney, D., Klingman, D., and Napier, A. A computation study on start procedures, basis change criteria, and solution algorithms for transportation problems. *Management Science 20*, 5 (1974), 793–813.

[50] Glover, F., and Klingman, D. A note on computational simplifications in solving generalized transportation problems. *Transportation Science 7*, 4 (1973).

[51] Glover, F., and Klingman, D. On the equivalence of some generalized network problems to pure network problems. *Mathematical Programming 4*, 1 (1973), 269–278.

[52] Glover, F., Klingman, D., and Napier, A. Basic dual feasible solutions for a class of generalized networks. *Operations Research 20*, 1 (1972), 126–136.

[53] Glover, F., Klingman, D., and Phillips, N. Netform modeling and applications. *Interfaces* (1990), 7–27.

[54] Glover, F., Klingman, D., and Phillips, N. *Network models in optimization and their applications in practice*, vol. 36. Wiley-Interscience, 1992.

[55] Glover, F., Klingman, D., and Stutz, J. Extensions of the augmented predecessor index method to generalized network problems. *Transportation Science 7*, 4 (1973), 377–384.

[56] Glover, F., Klingman, D., and Stutz, J. Augmented threaded index method for network optimization. *Infor 12*, 3 (1974), 293–298.

[57] Glover, F., and Laguna, M. Tabu search. In *Handbook of combinatorial optimization*. Springer, 1998, pp. 2093–2229.

[58] Glover, F., and Laguna, M. *Tabu search - I.* Springer, 1999.

[59] Glover, F., and Taillard, E. A user's guide to tabu search. *Annals of operations research 41*, 1 (1993), 1–28.

[60] Gray, P. Exact solution of the fixed-charge transportation problem. *Operations Research 19*, 6 (1971), 1529–1538.

[61] Greenberg, H. J. *Design and implementation of optimization software.* Sijthoff & Noordhoff [International Publishers], 1978.

[62] Hadley, G. *Linear programming.* Addison-Wesley Publishing Company Inc., 1962.

[63] Hadley, G. *Nonlinear and dynamic programming.* Addison-Wesley Publishing Company Inc., 1964.

[64] Hirsch, W. M., and Dantzig, G. B. The fixed charge problem. *Naval Research Logistics Quarterly 15*, 3 (1968), 413–424.

[65] Jarvis, J. J., Ratliff, H. D., and Trick, M. A. Generalized network implementations. Tech. rep., DTIC Document, 1986.

[66] Jensen, P., and Bard, J. *Operations research: models and methods*, vol. 1. John Wiley & Sons Inc, 2003.

[67] Jensen, P., and Bhaumik, G. A flow augmentation approach to the network with gains minimum cost flow problem. *Management Science 23*, 6 (1977), 631–643.

[68] Jensen, P. A., and Barnes, J. W. *Network flow programming.* John Wiley & Sons, New York, 1980.

[69] Jewell, W., W. Optimal flow through networks with gains. *Operations Research 10* (1962), 476–499.

[70] Jewell, W. S. Optimal flow through networks. In *OPERATIONS RESEARCH* (1958), vol. 6, pp. 633–633.

[71] Johnson, E. L. Networks and basic solutions. *Operations Research 14*, 4 (1966), 619–623.

[72] Jones, R. *Heuristic algorithm for solving the interval flow transshipment network*. PhD thesis, SMU, 2009.

[73] Kallrath, J. Mixed integer optimization in the chemical process industry. *Trans IChemE 78* (2000).

[74] Kantorovich, L. Mathematical methods of organizing and planning production. *Publication House of the Leningrad State University* (1939), 68. Translated in Management Science, V6, No.4:366-422, 1960.

[75] Kelly, J. Formulating production planning models. *Chemical engineering progress 100*, 1 (2004), 43–43.

[76] Kennington, J. The fixed-charge transportation problem: a computational study with a branch-and-bound code. *AIIE Transactions 8*, 2 (1974), 241–247.

[77] Kennington, J., and Helgason, R. *Algorithms for network programming*. John Wiley & Sons, 1980.

[78] Kennington, J., and Lewis. *Encyclopedia of Optimization*, vol. 1. Springer Science & Business Media, 2008, ch. Generalized Networks, pp. 1209–1214.

[79] Kennington, J., and Unger, E. A new branch-and-bound algorithm for the fixed-charge transportation problem. *Management Science 22*, 10 (1976), 1116–1126.

[80] Kennington, J. L., and Mohamed, R. A. An efficient dual simplex optimizer for generalized networks. In *Interfaces in Computer Science and Operations Research*. Springer, 1997, pp. 153–182.

[81] Kim, H., Sohn, H., and Bricker, D. Generation expansion planning using benders' decomposition and generalized networks. *International Journal of Industrial Engineering: Theory, Applications and Practice 18*, 1 (2011).

[82] Kim, S. A network transformation procedure for finding minimal-cost flows in networks with variable lower bounds. Master's thesis, Texas A&M University, (1991).

[83] Kim, Y. An optimal computational approach to the analysis of a generalized network of copper refining process. In *Joint National ORSA/TIMS/AIIE Conference, Atlantic City, NJ* (1972).

[84] Klingman, D., Mote, J., and Phillips, N. A logistics planning system at W.R. Grace. *Operations Research* (1988), 811–822.

[85] Kowalski, K. On the structure of the fixed charge transportation problem. *International Journal of Mathematical Education in Science and Technology 36*, 8 (2005), 879–888.

[86] Lee, Y., and Chen, E. BASF uses a framework for developing web-based production-planning-optimization tools. *Interfaces* (2002), 15–24.

[87] Lourie, J. R. Topology and computation of the generalized transportation problem. *Management Science 11*, 1 (1964), 177–187.

[88] Martin, R. K. *Large scale linear and integer optimization: a unified approach.* Kluwer Academic Publishers, (1999).

[89] McBride, R. D. Solving embedded generalized network problems. *European Journal of Operational Research 21*, 1 (1985), 82–92.

[90] McKeown, P. A vertex ranking procedure for solving the linear fixed-charge problem. *Operations Research 23*, 6 (1975), 1183–1191.

[91] McKeown, P. G. A branch-and-bound algorithm for solving fixed charge problems. *Naval Research Logistics Quarterly 28*, 4 (1981), 607–617.

[92] McKeown, P. G., and Ragsdale, C. T. A computational study of using preprocessing and stronger formulations to solve large general fixed charge problems. *Computers & Operations Research 17*, 1 (1990), 9–16.

[93] Mulvey, J. M. Pivot strategies for primal-simplex network codes. *Journal of the ACM (JACM) 25*, 2 (1978), 266–270.

[94] Mulvey, J. M. An optimization model for the US air-traffic system. Tech. rep., Princeton University, NASA, 1986.

[95] Mulvey, J. M., and Zenios, S. Solving large scale generalized networks. *Journal of Information and Optimization Sciences 6*, 1 (1985), 95–112.

[96] Murty, K. *Network programming*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1992.

[97] Murty, K. G. Solving the fixed charge problem by ranking the extreme points. *Operations research 16*, 2 (1968), 268–279.

[98] Nulty, W. G., and Trick, M. A. GNO/PC generalized network optimization system. *Operations Research Letters 7*, 2 (1988), 101–102.

[99] Orchard-Hays, W., et al. *Advanced linear-programming computing techniques*. McGraw-Hill, 1968.

[100] Orlin, J. On the simplex algorithm for networks and generalized networks. *Mathematical Programming Essays in Honor of George B. Dantzig Part I* (1985), 166–178.

[101] Palekar, U. S., Karwan, M. H., and Zionts, S. A branch-and-bound method for the fixed charge transportation problem. *Management Science 36*, 9 (1990), 1092–1105.

[102] Ramamurti, M. *Parallel algorithms for generalized networks.* PhD thesis, 1989.

[103] Roberti, R., Bartolini, E., and Mingozzi, A. The fixed charge transportation problem: An exact algorithm based on a new integer programming formulation. *Management Science 61*, 6 (2014), 1275–1291.

[104] Sun, M., Aronson, J. E., McKeown, P. G., and Drinka, D. A tabu search heuristic procedure for the fixed charge transportation problem. *European Journal of Operational Research 106*, 2 (1998), 441–456.

[105] Sun, M., and McKeown, P. G. Tabu search applied to the general fixed charge problem. *Annals of Operations Research 41*, 4 (1993), 405–420.

[106] Terlaky, T. *Encyclopedia of Optimization*, vol. 1. Springer Science & Business Media, (2008), ch. Lexicographic Pivoting Rules, pp. 1870–1873.

[107] Terlaky, T., and Zhang, S. Pivot rules for linear programming: a survey on recent theoretical developments. *Annals of Operations Research 46*, 1 (1993), 203–233.

[108] Voss, S. Metaheuristics. In *Encyclopedia of Optimization.* Springer, (2008), pp. 2061–2075.

[109] Walker, W. E. A heuristic adjacent extreme point algorithm for the fixed charge problem. *Management Science 22*, 5 (1976), 587–596.

[110] Wayne, K. D. *Generalized maximum flow algorithms.* PhD thesis, Citeseer, 1999.

[111] Wayne, K. D. A polynomial combinatorial algorithm for generalized minimum cost flow. *Mathematics of Operations Research 27*, 3 (2002), 445–459.

[112] Zionts, S. *Linear and integer programming.* Prentice-Hall, 1974.