MULTI-COMMODITY FLOW ESTIMATION WITH PARTIAL COUNTS ON SELECTED LINKS

A Dissertation

by

DONG HUN KANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2005

Major Subject: Industrial Engineering

MULTI-COMMODITY FLOW ESTIMATION WITH PARTIAL COUNTS ON SELECTED LINKS

A Dissertation

by

DONG HUN KANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Alberto Garcia-Diaz
Committee Members,	César O. Malavé
	Amarnath Banerjee
	Ken Kihm
Head of Department,	Brett A. Peters

December 2005

Major Subject: Industrial Engineering

ABSTRACT

Multi-Commodity Flow Estimation with Partial Counts on Selected Links.

(December 2005)

Dong Hun Kang, B.S., Hanyang University, Korea;

M.S., Hanyang University, Korea

Chair of Advisory Committee: Dr. Alberto Garcia-Diaz

The purpose of this research is to formulate a multi-commodity network flow model for vehicular traffic in a geographic area and develop a procedure for estimating traffic counts based on available partial traffic data for a selected subset of highway links. Due to the restriction of time and cost, traffic counts are not always observed for every highway link. Typically, about 50% of the links have traffic counts in urban highway networks. Also, it should be noted that the observed traffic counts are not free from random errors during the data collection process. As a result, an incoming flow into a highway node and an outgoing flow from the node do not usually match. They need to be adjusted to satisfy a flow conservation condition, which is one of the fundamental concepts in network flow analysis.

In this dissertation, the multi-commodity link flows are estimated in a two-stage process. First, traffic flows of "empty" links, which have no observation data, are filled with deterministic user equilibrium traffic assignments. This user equilibrium assignment scheme assumes that travelers select their routes by their own interests without considering total cost of the system. The assignment also considers congestion effects by taking a link travel cost as a function of traffic volume on the link. As a result, the assignment problem has a nonlinear objective function and linear network constraints. The modified Frank-Wolfe algorithm, which is a type of conditional gradient method, is used to solve the assignment problem.

The next step is to consider both of the observed traffic counts on selected links and the deterministic user equilibrium assignments on the group of remaining links to produce the final traffic count estimates by the generalized least squares optimization procedure. The generalized least squares optimization is conducted under a set of relevant constraints, including the flow conservation condition for all highway intersections.

To my parents and my family
for their steadfast love and support

ACKNOWLEDGEMENTS

I would like to give all my thanks to God who is faithful to answer my prayers during the journey of my Ph.D. program. I also want to express my deep appreciation to Dr. Alberto Garcia-Diaz for being my advisor and giving me his sincere guidance, care and encouragement throughout the years at Texas A&M. He was a good mentor in many aspects of a graduate student's life. Special thanks should be extended to Dr. César O. Malavé, Dr. Amarnath Banerjee, and Dr. Kenneth Kihm for serving on my advisory committee. Thanks to Judy Meeks for her administrative help over the years.

In my eight and half years at College Station, I have been helped by many friends and pastors. Their encouragement and friendship have made me keep going up the hill. I am in great debt of love and prayer to them.

I thank my wife, Ran-Yeong, and my son, Jee-Ho, for their love, patience and support to complete my degree. Finally, I want to thank my mother, father and parents-in-law for their endless love and support.

TABLE OF CONTENTS

CHAPTER Page

I INTRODUCTION....1

I.1. Problem Definition....2
I.2. Research Significance and Contributions....3
I.3. Organization of Dissertation....4

II LITERATURE REVIEW....5

II.1. Introduction....5
II.2. Literature on the Traffic Count Estimation....5
II.3. Literature on the Frank-Wolfe Algorithm....7
II.4. Literature on the Multiple Vehicle Classes....8

III MODEL AND SOLUTION APPROACH....9

III.1. Introduction....9
III.2. Mathematical Models....9
III.3. Overall Solution Approach....15
III.4. Summary....20

IV DEVELOPMENT OF SOLUTION PROCEDURES....21

IV.1. Introduction....21
IV.2. Deterministic User Equilibrium (DUE) Assignment....21
IV.3. Modification of the Frank-Wolfe Algorithm....25
IV.4. Lagrangian Relaxation of Problem....27

V COMPUTERIZATION AND APPLICATIONS....33

V.1. Introduction....33
V.2. Illustration of Real Field Application....33

VI SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS....53

VI.1. Summary and Contribution....54

Page

VI.2. Conclusions 55

VI.3. Recommendations for Further Research 57

REFERENCES 57

APPENDIX A FRANK-WOLFE ALGORITHM 60

APPENDIX B COMPUTER IMPLEMENTATION 73

APPENDIX C COMMON SOLUTION APPROACHES FOR MULTI-COMMODITY NETWORK PROBLEMS 92

VITA 113

LIST OF FIGURES

FIGURE Page

III.1. Travel Cost of a Link Due to Congestion 11

III.2. Overall Conceptual Approach 16

IV.1. Example of Path Flows 22

IV.2. Example of Possible Path Flows at User Equilibrium 22

IV.3. Importance of O-D Traffic Demand Matrix 23

IV.4. Improvement by the Modified Frank-Wolfe Algorithm 26

IV.5. GLS Optimization Example with 2-Commodity Arc Flow Observations 30

IV.6. Traffic Count Estimates of the GLS Example IV.1. 32

V.1. Map of a Metro Area Highway Network 34

V.2. Network Model of the Selected Area in Figure V.1. 36

A.1. Graph of a Frank-Wolfe Algorithm Example 62

A.2. Feasible Direction of Example A.1. 64

A.3. New Iteration Point $\boldsymbol{x_1}$. 65

A.4. Finding Feasible Direction $\boldsymbol{y}_1$ from $\boldsymbol{x}_1$. 66

A.5. New Iteration Point $\boldsymbol{x_2}$. 67

A.6. Finding Feasible Direction $\boldsymbol{y}_2$ from $\boldsymbol{x}_2$. 69

A.7. New Iteration Point $\boldsymbol{x_3}$. 70

A.8. Finding Feasible Direction $\boldsymbol{y}_3$ from $\boldsymbol{x}_3$. 71

Page

A.9. New Iteration Point $\boldsymbol{x_4}$. 72

B.1. Diagram of Matlab Program Modules 74

C.1. Two Commodity Minimum Cost Flow Problem 96

C.2. Matrix Represenatation of Example C.1. 97

C.3. Network Representation of Subproblem 1 of Iteration 1 99

C.4. Network Representation of Subproblem 2 of Iteration 1 100

C.5. Network Representation of Subproblem 1 of Iteration 2 102

C.6. Network Representation of Subproblem 2 of Iteration 2 103

C.7. Network Representation of Subproblem 1 of Iteration 3 105

C.8. Network Representation of Subproblem 2 of Iteration 3 106

C.9. Network Representation of Subproblem 1 of Iteration 4 108

C.10. Network Representation of Subproblem 2 of Iteration 4 109

LIST OF TABLES

TABLE Page

III.1. Notation for Problem (P1) 11

V.1. O-D Demand Matrix 37

V.2. Observed Link Flows for Commodity 1 38

V.3. Observed Link Flows for Commodity 2 39

V.4. Link Time Parameters for Commodity 1 40

V.5. Link Time Parameters for Commodity 2 41

V.6. User Equilibrium Link Flows for Commodity 1 45

V.7. User Equilibrium Link Flows for Commodity 2 46

V.8. Final Traffic Estimates for Commodity 1 49

V.9. Final Traffic Estimates for Commodity 2 51

C.1. Simplex Tableau of the Master Problem 98

C.2. Simplex Tableau of the Iteration 1 101

C.3. Simplex Tableau of the Iteration 2 104

C.4. Simplex Tableau of the Iteration 3 107

CHAPTER I

INTRODUCTION

There are multiple classes of vehicles transporting passengers and goods along their paths in the highway network system. Traffic flows on highway segments are one of the most commonly used data for transportation planning and analysis. Hence, the problem of estimating the number of vehicles traverse on a highway segment (traffic link) is an important issue commanding a great attention from many transportation agencies like the U.S. Federal Highway Administration (FHWA) and the state Department of Transportation (DOT). Many state highway agencies commit a significant portion of their resources to collect data and to determine traffic flow parameters, such as the average annual daily traffic, and the average daily traffic. Usually, the number of vehicles (traffic counts) are collected along major roads and branches to cover the transportation network of interest. However, due to personnel and financial restrictions, not all of the highway links have traffic counts. Approximately half of the links are missing traffic monitoring systems and an accurate analysis of these links require the development of count estimates.

The main objective of this research is to develop a combination of network flow optimization and statistical analysis methods to estimate multi-commodity link flows for a given transportation network with partial sets of link flow observations. Usually transportation system is well described by operation research techniques (Toint, 1997). Traffic flows of a link usually consist of several vehicle classes and each of the vehicle classes is considered as a commodity in the proposed model. The research is motivated by the fact that: (1) highway segments and traffic flows are well suited to network modeling and optimization methods since they are precisely represented by nodes and

---

This dissertation follows the format and style of European Journal of Operational Research.

arcs; (2) traffic count of multiple vehicle classes may be viewed as multi-commodity network flows in the modeling; (3) link flow observations, which are collected at some links of a transportation network, are not free from observational errors. Hence it is more appropriate to deal with them by statistical analysis technique.

I.1. Problem Definition

The purpose of this research is to develop a model and solution methodology to estimate multi-commodity traffic link flows for a given transportation network with O-D demands and partial sets of link flow observations on selected highway links. Given a transportation network it is assumed that there are two sets of links, a set of links having traffic flow observations and a set of links without any observations. A traffic count is a number of vehicles observed by their type (vehicle classes) on a highway link during a certain period of time. It is also assumed that all traffic demands between origins and destinations are known, deterministic values. Considering congestion effect, the travel cost of a link is a function of traffic volume on the link.

The estimation is based on the *deterministic user equilibrium* assignment and *generalized least squares (GLS) optimization*. *User equilibrium* state is achieved when travelers choose their routes with the least possible travel costs without considering the total system cost. At user equilibrium, for each O-D pair, the traffic flows are such that the travel costs of the routes taken are equal to or less than those of unused routes. Another type of equilibrium is the *system equilibrium* (or *system optimum*) which leads to a different assumption on travelers' behavior. At this equilibrium, traffic flows are distributed along the routes of a transportation network in such a manner that the sum of the travel times of all travelers is minimized (Soroush and Mirchandani, 1990). Hence, some travelers may experience relatively high costs of traveling at system equilibrium. Since it is more natural to assume that travelers are free to choose their least cost routes (usually the shortest path) along the given O-D demand matrix, user equilibrium is

adopted for our model. The user equilibrium assignment is also modeled to consider congestion due to high volume of traffic.

Traffic flows are usually observed by automatic traffic sensors installed in the road links, which are partial sets of a specific transportation network. The problem with the observed flows is that they are subject to observation errors and therefore do not satisfy flow conservation condition, which is pursued by the network flow optimization field. In other words, the sum of the incoming flow to a node is not equal to the sum of the outgoing flow from the node. Therefore the observed traffic counts on selected links as well as the user equilibrium assignments on the remaining links must to be adjusted to yield accurate traffic count estimates. This is achieved by using the GLS optimization technique. By GLS optimization the link flow estimates are determined in such a way that the deviation between observational data and the final estimates are minimized.

I.2. Research Significance and Contributions

According to highway statistics 2001 prepared by the Federal Highway Administration (2002), $452.4 million were spent on "Planning and Research" category in federal-aid account by all state governments. Among those states, State of Texas spent $31.7 million on "Planning and Research" during Fiscal Year 2000. Considering many state highway agencies use a significant portion of their personnel and financial resources to data collection and analysis which is performed by transportation planning divisions, a systematic and quantifiable method for estimating current traffic flows is important. It is especially true when only partial set of highway segments has traffic counts and the traffic flows of the remaining segments have to be estimated based on the current available observations.

The overall contributions of this research are summarized as follows.

- *Use of multi-commodity traffic flows in traffic count estimation*: So far, traffic counts are collected and analyzed regardless of the vehicle classes. If individual

traffic count data are needed they have to be split from the whole traffic count data.

- *Development of a computationally efficient solution procedure*: The coverage of transportation network grows as traffic monitoring systems are installed in more areas along the highway. As a result transportation network problem can be large in terms of number of nodes and links or of the O-D pairs. The proposed approach uses computationally efficient method based on network flow optimization methodology.
- *Possible applicability of the proposed procedure*: multi-commodity network flow model is widely used in many areas such as data communication network in which multiple types of signals are carried and need to be estimated.

I.3. Organization of Dissertation

This research is organized as follows: Chapter I has the problem definition, the significance of the research and expected contributions. Chapter II reviews relevant literatures on Traffic Count Estimation Problem. Chapter III presents the mathematical formulation of the problem and the overall solution approach. In Chapter IV the development of each procedure of the proposed solution methodology is provided. Chapter V presents implementation and computational results. Finally, summary, conclusions, and future research are presented in Chapter VI. In addition to the six chapters, detailed Frank-Wolfe Algorithm and actual computer codes implemented in MATLAB are provided in the appendices.

CHAPTER II

LITERATURE REVIEW

II.1. Introduction

The literature on different types of approaches to traffic count estimation problem will be reviewed in this section. Then, several studies on the modification of Frank-Wolfe algorithm to accelerate the slow convergence will be presented in the last section.

II.2. Literature on the Traffic Count Estimation

Numerous researches on traffic count estimation are based on the application of statistical analysis. Shen et al. (1999) developed regression analysis models, which classify roads into several categories and find parameters of each highway category which can be used to estimate annual average daily traffic for the off-system roads in Florida. Ivan and Allaire (2001) also used linear regression analysis to predict traffic volumes for all network links. Their study focused on peak-hour traffic volumes which affect the congestion on the highway links. In the effort to analyze the relationship between the traffic monitoring system location and traffic count estimation errors, Sharma et al. (1996) investigated the statistical precision of annual average daily traffic estimates from short period traffic count observations. Recently, Gazis and Liu (2003) applied Kalman filter for estimating vehicle counts for two roadway sections in tandem. The Kalman filter "is recursive estimator used to estimate the state of a linear time-varying state equation, in which the states are driven by noise and observations are made in the presence of noise" (Moon and Stirling, 2000). The fore mentioned studies used statistical tools to estimate certain type of traffic flows from the observational data.

Since they do not view the flow observations from the highway segments related by network structure, their flow estimation results are inconsistent. That is, there is a discrepancy between the sum of incoming flows into a certain intersection (network node) and the sum of outgoing flows from the intersection. By contrast, Wells and Evans (1989) proposed generalized least squares (GLS) optimization to solve the inconsistency problem in observational data. They estimated link flows and totals flows in a directed acyclic network when the measurement errors on the links are correlated. They used observed link flows and used covariance matrix to formulate a simple quadratic objective function, which gives minimum variances while satisfying network flow conservation constraints and nonnegativity constraints.

Another type of research in traffic flow estimation problem is closely related to the Origin-Destination (O-D) matrix estimation problem. The transportation engineering framework, O-D demand matrix, along with the path choice model and the network model, is an important input to the assignment problem which distributes appropriate traffic flows on the network links. Reversely, measured link flows, along with the path choice model and the network model, are main inputs to O-D estimation problem, which generates traffic demand estimates from every origin to every destination in the network. This relationship is well summarized by Cascetta E. (2001).

Cascetta (1984) introduced GLS estimator into O-D matrix estimation problem. In his work, a GLS estimator was used to combine trip table's direct estimation with traffic counts via an assignment model. Yang et al. (1992) studied the estimation of O-D trip matrices from traffic counts for a congested network case. Yang and Sasaki (1994) extended uncongested estimation model with a linear assignment map to the case with a user equilibrium assignment map, which is formulated as a bilevel optimization problem. In their bilevel optimization model, the upper-level problem seeks to minimize the sum of distance measurements between the observed values to the decision variables, while the lower-level problem represents a user optimal assignment which guarantees that the estimated O-D matrix and the corresponding link flows satisfy the user equilibrium conditions. In their continued study about O-D matrix estimation, Yang (1995)

transformed a bilevel optimization problem, which is computationally and analytically complex, into a single convex program under the assumptions that the traffic counts on each network link are available and constitute a user optimal flow pattern. Compared to the previous nonlinear approach, Sherali et al. (1994) proposed the linear programming approach for estimating O-D trip tables. Their procedure utilizes shortest path network flow programming subproblems to determine a path decomposition of flow that reproduces the observed flows as closely as possible, while seeking a user equilibrium based solution that comes closest to a specified target trip table. Their approach has an advantage of finite convergence of linear programming, while it has the weakness that it requires fairly reliable link flow estimates for the model to be meaningful. Also their approach does not always guarantee the user equilibrium solutions.

II.3. Literature on the Frank-Wolfe Algorithm

The Frank-Wolfe algorithm is one of the approximation algorithms in nonlinear optimization. It generates a *feasible direction* that minimizes the nonlinear objective function at each iteration to find the solution until it satisfies a predefined termination criterion. Since LeBlanc et al. (1975) used the Frank-Wolfe algorithm to solve the traffic assignment problem in their early research, the algorithm has been widely used in the transportation field, because it accounts the network flows problem structure into the approximation and is a relatively effective method in terms of easy procedure and moderate amount of data storage. Hearn and Ribera (1981) showed the convergence of the Frank-Wolfe algorithm when it was modified to include capacity restrictions on some links of a traffic assignment problem. Although the Frank-Wolfe algorithm has advantages for solving traffic assignment problems, it has a problem of slow convergence approaching the optimal point. Several modifications in this approach have been proposed to improve the convergence.

Fukushima (1984) proposed a modified Frank-Wolfe algorithm which utilizes the LP subproblem solutions in some previous iterations to improve search direction from the

current iteration point. Weintraub et al. (1985) suggested using different step sizes during the iteration. They experimented with various step sizes to find the best combination for a set of traffic assignment problems. Another modification to previous studies that did not use the path flow information during iteration was made by Chen et al. (2002), who proposed an algorithm utilizing the path flow information to accelerate the slow convergence.

II.4. Literature on the Multiple Vehicle Classes

So far, there was no distinction of vehicle types for traffic flows occurred in a transportation network. That means that each origin-destination pair has only one type of traffic flow on the paths. However, in some literature, multiple vehicle classes has been considered in the transportation network. Dafermos (1972) presented a multiclass-user model that considered different driver-vehicle combinations sharing a transportation network. Each combination has an individual cost function in an individual way. Her model is also viewed as a multi-commodity model having each combination as a single commodity and formulated to find a *system optimization* flow pattern for the network. Marcotte and Wynter (2004) portrayed the multiclass network equilibrium problem as a nonmonotone, asymmetric, a variational inequality problem. They showed that the problem may have a weaker property in certain conditions and proposed an algorithm utilizing the single-class network equilibrium problem solution technique.

CHAPTER III

MODEL AND SOLUTION APPROACH

III.1. Introduction

Traffic flow estimation problem can be modeled as the application of network flow optimization methods. Since highway intersections and roads can be easily and accurately translated into nodes and links of a network, network flow optimization is a popular tool in the modeling and the analysis of transportation systems. In this chapter, mathematical formulation of models to solve the current problem is described; then the overall conceptual solution approach is presented; and a brief summary of the chapter closes the chapter.

III.2. Mathematical Models

The traffic flow estimation problem goes through two major steps. First, all of the missing link flows are calculated by *multi-commodity deterministic user equilibrium assignment* (MDUE) problem. Then, the assigned link flows and the observed link counts are adjusted by the GLS optimization problem to produce final estimates. These two steps are modeled and represented as mathematical formulation in the following sections.

III.2.1. Deterministic user equilibrium assignment

Transportation planners, after determining the traffic demands between origin and destination nodes, use a traffic assignment model to distribute the traffic demands to

traffic links of the transportation network. This assignment model helps the planners determine the performance of various routes and road segments of the current transportation network. Since the traffic flows are generated by travelers, an assignment model has to reflect the actual behavior of the travelers' route choice as close as possible. Traffic assignment model can be classified by *system optimum* assignment and *user equilibrium* assignment.

System optimum assignment assumes that travelers choose their routes for the benefit of the whole system. Hence, at *system optimum* assignment, traffic flows are distributed to minimize the total cost in the system. The system optimum may be achieved at the expense of some travelers unreasonably high cost. This assignment would be suitable when all trips can be managed by a *supervisor* regardless of the travelers' preferences, such as the transportation schedule of a trucking company.

User equilibrium assignment, on the other hand, assumes that travelers choose their route separately for their own interests. In this case, each traveler compares all possible paths connecting the origin and the destination nodes and selects a minimum cost path. At user equilibrium state, all paths of an origin-destination taken by travelers cost the same or less than those not chosen. For example, suppose there are five paths, r_1, r_2, r_3, r_4 and r_5, where only r_1, r_2, and r_3 are used to satisfy the demand between the origin and the destination.. Then, at equilibrium, paths r_1, r_2 and r_3 cost less than or equal to the travel costs of path r_4 and r_5. Moreover, the travel costs of r_1, r_2 and r_3 are the same.

Since *user equilibrium* assignment reflects travelers' behavior more realistically, it is used in our assignment model. In the assignment model, travel costs of the links are not constant due to the congestion effect. A link travel cost remains constant until the traffic flow on the link reaches at certain congestion point and then increases exponentially as the traffic flow increases. Figure III.1shows the relationship between cost function and traffic flow of a link.

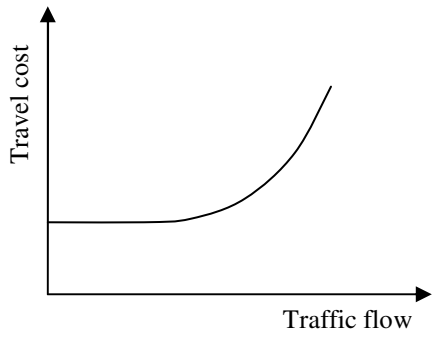


Figure III.1. Travel Cost of a Link Due to Congestion

III.2.2. Formulation of MDUE assignment

Table III.1. shows the parameters and decision variables. Suppose that there is a transportation network $G(N, A)$ where N and A are defined in Table III.1.

Table III.1. Notation for **Problem (*P1*)**

Parameters	
A	set of links, indexed by a
D	set of destination, indexed by j
K	set of commodities, indexed by k
N	set of nodes
O	set of origins, indexed by i
R_{ij}^k	set of all possible paths connecting O-D pair (i, j) for commodity k on link a, indexed by r
t_{ij}^k	demand from origin i to destination j for commodity k
δ_{ijar}^k	1 if h_{ijr}^k passes through arc a, and 0 otherwise
$c_a^k(v)$	travel cost of commodity k on link a when traffic volume is v
m	total number of links
n	total number of nodes
Decision Variables	
x_a^k	traffic flow on link a for commodity k
h_{ijr}^k	r^{th} path flow from origin i to destination j for commodity k

Also suppose that there are traffic flow demands from the origin $i, i \in O$ to the destination $j,\ j \in D$ for each commodity $k, k \in K$.

The decision variable $x^k{}_a$ is traffic flow on link a for commodity k. When the cost functions are monotonically increasing and separable for all links, **Problem (*P1*)** becomes a MDUE *traffic assignment problem*.

Problem (*P1*)

$$\underset{x}{Min} \quad f(\mathbf{x}) = \sum_{k \in K} \sum_{a \in A} \int_0^{x_a^k} c_a^k(v)dv \tag{III.1}$$

$$s.t. \quad x_a^k = \sum_{i \in O} \sum_{j \in D} \sum_{r \in R_{ij}^k} \delta_{ijar}^k h_{ijr}^k \qquad \forall a \in A, k \in K \tag{III.2}$$

$$\sum_{r \in R_{ij}^k} h_{ijr}^k = t_{ij}^k \qquad \forall i \in O,\ j \in D, k \in K \tag{III.3}$$

$$x_a^k \geq 0, h_{ijr}^k \geq 0 \quad \forall a \in A, i \in O,\ j \in D, k \in K, r \in R_{ij}^k \tag{III.4}$$

where $\delta_{ijar}^k = 1$ if h_{ijr}^k passes through arc a, and 0 otherwise.

In the formulation of the **Problem (*P1*)** shown above, the single commodity formulation of the object function was first introduced by Beckmann et al. (1956) and commonly used in transportation engineering discipline. In the objective function (III.1), multi-commodity index k is used to consider multiple vehicle classes for each O-D pair. Appendix C.1 shows the general solution procedures for multi-commodity problems. Notice that current object function is different from the one used in *system optimum* assignment problem having the form:

$$\underset{h}{Min} \quad \sum_{i \in O} \sum_{j \in D} \sum_{k \in K} \sum_{r \in R_{ij}^k} c(h_{ijr}^k) h_{ijr}^k \tag{III.5}$$

where $c(h_{ijr}^k)$ is a path cost and h_{ijr}^k is path flow shown in Table III.1.

The constraints (III.2) characterize the conservation of flow between the link flows and path flows. That is, a traffic flow on link *a* for commodity *k* is the sum of the all path flows about commodity *k* passing through link *a*. The **Problem (*P1*)** is represented as a path-flow formulation because a link-flow representation can not depict individual O-D demand flows in the transportation network. Another type of flow conservation constraints between path flows and traffic demands is shown in (III.3). These constraints set the flow between any origin and destination for commodity *k* equal to the sum of the flows for that commodity on all paths connecting a specific origin to its destination.

III.2.3. Generalized least squares (GLS) optimization problem

By *user equilibrium* assignment, every link has assigned link flows for each commodity. Even though current assignment could be a reasonable approximation of the actual traffic occurred in the transportation network, it can be improved by exploiting the traffic count data which are collected from different part of the network. In order to do so, any assigned link flow is replaced by the corresponding observational traffic count. Afterwards, the combined link traffic flows will be adjusted to produce the final traffic count estimates by GLS optimization procedure.

GLS optimization is conducted under a set of relevant constraints, including the flow-conservation condition for all highway intersections. Additional constraints could be formulated to represent multiple types of vehicle classes.

Suppose that we have an actual traffic count observation y_a, $a \in A$. In general, y_a is not the true flow on link *a* due to unknown errors and has a variance σ_a^2. We assume that the observation of the flow on a link *a*, y_a is a random variable which is modeled as

$$y_a = x_a + \varepsilon_a, \quad a \in A \tag{III.6}$$

where x_a is true flow on link a, ε_a is a normal random variable with mean 0 and variance σ_a^2. By taking expectations on (III.6) we can confirm y_a is an unbiased estimator of x_a.

$$E(y_a) = E(x_a + \varepsilon_a) = x_a, \quad a \in A \tag{III.7}$$

The variance is a measure of the amount of variability inherent in observing the flow along link a. There may also be covariance between the link flow observations. In this situation, the mathematical modeling of the GLS optimization is shown in the matrix form of **Problem (P2)**.

Problem (*P2*)

$$\underset{x}{Min} \quad \sum_{k \in K} (\mathbf{y}^k - \mathbf{x}^k)'(\mathbf{V}^k)^{-1}(\mathbf{y}^k - \mathbf{x}^k) \tag{III.8}$$

$$s.t. \quad \mathbf{A}^k \mathbf{x}^k = \mathbf{b}^k \qquad \forall\, k \in K \tag{III.9}$$

$$\sum_{k \in K} \mathbf{x}^k \le \mathbf{u} \tag{III.10}$$

$$\mathbf{x}^k \ge \mathbf{0} \qquad \forall\, k \in K \tag{III.11}$$

Suppose that there is a set of links P with traffic flow observations, Q without observations, and p and q, the respective number of elements of set P and Q. Here, $P \cup Q \equiv A$ and $p + q = m$. In the objective function (III.8), the observation vector $\mathbf{y}^k$ consists of $\mathbf{y}_\mathbf{P}^k$, the vector of actual flow observation for commodity k and $\mathbf{y}_\mathbf{Q}^k$, the vector of user equilibrium traffic flows given from the solution of **Problem (*P1*)**. $\mathbf{V}^k$ is a variance-covariance matrix of commodity k, whose diagonal elements represent the variances of the link flow observations, and the remaining elements show covariance between the link flows. In the model, we take the inverse of the matrix to give more weight on the flow observation with less variance. Constraints (III.9) indicate that flow conservation should be met at each node for all commodity k, meaning the sum of

incoming flows into a node is equal to the sum of outgoing flows from the node. This holds for each commodity. Here, $\mathbf{A}^k$ is the node-link incident matrix for the given network *G(N,A)* and it has the same elements for all commodity. All or most of the constraints (III.10) would not be bound, since an assigned or observed link flow would not exceed the link capacity. In the user equilibrium assignment, link capacity is maintained implicitly by nonlinear link cost functions. Observed link flows also satisfy link capacity by nature. Thus, constraints (III.10) could be changed with other types of restrictions imposed on the arc in accordance with multi-commodity flows.

III.3. Overall Solution Approach

Traffic flow estimates are obtained by sequentially solving the previous two problems, **Problem (*P1*)** and **Problem (*P2*)**. First, the MDUE traffic assignment **Problem (*P1*)** is solved to distribute traffic flows based upon the user equilibrium scheme. Using the calculated MDUE assignment solution, partial link counts and a variance-covariance matrix, the GLS optimization **Problem (*P2*)** is solved to obtain final link flow estimates.

Due to the adoption of nonlinear cost functions, which take congestion effects into account, the traffic assignment **Problem (*P1*)** has a nonlinear objective function with linear network flow constraints. An efficient approach to solving this nonlinear optimization problem is to use a linear approximation methodology known as the *conditional gradient method* or, the *Frank-Wolfe* algorithm. The *Frank-Wolfe* algorithm solves a linear programming problem (network flow optimization problem in the current case) and a line search repeatedly until a predefined stopping criterion is satisfied (Bell and Iida, 1997).

So far partial link flow observations are not used in **Problem (*P1*)** above because the observational data are incompatible with the deterministic network modeling structure. This prompts the use of a GLS optimization method as a next step. GLS estimation is used to find estimators which minimize the weighted sum of squared distances between

observed values and estimated variables. The GLS estimation was utilized in the current problem to give observed data with smaller variance the more weight and to consider the possible covariance between the link flows.

Figure III.2 shows the overall conceptual approach of the proposed methodology.

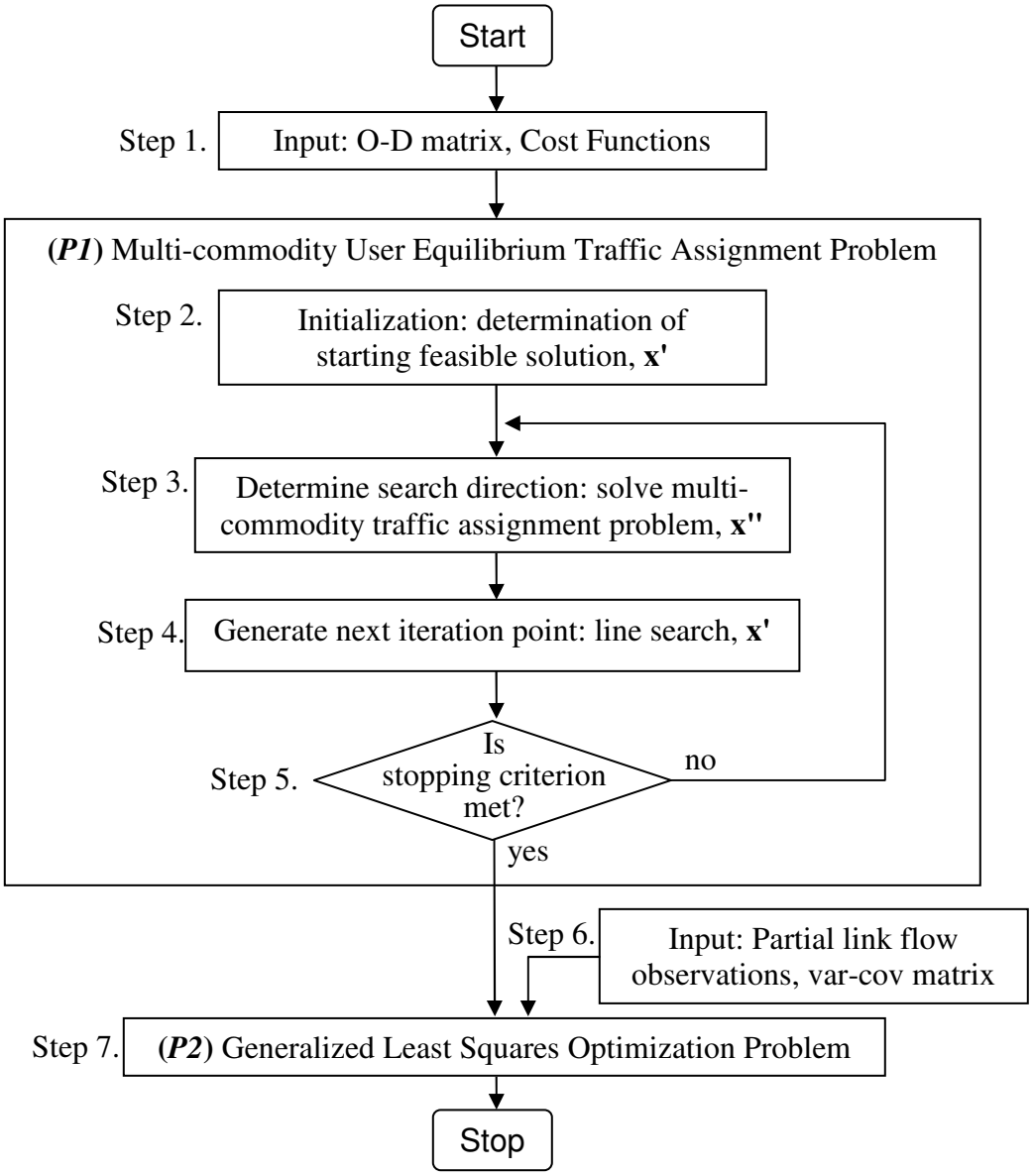


Figure III.2. Overall Conceptual Approach

The proposed approach starts with the preparation of input data. In step 1, O-D matrix gives the multi-commodity traffic demands between every origin-destination pair in the transportation network. Regarding cost functions, each link may have different parameters considering the congestion effects on it. Furthermore, the sensitivity of the cost function in response to congestion may be different for each vehicle classes. One of the widely used cost function is suggested by Bureau of Public Roads (BPR) (1964):

$$c_a(v_a) = c_a(o)\left[1+\beta\left(\frac{v_a}{u_a}\right)^{\gamma}\right] \tag{III.12}$$

where $c_a(0)$ is a free flow travel cost on link a, u_a is the a flow capacity of link a. Here, β and γ are pre-given parameters and set to $\beta = 0.15$ and $\gamma = 4$. Since the BPR cost function does not give different link costs for different commodities (vehicle classes) on the same link, the BPR function is modified to include indices for multi-commodity flows as shown below.

$$c_a^k(v_a^k) = c_a^k(o)\left[1+\beta\left(\frac{v_a^k}{u_a^k}\right)^{\gamma}\right] \tag{III.13}$$

From Step 2 to Step 5, MDUE traffic assignment **Problem (*P1*)** is solved. **Problem (*P1*)** is solved by Frank-Wolfe algorithm which is customized to solve our traffic assignment problem. In Step 2, a starting feasible solution (**x**') is determined by solving **Problem (*P3*)** with initial flows of zero on all links. The **Problem (*P3*)** is having an objective function of the first-order Taylor expansion around $\overline{x_a^k}$ of the **Problem (*P1*)** with identical constraints. The more detailed description of the Frank-Wolfe algorithm is explained in the Appendix.

Problem (*P3*)

$$\underset{x}{Min} \quad \sum_{k\in K}\sum_{a\in A} c_a^k(\overline{x_a^k})x_a^k \tag{III.14}$$

$$s.t. \quad x_a^k = \sum_{i\in O}\sum_{j\in D}\sum_{r\in R_{ij}^k} \delta_{ijar}^k h_{ijr}^k \qquad \forall a\in A, k\in K \tag{III.15}$$

$$\sum_r h_{ijr}^k = t_{ij}^k \qquad \forall i\in O,\ j\in D, k\in K \tag{III.16}$$

$$x_a^k \geq 0, h_{ijr}^k \geq 0 \quad \forall a\in A, i\in O,\ j\in D, k\in K, r\in R_{ij}^k \tag{III.17}$$

where $\overline{x_a^k}$ (or **x**' in vector form) is the solution from Step 2 and

$\delta_{ijar}^k = 1$ if h_{ijr}^k passes through arc *a*, and 0 otherwise.

In Step 3, the algorithm determines the search direction of our linear approximation scheme. The starting feasible flows (**x**') in Step 2 generate new link cost coefficients that convert the original nonlinear **Problem (*P1*)** to a linear programming problem, more specifically, a multi-commodity traffic assignment **Problem (*P3*)**.

In order to consider O-D demand flows explicitly, current multi-commodity traffic assignment **Problem (*P3*)** is represented by the path-flow based formulation. Since the problem is a variation of a multi-commodity network flows optimization problem, any solution technique used for multi-commodity network flows optimization problem, such as the column generation procedure could be used. The column generation procedure generates the columns "as needed" bases during the solution process. After solving **Problem (*P3*)**, new link flows solution (**x**") is generated. In Step 4, a convex combination of the current link flows (**x**") and the previous link flows **x**' is sought to minimize the original objective function (III.1) as shown in **Problem (*P4*)**.

Problem (*P4*)

$$Min\ f(\mathbf{x} = \mathbf{x}'\lambda + \mathbf{x}''(1-\lambda)) \tag{III.18}$$

$$s.t. \quad 0 \leq \lambda \leq 1$$

Problem (*P4*) is an simple nonlinear optimization problem in one variable, λ, for which a number of techniques are available such as the *Golden Section* search method and the *bisection method*. This generates a new link flows solution (**x**') with which convergence test is performed in Step 5. In Step 5, newly obtained link flows (**x**') and the link flows solution from Step 3 (**x**") are compared. If the predefined convergence condition is met, the procedure goes to Step 7 along with the inputs from Step 6. Otherwise the algorithm returns to Step 3 with the current solution **x**' as an input to the step. There is a number of ways to access the degree of convergence. One way is to observe the relative changes in the vector of current link flows between iterations. Another way is to compare the changes in the final trip costs resulted from the current link flows.

After completing Step 1 through 5, we obtain user equilibrium link flows based on a given O-D demand matrix and link cost functions. These flows, however, may be different from the true traffic volumes of the links, since there always exist a possibility that O-D matrix and/or link cost functions are inaccurately describe a given transportation network. Therefore, current link flows must be adjusted to comply with the real network behavior as closely as possible. This process is performed by considering the partial link flow observations in the GLS optimization model **(*P2*)**. The actual observation data replace the corresponding link flows, which are calculated from the first 5 steps. Also, variance-covariance matrix is given as an input to the GLS optimization problem.

In Step 7, GLS estimation starts with the inputs of link flows calculated from the previous assignment problem, partial link flow observations, and variance-covariance matrix, **V**, of the network links. The GLS optimization **Problem (*P2*)** reduces to a quadratic programming with linear constraints. This quadratic programming can be solved with the Lagrangian relaxation method without considering multi-commodity capacity constraints (III.10). The solution vector of the Lagrangian relaxation problem is as follows:

$$\mathbf{x} = [\mathbf{I} - \mathbf{V}\mathbf{A}' (\mathbf{A}\mathbf{V}\mathbf{A}')^{-1} \mathbf{A}] \mathbf{y} + \mathbf{V}\mathbf{A}' (\mathbf{A}\mathbf{V}\mathbf{A}')^{-1} \mathbf{b} \qquad \text{(III.19)}$$

Detailed equations of the solution steps are given in Chapter IV. Notice that constrains (III.10) will not be active in general, since the user equilibrium link flows calculated from **Problem (*P1*)** implicitly consider the link capacity by flow-dependent nonlinear cost functions. However, if any of the capacity constraint is active after replacing user equilibrium link flows with the partial link flow observations, the quadratic programming problem can be solved by the *active set method*.

III.4. Summary

Multi-commodity deterministic user equilibrium (MDUE) traffic assignment is formulated to allocate the traffic demands to the traffic links, based on the selfish route choice assumption. The problem also take congestion effects into account in the flow volume dependent cost function. The formulation of the assignment problem turns out to be a nonlinear programming problem with linear network flow constraints and is solved by a linear approximation method called *Frank-Wolfe* algorithm using the conditional gradient method.

As a second stage of the proposed traffic flow estimation procedure, GLS optimization is formulated to determine the final traffic flow estimates, that minimizes the total deviation between the observational data and the final estimates. The proposed solution approach is described in detail in Chapter IV.

CHAPTER IV

DEVELOPMENT OF SOLUTION PROCEDURES

IV.1. Introduction

In this chapter, the procedure of the solution approach for algorithm described in the previous chapter is explained in detail. Section IV.2. describes the basic assumptions and properties of the deterministic user equilibrium assignment. Section IV.3 presents a modified Frank-Wolfe algorithm, which is based on the heuristic method by Weintraub et al. (1985), to accelerate the slow convergence near optimal point. Section IV.4 shows that GLS optimization problem is solved by Lagrangian relaxation method in detail.

IV.2. Deterministic User Equilibrium (DUE) Assignment

A transportation network $G(N, A)$ consists of a set of nodes, N, and a set of links, A. Nodes represent conceptual or physical intersections in the transportation network and links represent road segments connecting the nodes. A link cost represents the sum of all costs required to travel the link. An origin is a node where traffic flows are generated and a destination is a node where the traffic flows terminate. In the proposed model, all network links are directed links in order to represent the real highway network. Also, a commodity represents a type of vehicle classes.

Path-flow representation is important for modeling DUE assignment since all selected paths of an O-D pair should have the same travel cost at user equilibrium. Figure IV.1 shows an example of path flows between origin node 1 and destination node 3 in a transportation network.

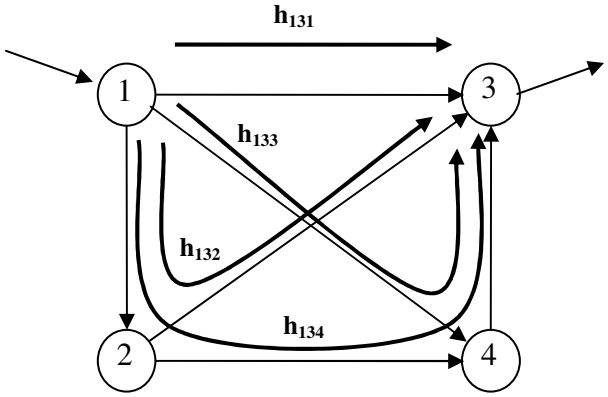


Figure IV.1. Example of Path Flows

In the Figure IV.1, h_{ijr} represents a r^{th} path flow from origin i to destination j. Even though there are four possible paths in the example above, all paths do not necessarily have to be taken for traveling between the O-D pair at user equilibrium.

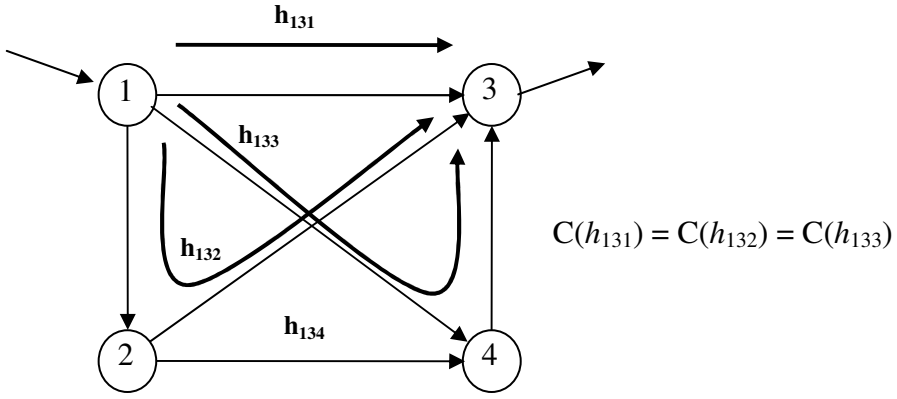
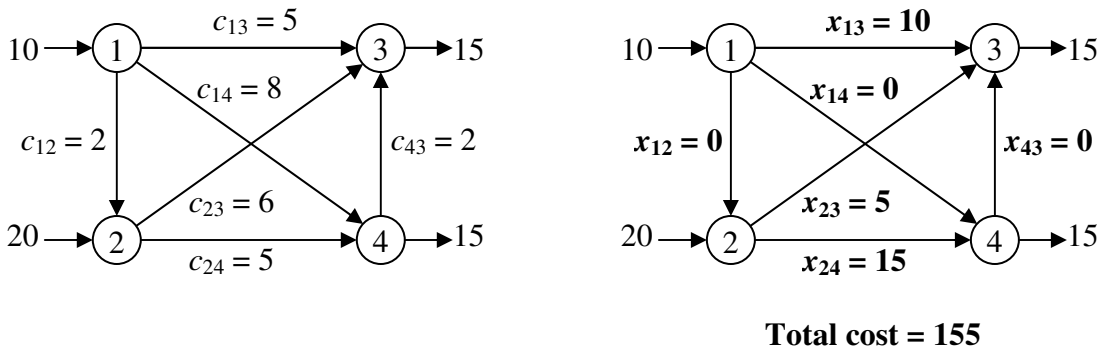


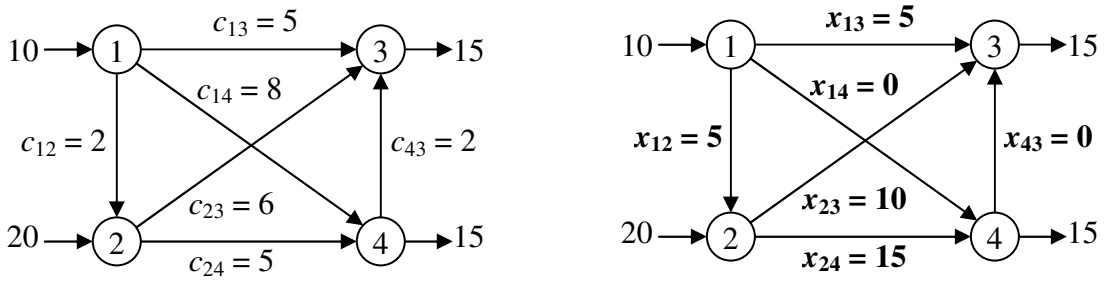
Figure IV.2. Example of Possible Path Flows at User Equilibrium

Figure IV.2. shows one of possible scenarios of user equilibrium when it is assumed that three path, h_{131}, h_{132}, and h_{133}, are selected. However, the travel cost between the selected paths must the same at user equilibrium condition.

Another important assumption for DUE is that O-D traffic demand are given as fixed values. Consider the following example shown in Figure IV.3.



(A) Min cost assignment



O\D	3	4
1	5	5
2	10	10

(B) Min cost assignment with O-D demand matrix

Figure IV.3. Importance of O-D Traffic Demand Matrix

In the above Figure IV.3, c_{ij} is total link costs required to travel link (i, j) and x_{ij} is a assigned link flows. The two problems have the same network with identical link costs

except the second one is given an O-D traffic demand matrix. As shown in the Figure above, the two problem have different link flow assignments and total minimum costs. Notice that the problem with O-D traffic demand matrix has a higher total minimum cost of 170 due to additional constraints by the O-D matrix.

IV.2.1 Variational inequality formulation for DUE

Consider link flows, $\boldsymbol{x}$, path flows, $\boldsymbol{h}$, link cost, $\boldsymbol{c}$, and path cost, $\boldsymbol{g}$. Then the variational inequality is formulated as follows:

$$(\boldsymbol{x} - \boldsymbol{x}^*)' \boldsymbol{c}(\boldsymbol{x}^*) \geq 0 \qquad \text{(IV.1)}$$

or

$$(\boldsymbol{h} - \boldsymbol{h}^*)' \boldsymbol{g}(\boldsymbol{h}^*) \geq 0 \qquad \text{(IV.2)}$$

where $\boldsymbol{x}^*$ is the vector of DUE link flows and $\boldsymbol{h}^*$ is the vector of DUE path flows.

The variational inequality describes that, given user equilibrium path costs (link costs), any deviation from the user equilibrium path flows (link flows) can not reduce total costs. That is, at user equilibrium, any traveler can not reduce his trip cost by changing his path.

IV.2.2 Existence of DUE link flows

Theorem 1. The variational inequality (IV.1) and (IV.2) have at least one solution if the cost functions are continuous functions, defined on the non-empty, compact and convex set of the feasible link flows or path flows

Proof. See Cascetta, E. (2001).

IV.2.3 Uniqueness of DUE link flows

Theorem 2. Given the existence of a DUE assignment, if the *Jacobian* of the link cost functions is positive definite then the assignment is unique and *vice versa*.

Proof. See Bell, M. G. and Iida, Y. (1997).

IV.3. Modification of the Frank-Wolfe Algorithm

Frank-Wolfe algorithm is a popular algorithm for solving nonlinear programming problems. It is especially advantageous in traffic assignment problems because it does not need enumerating all possible paths between origins and destinations, which could be very demanding work as the size of the network grows. Also, it allows to use very efficient network flows algorithms as its subproblems. However, it has a tendency that the convergence becomes slow as it approaches to the optimal point. In this section, the modification of the Frank-Wolfe algorithm is proposed based on Weintraub et al.'s (1985) approach by a using different step sizes.

The modified Frank-Wolfe algorithm:

0. Choose a starting feasible solution $\boldsymbol{x}^1$ of **Problem (*P1*)**
   Let iteration counter $k = 1$
1. Solve **Problem (*P3*)**. Let $\boldsymbol{y}^k$ be the solution.
   Search direction $\boldsymbol{d}^k = \boldsymbol{y}^k - \boldsymbol{x}^k$
2. Solve **Problem (*P4*)**. Let λ_0 be the solution.
3. Let $\lambda^k = \alpha^k \lambda_0$, α^k is a predefined modification for k^{th} step size where $\alpha^k > 1$.
4. Let $\lambda^k = \min(\lambda^k, 1)$
5. Calculate the next iteration point.
   $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \lambda^k \boldsymbol{d}^k$
6. Check the improvement by new step size.
   If $f(\boldsymbol{x}^{k+1}) < f(\boldsymbol{x}^k)$ then $\boldsymbol{x}^{k+1} = \boldsymbol{x}^{k+1}$. Otherwise $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \lambda_0 \boldsymbol{d}^k$

The modified Frank-Wolfe algorithm uses, whenever possible, the larger step sizes than the original one. The motivation of this heuristic is based on the observation that the step sizes of the original Frank-Wolfe algorithm diminishes while approaching the optimal point. By using the larger step sizes, the modified Frank-Wolfe algorithm compensate the diminishing effect.

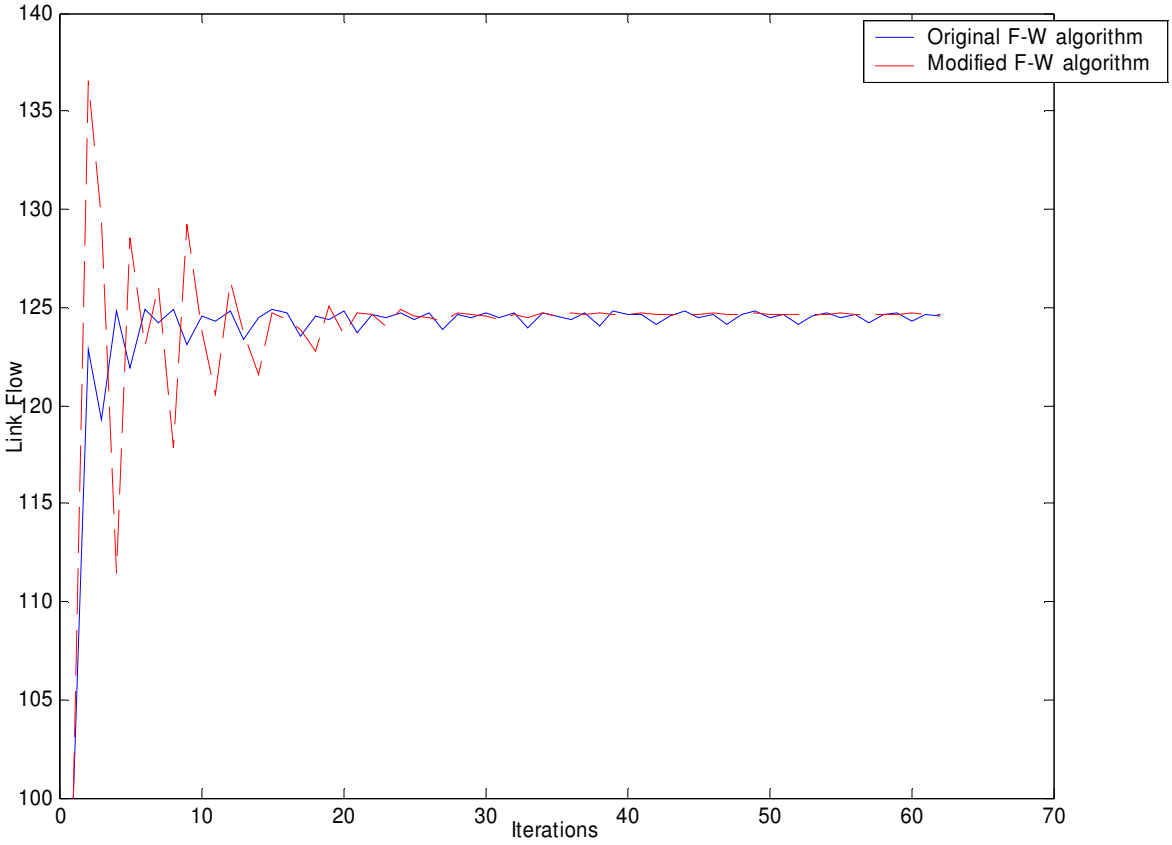


Figure IV.4. Improvement by the Modified Frank-Wolfe Algorithm

Figure. IV.4. shows outputs from the original F-W algorithm and the modified F-W algorithm. As shown in the graph above, the modified F-W algorithm reaches an optimal point around after 20 iterations. However, the original F-W algorithm shows it

is still seeking the optimal point after 60 iterations. Notice that the modified F-W algorithm has wider zigzag pattern than the original F-W algorithm.

IV.4. Lagrangian Relaxation of Problem

In order to solve the GLS optimization **Problem (*P2*),** the Lagrangian relaxation approach is used. A Lagrangian relaxation method removes the constraints and include them in the objective function by adopting Lagrangian multipliers for each constraints. Therefore, a constrained optimization problem is reduced to an optimization of the Lagrangian function.

For the sake of simplicity, following conversions are made from the GLS optimization formulation.

$$\mathbf{y}=\begin{bmatrix}\mathbf{y}^1\\ \vdots\\ \mathbf{y}^k\end{bmatrix},\ \mathbf{x}=\begin{bmatrix}\mathbf{x}^1\\ \vdots\\ \mathbf{x}^k\end{bmatrix},\ \mathbf{V}=\begin{bmatrix}\mathbf{V}^1 & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \ddots & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{V}^k\end{bmatrix},\ \mathbf{A}^*=\begin{bmatrix}\mathbf{A}^1 & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \ddots & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{A}^k\end{bmatrix},\ \mathbf{b}^*=\begin{bmatrix}\mathbf{b}^1\\ \vdots\\ \mathbf{b}^k\end{bmatrix}$$

In the notations above, node-are incident matrices A^1 , …, A^k are not full rank (see Bazaraa et al. (1990)) since the sum of its rows is zero vector. Therefore, in actual calculation, one constraint from the node-arc incident matrix for each commodity is removed to make it nonsingular. Let's assume that A and b are a resulting matrix and a vector after removing *k* rows from A^* and from b^*. Then, the GLS optimization can be reformulated as:

Problem (*GLS*)

$$Min \quad (\mathbf{y}-\mathbf{x})'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{x}) \tag{IV.3}$$

$$s.t. \quad \mathbf{Ax}=\mathbf{b} \tag{IV.4}$$

$$\mathbf{x}\geq\mathbf{0} \tag{IV.5}$$

where y is a vector obtained by combining observed counts and traffic assignments. V is a variance-covariance matrix. A is a node-link incidence matrix of the network. b is a vector having elements of zero when a node is a transshipment node, positive-value when a node is an origin (or source) node, and negative-value when a node is a destination (or terminal) node.

The constraints (IV.4) are included in the objective function by multiplying Lagrangian multipliers λ. Then the following unconstrained minimization problem is obtained by equation:

Problem (*LR*)

$$Min \quad L(\mathbf{x}, \boldsymbol{\lambda}) = (\mathbf{y} - \mathbf{x})'\mathbf{V}^{-1}(\mathbf{y} - \mathbf{x}) - \boldsymbol{\lambda}' (\mathbf{A}\mathbf{x} - \mathbf{b}) \qquad \text{(IV.6)}$$

Notice that the inequality constraints (IV.5) are not considered in (IV.6) for a practical situation, where the solution of **Problem (GLS)** would not be bound by constraints (IV.5). If we assume that x^* and λ^* as the optimal solution of **Problem (*LR*)**, then they must satisfy the following conditions:

$$\partial L / \partial \mathbf{x}^* = \mathbf{0} \text{ and}$$
$$\partial L / \partial \boldsymbol{\lambda}^* = \mathbf{0}.$$

By differential operations for vectors (see Searle, 1982), it is obtained that:

$$\partial L / \partial \mathbf{x}^* = \partial / \partial \mathbf{x}^* \left[(\mathbf{y} - \mathbf{x}^*)'\mathbf{V}^{-1}(\mathbf{y} - \mathbf{x}^*) - \boldsymbol{\lambda}^{*\prime}(\mathbf{A}\mathbf{x}^* - \mathbf{b}) \right]$$
$$= \partial / \partial \mathbf{x}^* \left[(\mathbf{y}'\mathbf{V}^{-1} - \mathbf{x}^{*\prime}\mathbf{V}^{-1})(\mathbf{y} - \mathbf{x}^*) - \boldsymbol{\lambda}^{*\prime}(\mathbf{A}\mathbf{x}^* - \mathbf{b}) \right]$$
$$= \partial / \partial \mathbf{x}^* \left[\mathbf{y}'\mathbf{V}^{-1}\mathbf{y} - \mathbf{x}^{*\prime}\mathbf{V}^{-1}\mathbf{y} - \mathbf{y}'\mathbf{V}^{-1}\mathbf{x}^* + \mathbf{x}^{*\prime}\mathbf{V}^{-1}\mathbf{x}^* - \boldsymbol{\lambda}^{*\prime}\mathbf{A}\mathbf{x}^* - \boldsymbol{\lambda}^{*\prime}\mathbf{b} \right]$$

$$= -\mathbf{V}^{-1}\mathbf{y} - (\mathbf{y}'\mathbf{V}^{-1})' + 2\mathbf{V}^{-1}\mathbf{x}^* - (\boldsymbol{\lambda}^{*\prime}\mathbf{A})'$$

$$= -2\mathbf{V}^{-1}\mathbf{y} + 2\mathbf{V}^{-1}\mathbf{x}^* - \mathbf{A}'\boldsymbol{\lambda}^*$$

$$= -2\mathbf{V}^{-1}(\mathbf{y} - \mathbf{x}^*) - \mathbf{A}'\boldsymbol{\lambda}^* = \mathbf{0}$$

Then, $2\mathbf{V}^{-1}(\mathbf{y} - \mathbf{x}^*) = \mathbf{A}'\boldsymbol{\lambda}^*$

$$\mathbf{y} - \mathbf{x}^* = \frac{1}{2}\mathbf{V}\mathbf{A}'\boldsymbol{\lambda}^*$$

$$\mathbf{x}^* = \mathbf{y} - \frac{1}{2}\mathbf{V}\mathbf{A}'\boldsymbol{\lambda}^* \qquad \text{(IV.7)}$$

Also,

$$\partial L / \partial \boldsymbol{\lambda}^* = \partial / \partial \boldsymbol{\lambda}^* \left[(\mathbf{y} - \mathbf{x}^*)'\mathbf{V}^{-1}(\mathbf{y} - \mathbf{x}^*) - \boldsymbol{\lambda}^{*\prime}(\mathbf{A}\mathbf{x}^* - \mathbf{b}) \right]$$

$$= \partial / \partial \boldsymbol{\lambda}^* \left[-\boldsymbol{\lambda}^{*\prime}(\mathbf{A}\mathbf{x}^* - \mathbf{b}) \right]$$

$$= \mathbf{b} - \mathbf{A}\mathbf{x}^* = \mathbf{0}$$

Hence, $\mathbf{b} = \mathbf{A}\mathbf{x}^*$ (IV.8)

From (IV.7) and (IV.8)

$$\mathbf{b} = \mathbf{A}\mathbf{x}^* = \mathbf{A}(\mathbf{y} - \frac{1}{2}\mathbf{V}\mathbf{A}'\boldsymbol{\lambda}^*)$$

$$= \mathbf{A}\mathbf{y} - \frac{1}{2}\mathbf{A}\mathbf{V}\mathbf{A}'\boldsymbol{\lambda}^*$$

By solving about $\boldsymbol{\lambda}^*$,

$$\mathbf{A}\mathbf{y} - \frac{1}{2}\mathbf{A}\mathbf{V}\mathbf{A}'\boldsymbol{\lambda}^* = \mathbf{b}$$

$$\mathbf{A}\mathbf{V}\mathbf{A}'\boldsymbol{\lambda}^* = 2(\mathbf{A}\mathbf{y} - \mathbf{b})$$

$$\boldsymbol{\lambda}^* = 2(\mathbf{A}\mathbf{V}\mathbf{A}')^{-1}(\mathbf{A}\mathbf{y} - \mathbf{b}) \qquad \text{(IV.9)}$$

By applying $\boldsymbol{\lambda}^*$ to (IV.7),

$$\begin{aligned}\mathbf{x}^* &= \mathbf{y} - \frac{1}{2}\mathbf{V}\mathbf{A}'\boldsymbol{\lambda}^* \\ &= \mathbf{y} - \mathbf{V}\mathbf{A}'(\mathbf{A}\mathbf{V}\mathbf{A}')^{-1}(\mathbf{A}\mathbf{y} - \mathbf{b}) \\ &= \mathbf{y} - \mathbf{V}\mathbf{A}'(\mathbf{A}\mathbf{V}\mathbf{A}')^{-1}\mathbf{A}\mathbf{y} + \mathbf{V}\mathbf{A}'(\mathbf{A}\mathbf{V}\mathbf{A}')^{-1}\mathbf{b} \\ &= [\mathbf{I} - \mathbf{V}\mathbf{A}'(\mathbf{A}\mathbf{V}\mathbf{A}')^{-1}\mathbf{A}]\mathbf{y} + \mathbf{V}\mathbf{A}'(\mathbf{A}\mathbf{V}\mathbf{A}')^{-1}\mathbf{b} \qquad \text{(IV.10)}\end{aligned}$$

Since V^{-1} is positive definite $(AVA)^{-1}$ is also a positive definite matrix and therefore nonsingular. Thus (IV.9) and (IV.10) hold for optimal solution λ^* and x^*.

Example IV.1. Consider the following two-commodity transportation network with the given link flow observation data and a variance-covariance matrix. Due to unknown observation errors, as shown in the Figure IV.5, the link flows do not satisfy flow conservation condition.

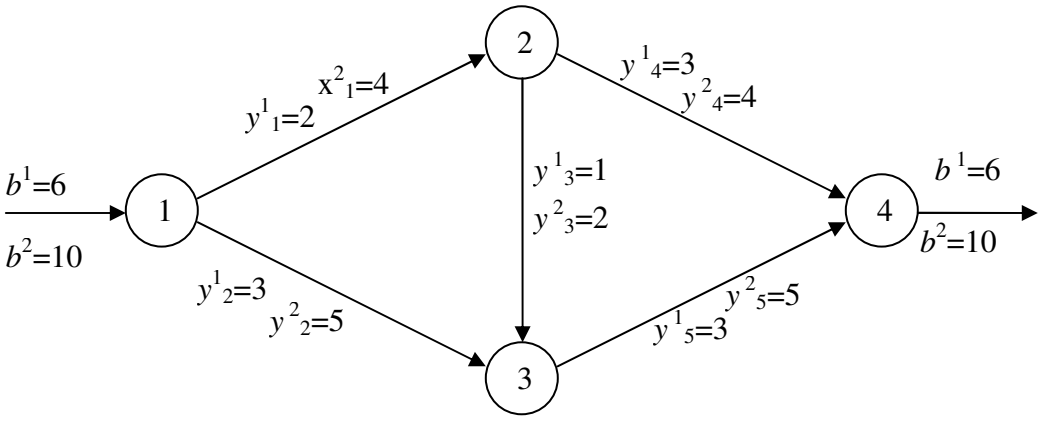


Figure IV.5. GLS Optimization Example with 2-Commodity Arc Flow Observations

a) For commodity 1

$$\mathbf{V}^1 = \begin{pmatrix} 1 & -.5 & 0 & 0 & 0 \\ -.5 & 1 & 0 & 0 & 0 \\ 0 & 0 & .75 & 0 & 0 \\ 0 & 0 & 0 & .5 & -.2 \\ 0 & 0 & 0 & -.2 & .5 \end{pmatrix}$$

$$\mathbf{A}^1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{pmatrix} \qquad \mathbf{A_r}^1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \end{pmatrix}$$

$$\mathbf{b}^1 = \begin{pmatrix} 6 \\ 0 \\ 0 \\ 6 \end{pmatrix} \qquad \mathbf{b_r}^1 = \begin{pmatrix} 6 \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{x}^1 = \begin{pmatrix} 2 \\ 3 \\ 1 \\ 3 \\ 3 \end{pmatrix}$$

b) For commodity 2

$$\mathbf{V}^2 = \begin{pmatrix} 1 & -.5 & 0 & 0 & 0 \\ -.5 & 1 & 0 & 0 & 0 \\ 0 & 0 & .75 & 0 & 0 \\ 0 & 0 & 0 & .5 & -.2 \\ 0 & 0 & 0 & -.2 & .5 \end{pmatrix}$$

$$\mathbf{A}^2 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{pmatrix} \qquad \mathbf{A_r}^2 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \end{pmatrix}$$

$$\mathbf{b}^2 = \begin{bmatrix} 10 \\ 0 \\ 0 \\ 10 \end{bmatrix} \qquad \mathbf{b_r}^2 = \begin{bmatrix} 10 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{x}^2 = \begin{bmatrix} 4 \\ 5 \\ 2 \\ 4 \\ 5 \end{bmatrix}$$

$$\mathbf{x} = [\mathbf{I} - \mathbf{V}\mathbf{A_r}'(\mathbf{A_r}\mathbf{V}\,\mathbf{A_r}')^{-1}\,\mathbf{A_r}]\mathbf{y} + \mathbf{V}\mathbf{A_r}'(\mathbf{A_r}\mathbf{V}\,\mathbf{A_r}')^{-1}\mathbf{b_r}$$

$$= [5.3108 \;\; 4.6892 \;\; 1.1892 \;\; 4.1216 \;\; 5.8784$$
$$3.1081 \;\; 2.8919 \;\; 0.3919 \;\; 2.7162 \;\; 3.2838]'$$

In the above solution (**x**), the final traffic count estimates satisfy the network flow conservation constraints. For example, an incoming flow estimate (x^1_1), of node 2 is the sum of the outgoing flows, x^1_3 and x^1_4 (3.1081 = 0.3919 + 2.7162).

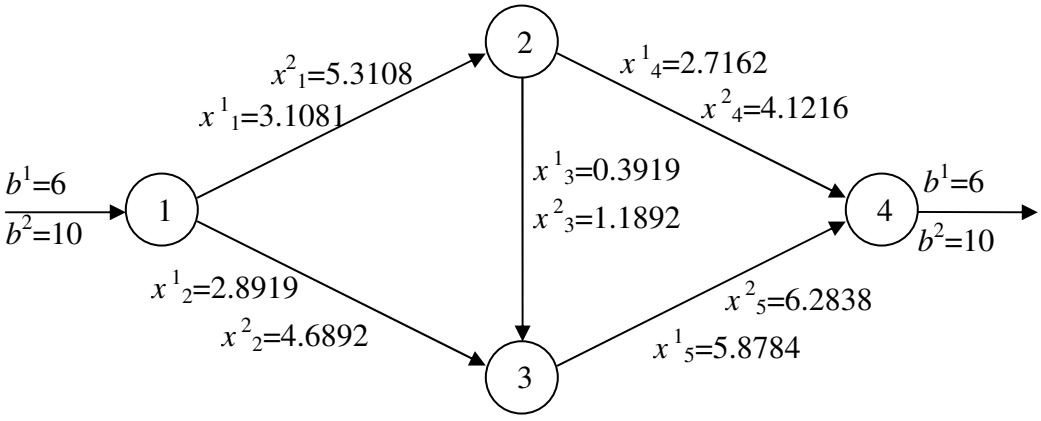


Figure IV.6. Traffic Count Estimates of the GLS Example IV.1

Figure IV.6. shows the solution of the GLS optimization problem. The link flows are adjusted by the given variance-covariance matrix in order to minimize the total deviation between the original flows and the estimates.

CHAPTER V

COMPUTERIZATION AND APPLICATIONS

V.1. Introduction

The proposed procedures have been computerized and run under various scenarios to test the performance of the proposed methodology. Such scenarios include the changes in number of commodities (vehicle classes), different variance-covariance matrices between network link flows, and various cost functions by changing the parameters corresponding to different road conditions. All procedures are written in MATLAB language and run on a personal computer with Intel Pentium IV 3.06 processor. In the following sections, the computer implementation of the proposed methodology is described.

V.2. Illustration of Real Field Application

A metro area was randomly selected to show the computational procedure of the methodology in real highway network. In Figure V.1, the circled area is considered as a specific region for which traffic counts are going to be estimated based on partial traffic counts. It is assumed that the network has 6 origins and 8 destinations per each commodity, total 96 O-D pairs.

In the map, the thicker lines (red in color) represent the links without traffic count data and the thin lines (gray in color), the links with traffic count observations.

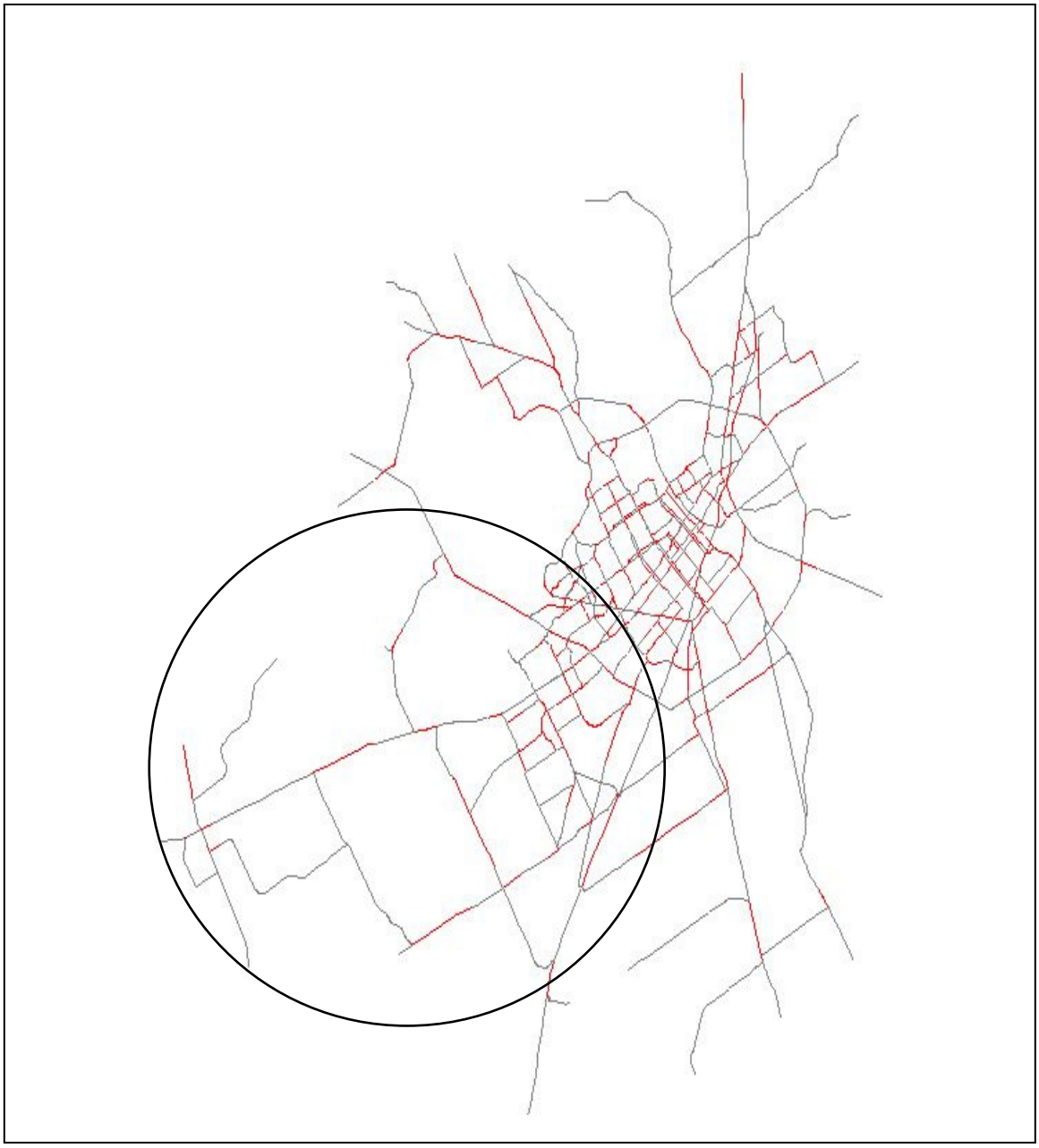


Figure V.1. Map of a Metro Area Highway Network

V.2.1. Constructing a network modeling

A network model is constructed based on the highway network shown in Figure V.1. In the map each intersection becomes a node. Also, a starting or an ending point of an unobserved link becomes a node. Figure V.2 shows the network model of the circled area in the above map. The traffic demands are assumed to have 6 origins and 8 destinations as shown in Figure V.2. In order to satisfy the traffic demands between the 6 origins and the 8 destinations, some links need to be bi-directional. In that case, the nodes and the links are divided into two separate nodes and links in order to consider inbound and outbound flows separately. In Figure V.2, dotted lines represents links without counts. The different thickness of the links also represents different road capacity, which is explained in "*Link Cost Flow*" subsection below. The node numbers and the link numbers are randomly assigned and do not follow any specific sequential scheme.

V.2.2. Origin-destination demand matrix

For simplicity of the example, it is assumed that incoming flows toward downtown area originate from the suburban areas. That is, the six origins, *node 1*, *node 2*, *node 10*, *node 13*, *node 28*, and *node 42*, are located in south-west side of the map and the eight destinations, *node 58*, *node 59*, *node 60*, *node 64*, *node 65*, *node 66*, *node 68*, and *node 9*, in north-east side of the map shown in Figure V.2.

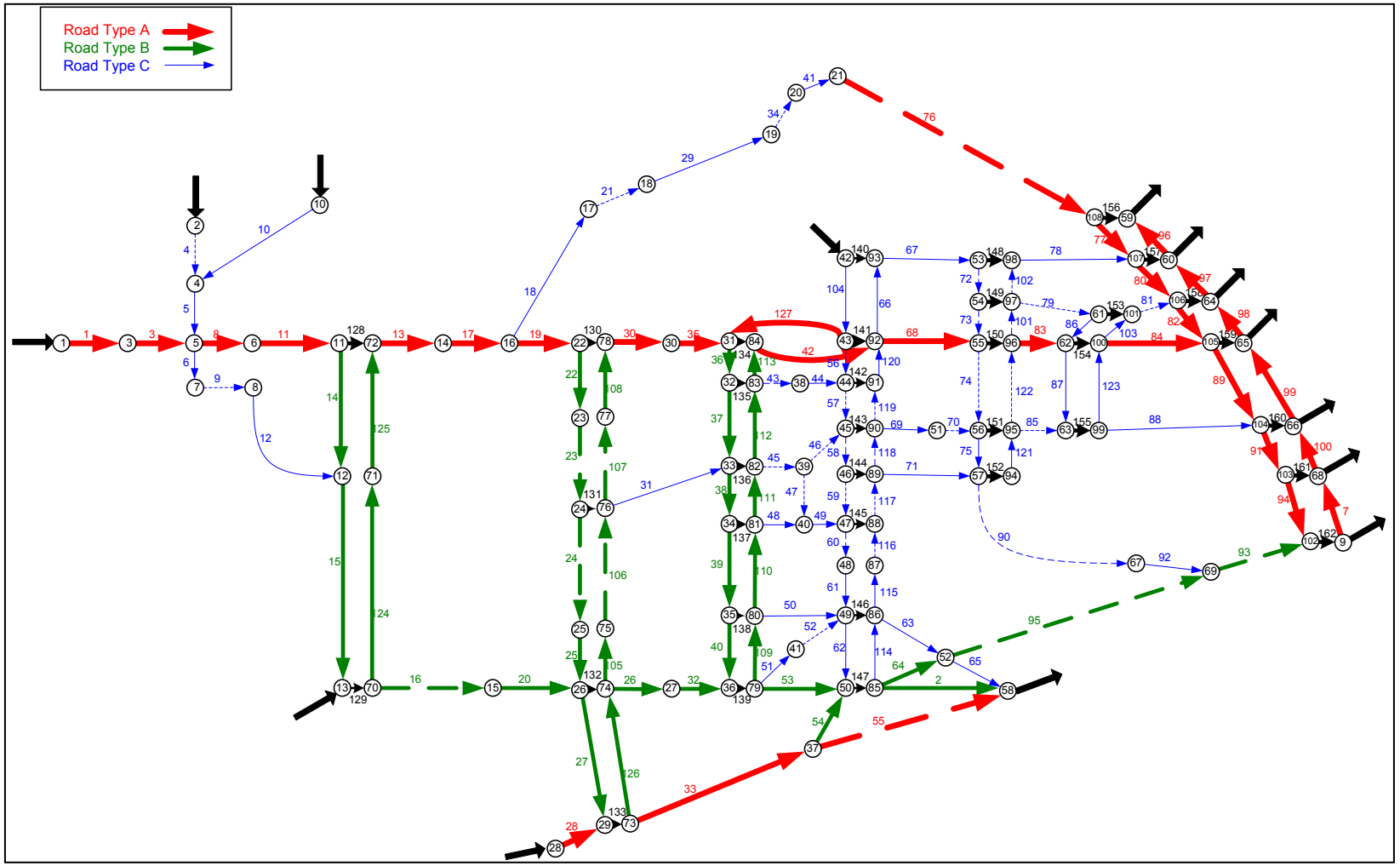


Figure V.2. Network Model of the Selected Area in Figure V.1

The demands are given in Table V.1. The demands in Table V.1 are assumed for two commodities.

Table V.1. O-D Demand Matrix

	O \ D	58	59	60	64	65	66	68	9	Total
Comm. 1	1	100	50	70	150	200	170	160	180	1080
	2	40	20	25	50	70	60	60	75	400
	10	50	20	30	80	100	90	80	80	530
	13	60	30	40	90	120	100	90	100	630
	28	280	70	90	180	250	210	240	260	1580
	42	80	50	70	100	130	120	110	120	780
	Total	610	240	325	650	870	750	740	815	5000
Comm. 2	1	75	37	53	110	145	120	116	130	786
	2	30	15	18	30	50	45	45	55	288
	10	35	15	20	60	75	63	60	60	388
	13	45	21	30	60	90	75	62	75	458
	28	200	50	61	135	182	158	180	189	1155
	42	60	32	50	72	91	90	82	90	567
	Total	445	170	232	467	633	551	545	599	3642

V.2.3. Observed link flows

Table V.2 and Table V.3 show the observed link counts for each commodity. It is assumed that all count data are directional data. Also, there could be a sizable difference between the inbound count and the outbound count of a link. According to the map shown in Figure V.1, 107 links are assumed to have link counts. Like O-D demand matrix, link counts are randomly generated numbers and assumed to be given as input values of the traffic count estimation problem.

Table V.2. Observed Link Flows for Commodity 1

Link No	Counts	Link No	Counts	Link No	Counts	Link No	Counts
1	1050	44	454	99	571	137	178
2	80	48	276	103	250	138	40
3	1030	49	291	104	240	139	853
5	900	50	99	105	619	140	528
6	520	51	237	108	250	141	140
10	520	53	194	109	402	142	320
11	1440	54	627	110	332	143	478
12	525	56	81	112	220	144	100
14	6	61	76	113	4	145	309
15	537	62	81	114	268	146	338
17	1548	63	301	115	310	147	900
18	573	64	548	120	400	148	318
20	1048	65	2	121	386	149	250
22	141	66	58	123	140	150	1139
25	3	67	598	124	109	151	580
27	210	68	1146	125	108	152	378
29	572	69	579	126	617	153	434
30	1083	71	562	127	1	154	1164
31	515	75	9	128	1438	155	628
32	845	77	483	129	1171	156	89
33	1139	78	476	130	838	157	386
36	410	86	52	131	136	158	707
37	160	87	51	132	853	159	300
39	41	88	480	133	1779	160	1176
40	9	89	249	134	680	161	374
41	572	92	188	135	229	162	1300
42	678	96	150	136	430		

Table V.3. Observed Link Flows for Commodity 2

Link No	Counts	Link No	Counts	Link No	Counts	Link No	Counts
1	788	44	341	99	428	137	134
2	60	48	207	103	188	138	30
3	773	49	218	104	180	139	640
5	675	50	74	105	464	140	396
6	390	51	178	108	188	141	105
10	390	53	0	109	302	142	240
11	1080	54	146	110	249	143	359
12	394	56	61	112	165	144	75
14	5	61	57	113	3	145	232
15	403	62	61	114	201	146	254
17	1161	63	226	115	233	147	675
18	430	64	411	120	300	148	239
20	786	65	2	121	290	149	188
22	106	66	44	123	105	150	854
25	2	67	449	124	82	151	435
27	158	68	860	125	81	152	284
29	429	69	434	126	463	153	326
30	812	71	422	127	1	154	873
31	386	75	9	128	1079	155	471
32	634	77	362	129	878	156	67
33	854	78	357	130	629	157	290
36	308	86	39	131	90	158	530
37	120	87	38	132	600	159	225
39	31	88	360	133	1334	160	882
40	7	89	187	134	510	161	281
41	429	92	141	135	172	162	975
42	509	96	113	136	323		

V.2.4. Link cost functions

This example uses common BPR cost function with link parameters shown in Table V.4 and Table V.5. In order to represent different classes of roads, three different types of roads, which have u_a values of 30, 45 and 60, are assumed. In Figure V.2, the thickest line corresponds to road type A, which has largest capacity of 60, and the thinnest line to road type C, which has smallest capacity of 30.

Table V.4. Link Time Parameters for Commodity 1

Link No	$C_a(0)^*$	U_a^{**}	Link No	$C_a(0)$	U_a	Link No	$C_a(0)$	U_a
1	3	60	55	8	60	109	7	45
2	6	45	56	2	30	110	7	45
3	4	60	57	3	30	111	4	45
4	6	30	58	4	30	112	5	45
5	4	30	59	2	30	113	2	45
6	4	30	60	3	30	114	7	30
7	3	60	61	5	30	115	5	30
8	2	60	62	7	30	116	3	30
9	3	30	63	6	30	117	2	30
10	24	30	64	5	45	118	4	30
11	12	60	65	4	30	119	3	30
12	24	30	66	8	30	120	2	30
13	7	60	67	8	30	121	4	30
14	8	45	68	4	60	122	5	30
15	13	45	69	2	30	123	5	30
16	8	45	70	4	30	124	13	45
17	4	60	71	6	30	125	8	45
18	10	30	72	4	30	126	8	45
19	3	60	73	4	30	127	11	45
20	5	45	74	5	30	128	0	n/a
21	4	30	75	4	30	129	0	n/a
22	7	45	76	10	60	130	0	n/a
23	5	45	77	3	60	131	0	n/a
24	7	45	78	8	30	132	0	n/a
25	3	45	79	3	30	133	0	n/a
26	4	45	80	3	60	134	0	n/a
27	8	45	81	1	30	135	0	n/a
28	4	60	82	3	60	136	0	n/a

Table V.4. Continued

Link No	$C_a(0)^*$	U_a^{**}	Link No	$C_a(0)$	U_a	Link No	$C_a(0)$	U_a
29	6	30	83	2	60	137	0	n/a
30	6	60	84	3	60	138	0	n/a
31	10	30	85	3	30	139	0	n/a
32	6	45	86	4	30	140	0	n/a
33	7	60	87	5	30	141	0	n/a
34	3	30	88	4	30	142	0	n/a
35	1	60	89	2	60	143	0	n/a
36	2	45	90	7	30	144	0	n/a
37	5	45	91	2	60	145	0	n/a
38	4	45	92	4	30	146	0	n/a
39	7	45	93	3	45	147	0	n/a
40	7	45	94	3	60	148	0	n/a
41	2	30	95	10	45	149	0	n/a
42	11	45	96	3	60	150	0	n/a
43	4	30	97	3	60	151	0	n/a
44	4	30	98	3	60	152	0	n/a
45	6	30	99	2	60	153	0	n/a
46	4	30	100	2	60	154	0	n/a
47	4	30	101	4	30	155	0	n/a
48	7	30	102	4	30	156	0	n/a
49	2	30	103	4	30	157	0	n/a
50	8	30	104	8	30	158	0	n/a
51	6	30	105	3	45	159	0	n/a
52	6	30	106	7	45	160	0	n/a
53	7	45	107	5	45	161	0	n/a
54	7	45	108	7	45	162	0	n/a

* $C_a(0)$: Free Flow travel cost on link *a*
** U_a : Level of service of link *a*

Table V.5. Link Time Parameters for Commodity 2

Link No	$C_a(0)^*$	U_a^{**}	Link No	$C_a(0)$	U_a	Link No	$C_a(0)$	U_a
1	6	48	55	15	48	109	13	36
2	11	36	56	4	24	110	13	36
3	8	48	57	6	24	111	8	36
4	11	24	58	8	24	112	10	36
5	8	24	59	4	24	113	4	36
6	8	24	60	6	24	114	13	24
7	6	48	61	10	24	115	10	24
8	4	48	62	13	24	116	6	24
9	6	24	63	11	24	117	4	24

Table V.5. Continued

Link No	$C_a(0)^*$	U_a^{**}	Link No	$C_a(0)$	U_a	Link No	$C_a(0)$	U_a
10	46	24	64	10	36	118	8	24
11	23	48	65	8	24	119	6	24
12	46	24	66	15	24	120	4	24
13	13	48	67	15	24	121	8	24
14	15	36	68	8	48	122	10	24
15	25	36	69	4	24	123	10	24
16	15	36	70	8	24	124	25	36
17	8	48	71	11	24	125	15	36
18	19	24	72	8	24	126	15	36
19	6	48	73	8	24	127	21	36
20	10	36	74	10	24	128	0	n/a
21	8	24	75	8	24	129	0	n/a
22	13	36	76	19	48	130	0	n/a
23	10	36	77	6	48	131	0	n/a
24	13	36	78	15	24	132	0	n/a
25	6	36	79	6	24	133	0	n/a
26	8	36	80	6	48	134	0	n/a
27	15	36	81	2	24	135	0	n/a
28	8	48	82	6	48	136	0	n/a
29	11	24	83	4	48	137	0	n/a
30	11	48	84	6	48	138	0	n/a
31	19	24	85	6	24	139	0	n/a
32	11	36	86	8	24	140	0	n/a
33	13	48	87	10	24	141	0	n/a
34	6	24	88	8	24	142	0	n/a
35	2	48	89	4	48	143	0	n/a
36	4	36	90	13	24	144	0	n/a
37	10	36	91	4	48	145	0	n/a
38	8	36	92	8	24	146	0	n/a
39	13	36	93	6	36	147	0	n/a
40	13	36	94	6	48	148	0	n/a
41	4	24	95	19	36	149	0	n/a
42	21	36	96	6	48	150	0	n/a
43	8	24	97	6	48	151	0	n/a
44	8	24	98	6	48	152	0	n/a
45	11	24	99	4	48	153	0	n/a
46	8	24	100	4	48	154	0	n/a
47	8	24	101	8	24	155	0	n/a
48	13	24	102	8	24	156	0	n/a
49	4	24	103	8	24	157	0	n/a
50	15	24	104	15	24	158	0	n/a

Table V.5. Continued

Link No	$C_a(0)^*$	U_a^{**}	Link No	$C_a(0)$	U_a	Link No	$C_a(0)$	U_a
51	11	24	105	6	36	159	0	n/a
52	11	24	106	13	36	160	0	n/a
53	13	36	107	10	36	161	0	n/a
54	13	36	108	13	36	162	0	n/a

* $C_a(0)$: Free Flow travel cost on link *a*
** U_a : Level of service of link *a*

Regarding the link numbers from 128 to 162, link travel times are assumed to be zero since the links are imaginary links and do not reflect actual roads in the original road network.

V.2.5. Connectivity list of the network model

In order to keep track of the connections between the nodes and the links, connectivity lists are used in the model.

List 1: | - - 1 2 10 3 4 5 …… 21 |

Node 1 2 3 4 4 5 5 6 ……108

List 2: | 0 0 1 2 2 1 …… 1 |

Node 1 2 3 4 5 6 ……108

The first list keeps the node numbers at the beginning of directed arcs going into nodes *1,2,3,...,108*. The second list records the number of directed arcs into nodes

1,2,3,...,108. For example, the fourth and the fifth value of 2 and 10 in list 1 represent node 4 having two incoming links from node 2 and node 10.

V.2.6. Multi-commodity user equilibrium traffic assignment problem

The vehicle count assignment for those links without actual count data is carried out by using a traffic assignment problem based on Wardrop's *user equilibrium* condition. Following formulation uses time functions and parameters defined in Table V.4 and Table 5 in the objective function.

$$Min \quad f(x) = \int_0^{x_1^1} c_1^1(v)dv + ... + \int_0^{x_{127}^2} c_{127}^2(v)dv$$

$$= 3\left[1 + 0.15\left(\frac{x_1^1}{60}\right)^4\right] + ... + 21\left[1 + 0.15\left(\frac{x_{127}^2}{36}\right)^4\right]$$

$$s.t. \quad x_1^1 = h_1^1 + h_2^1 + \cdots + h_{57}^1 + h_{58}^1$$

$$x_2^1 = h_2^1 + h_4^1 + h_6^1 + h_8^1 + h_{10}^1 + h_{11}^1 + h_{12}^1 + h_{13}^1 + h_{60}^1 + h_{62}^1 + h_{64}^1 + h_{66}^1 + h_{68}^1 + h_{69}^1 + h_{70}^1 + h_{71}^1 + h_{118}^1 + h_{120}^1 + h_{122}^1 + h_{124}^1 + h_{126}^1 + \cdots + h_{129}^1 + h_{175}^1 + h_{755}^1 + h_{1407}^1 + h_{1408}^1$$

$$x_3^1 = x_1^1$$

$$x_4^1 = h_{59}^1 + \cdots + h_{116}^1$$

$$\vdots$$

$$x_1^2 = h_1^2 + h_2^2 + \cdots + h_{57}^2 + h_{58}^2$$

$$x_2^2 = h_2^2 + h_4^2 + h_6^2 + h_8^2 + h_{10}^2 + h_{11}^2 + h_{12}^2 + h_{13}^2 + h_{60}^2 + h_{62}^2 + h_{64}^2 + h_{66}^2 + h_{68}^2 + h_{69}^2 + h_{70}^2 + h_{71}^2 + h_{118}^2 + h_{120}^2 + h_{122}^2 + h_{124}^2 + h_{126}^2 + \cdots + h_{129}^2 + h_{175}^2 + h_{755}^2 + h_{1407}^2 + h_{1408}^2$$

$$\vdots$$

$$\vdots$$

$$x_i \geq 0 \ , \ h_i \geq 0$$

Table V.6 and Table V.7 shows the output of the computer program which solves the problem above. Since the assignment is generated from the pre-given O-D demands and the time functions, the accuracy of the assignment relies on them. If a link seems to have an unreasonable assignment, it is beneficial to review the corresponding time parameters and O-D demands at this stage and adjust them as necessary.

Table V.6. User Equilibrium Link Flows for Commodity 1

Link No	Counts	Link No	Counts	Link No	Counts	Link No	Counts
1	1080	42	704	83	1205	124	252
2	138	43	484	84	1087	125	252
3	1080	44	484	85	593	126	624
4	400	45	456	86	112	127	44
5	930	46	456	87	121	128	1290
6	579	47	0	88	515	129	1350
7	651	48	307	89	373	130	822
8	1431	49	307	90	219	131	120
9	579	50	131	91	856	132	924
10	530	51	273	92	219	133	1790
11	1431	52	273	93	1061	134	645
12	579	53	179	94	405	135	281
13	1541	54	763	95	842	136	420
14	141	55	403	96	176	137	187
15	720	56	89	97	214	138	105
16	1099	57	252	98	213	139	896
17	1541	58	213	99	731	140	457
18	565	59	54	100	362	141	190
19	977	60	72	101	425	142	320
20	1099	61	72	102	206	143	495
21	565	62	106	103	308	144	158
22	155	63	320	104	323	145	289
23	155	64	590	105	685	146	371

Table V.6. Continued

Link No	Counts	Link No	Counts	Link No	Counts	Link No	Counts
24	35	65	69	106	685	147	1047
25	35	66	108	107	291	148	262
26	863	67	565	108	291	149	279
27	210	68	1268	109	443	150	1248
28	1580	69	582	110	417	151	545
29	565	70	582	111	297	152	430
30	1113	71	569	112	261	153	387
31	514	72	303	113	59	154	1196
32	863	73	24	114	319	155	714
33	1166	74	43	115	369	156	64
34	565	75	81	116	369	157	287
35	1113	76	565	117	658	158	652
36	511	77	501	118	248	159	352
37	230	78	468	119	161	160	1118
38	324	79	498	120	481	161	451
39	137	80	682	121	430	162	1466
40	32	81	694	122	382		
41	565	82	725	123	199		

Table V.7. User Equilibrium Link Flows for Commodity 2

Link No	Counts	Link No	Counts	Link No	Counts	Link No	Counts
1	786	42	520	83	878	124	185
2	100	43	348	84	795	125	185
3	786	44	348	85	427	126	462
4	288	45	337	86	89	127	39
5	676	46	337	87	71	128	945
6	414	47	0	88	370	129	975
7	461	48	230	89	286	130	592

Table V.7. Continued

Link No	Counts	Link No	Counts	Link No	Counts	Link No	Counts
8	1048	49	230	90	158	131	97
9	414	50	92	91	636	132	658
10	388	51	188	92	158	133	1312
11	1048	52	188	93	772	134	459
12	414	53	134	94	288	135	184
13	1130	54	555	95	614	136	334
14	103	55	295	96	123	137	146
15	517	56	61	97	192	138	78
16	790	57	214	98	129	139	648
17	1130	58	172	99	527	140	323
18	417	59	34	100	264	141	144
19	713	60	47	101	306	142	195
20	790	61	47	102	172	143	378
21	417	62	72	103	230	144	138
22	121	63	241	104	244	145	218
23	121	64	424	105	501	146	255
24	25	65	51	106	501	147	760
25	25	66	91	107	221	148	173
26	620	67	415	108	221	149	224
27	157	68	925	109	326	150	910
28	1155	69	419	110	312	151	392
29	417	70	419	111	227	152	309
30	814	71	408	112	224	153	269
31	376	72	241	113	61	154	896
32	620	73	18	114	237	155	498
33	849	74	32	115	251	156	47
34	417	75	59	116	251	157	162
35	814	76	417	117	469	158	530
36	393	77	369	118	198	159	235
37	210	78	346	119	158	160	814

Table V.7. Continued

Link No	Counts	Link No	Counts	Link No	Counts	Link No	Counts
38	252	79	357	120	353	161	348
39	105	80	552	121	309	162	1060
40	28	81	498	122	274		
41	417	82	521	123	129		

V.2.7. Generalized least squares (GLS) optimization problem

Up to this point, the observed counts were not directly used in traffic assignments. Furthermore, the user equilibrium traffic assignments shown in Table V.6 and Table V.7 do not agree with the observed counts shown in Table V2. and Table V3. It is because the observed counts are usually involved with observation errors and the traffic assignments are also estimates based on limited information such as time functions and O-D demand matrix. Therefore, it is desirable to adjust the current user equilibrium solution using observed link counts. The adjustment is done by solving GLS optimization problem, which is formulated in Chapter III. The following sections explain variance-covariance matrix and node-link incident matrix used in GLS optimization problem.

V.2.8. Variance-covariance matrix between links

A variance is a measure of the amount of variability inherent in observing the flow of a link. There may also be a certain relationship between two link counts, which will incur covariance between the two links. The following variance-covariance matrix, **(V)** is assumed to be given for 162 links to solve the GLS optimization problem. Each row (or column) shows the relationship between the link and all other links in the network.

Below is the matrix showing the variances and covariances between first five link flows of the network.

$$\mathbf{V} = \begin{pmatrix} 1 & 0 & 0 & 0.8 & 0 & \dots \\ 0 & 1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & 0.4 & \dots \\ 0.8 & 0 & 0 & 1 & 0 & \dots \\ 0 & 0 & 0.4 & 0 & 1 & \dots \end{pmatrix}$$

V.2.9. Final traffic flow estimates

Table V.8 and Table V.9 shows the final traffic count estimates, which satisfy the network flow conservation condition.

Table V.8. Final Traffic Estimates for Commodity 1

Link No	Estimates	Link No	Estimates	Link No	Estimates	Link No	Estimates
1	1080	42	579	83	689	124	108
2	103	43	690	84	1183	125	108
3	1080	44	469	85	1095	126	639
4	400	45	469	86	593	127	2
5	930	46	467	87	53	128	1457
6	553	47	452	88	50	129	1183
7	606	48	15	89	496	130	838
8	1457	49	286	90	318	131	127
9	553	50	302	91	198	132	882
10	530	51	112	92	804	133	1794
11	1457	52	263	93	198	134	686

Table V.8. Continued

Link No	Estimates	Link No	Estimates	Link No	Estimates	Link No	Estimates
12	553	53	263	94	1042	135	242
13	1565	54	200	95	379	136	418
14	0	55	661	96	844	137	223
15	553	56	493	97	170	138	53
16	1075	57	87	98	175	139	866
17	1565	58	255	99	147	140	547
18	579	59	216	100	646	141	145
19	986	60	81	101	291	142	301
20	1075	61	64	102	403	143	492
21	579	62	64	103	180	144	135
22	148	63	72	104	238	145	318
23	148	64	302	105	233	146	368
24	21	65	555	106	662	147	933
25	21	66	14	107	662	148	316
26	859	67	73	108	263	149	274
27	214	68	619	109	263	150	1181
28	0	69	1187	110	404	151	607
29	579	70	591	111	344	152	390
30	1101	71	591	112	281	153	444
31	527	72	569	113	231	154	1185
32	859	73	303	114	4	155	643
33	1155	74	29	115	274	156	70
34	579	75	35	116	339	157	320
35	1101	76	19	117	339	158	678
36	416	77	579	118	658	159	372
37	174	78	509	119	224	160	1104
38	283	79	496	120	124	161	425
39	60	80	497	121	425	162	1421
40	7	81	685	122	390		
41	1080	82	682	123	404		

Table V.9. Final Traffic Estimates for Commodity 2

Link No	Estimates	Link No	Estimates	Link No	Estimates	Link No	Estimates
1	786	42	346	83	868	124	83
2	79	43	346	84	803	125	83
3	786	44	343	85	434	126	485
4	288	45	331	86	42	127	2
5	676	46	12	87	38	128	1060
6	398	47	210	88	367	129	860
7	438	48	222	89	243	130	616
8	1064	49	80	90	141	131	89
9	398	50	187	91	596	132	625
10	388	51	187	92	141	133	1327
11	1064	52	146	93	754	134	505
12	398	53	492	94	283	135	173
13	1143	54	350	95	613	136	302
14	3	55	66	96	134	137	165
15	402	56	207	97	149	138	35
16	777	57	169	98	99	139	636
17	1143	58	59	99	472	140	396
18	419	59	47	100	206	141	103
19	725	60	47	101	296	142	205
20	777	61	53	102	147	143	369
21	419	62	221	103	173	144	110
22	109	63	408	104	171	145	234
23	109	64	16	105	487	146	262
24	20	65	57	106	487	147	690
25	20	66	453	107	196	148	221
26	622	67	876	108	196	149	205
27	172	68	426	109	304	150	873
28	0	69	426	110	258	151	440
29	419	70	410	111	213	152	285
30	812	71	232	112	172	153	312

Table V.9. Continued

Link No	Estimates	Link No	Estimates	Link No	Estimates	Link No	Estimates
31	381	72	26	113	0	154	872
32	622	73	29	114	203	155	471
33	842	74	16	115	243	156	36
34	419	75	419	116	243	157	217
35	812	76	383	117	477	158	517
36	308	77	369	118	177	159	260
37	135	78	354	119	120	160	817
38	214	79	535	120	325	161	314
39	48	80	485	121	285	162	1037
40	14	81	503	122	291		
41	419	82	346	123	104		

Notice that the figures in Table V.6 and Table V.8 are similar in this example. There are two major reasons for this similarity. First, the variance-covariance matrix was assumed to have a simple form, whose main diagonal has the value of one, the rest of the elements are mostly zero. Second, some link flows are considered fixed constants under the given network structure.

CHAPTER VI

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

VI.1. Summary and Contribution

The purpose of this dissertation is to develop a model and solution methodology for estimating multi-commodity traffic flows in a transportation network having traffic counts on selected links. It is assumed that there are known, fixed demands between multiple origins and destinations. Considering congestion effects, a link cost is not a fixed value but rather a function of link traffic volume. Moreover, it is assumed separable with respect to traffic links. Travelers' are considered to choose the minimum cost route at the beginning of their travel without considering other travelers. Based on the assumptions above a MDUE assignment is formulated as a nonlinear programming problem with linear multi-commodity network constraints, and solved by a linear approximation method using the Frank-Wolfe algorithm. The Frank-Wolfe algorithm is a *conditional gradient method* that uses the gradient of the original objective function conditioned at any given iteration point in determination of a search direction. Due to the slow convergence, the Frank-Wolfe algorithm is modified by a heuristic method using different step size during iteration.

At the second stage of the proposed approach, a GLS optimization problem is modeled to consider actual traffic count observations and the user equilibrium solution from the assignment problem simultaneously to find final traffic count estimates for all highway links from the given variance-covariance matrix between the links. The final traffic count estimates are determined to minimize the deviations between traffic counts. The problem is a quadratic programming equation with network flow conservation constraints and is solved by a Lagrangian relaxation method.

The proposed model is developed to utilize a network flow optimization and a statistical analysis method. In addition to this model, several contributions are made; first, multiple vehicle classes (multi-commodities) are explicitly considered in the model. In general, traffic counts are collected and analyzed regardless of the vehicle classes. Second, a computationally efficient solution procedure is devised. Due to the multiplicative property of multi-commodity flow problem, actual application of the algorithm could be restricted by its size. Proposed approach uses the modified Frank-Wolfe algorithm, which has a fast convergence yet does not require the enumeration of all possible paths between origins and destinations, in the solution procedure.

The proposed algorithm is implemented in the MATLAB language on a personal computer equipped with an Intel Pentium IV 3.06 GHz processor. Tests for the proposed solution methodology were performed with fictitious scenarios on a real highway network.

VI.2. Conclusions

In this dissertation, a model and solution algorithm is developed to obtain multi-commodity traffic flow estimates from the given traffic counts on selected highway links. Most traffic count estimation methods use a traffic assignment approach or a statistical methodology, such as the regression analysis in determination of traffic link flows. Traffic assignment approaches calculate traffic link flows based on the given link cost function and origin-destination traffic demands. The limiting factor of the approaches is that they do not utilize additional information from traffic count observations in their procedures. On the other hand, most statistical estimation procedures rely only on observed traffic counts and are difficult to produce estimates on links without observational data. They also fail to satisfy the flow conservation condition, which is a plausible assumption in transportation network modeling.

The proposed algorithm utilizes the data mentioned in the approaches mentioned above by solving a MDUE assignment problem and a GLS optimization problem in

sequence. The MDUE assignment problem is solved by the modified Frank-Wolfe algorithm. Then, the GLS optimization problem is solved by Lagrangian relaxation method.

In the experiments testing the algorithm, direct comparison with other traffic count estimation algorithm is not possible, since there is no corresponding algorithm considering multi-commodity traffic flows. Also, it is regretful that real data was not available to test this algorithm in spite of the efforts to receive inputs from the real field. However, it can be concluded that there are improvements and changes by this algorithm. The modified Frank-Wolfe algorithm, compared to the original one, reduces the number of iterations by a range of 38 % - 94 %. The effect of the GLS optimization, which utilizes the traffic count observation and the relationship between traffic links, is observed by the changes in the final traffic count estimates from the user equilibrium.

VI.3. Recommendations for Further Research

Future research efforts to the extension of the proposed model and solution methodology are suggested on the following issues:

- *More efforts to elaborate the modified Frank-Wolfe algorithm*. In the proposed approach, the Frank-Wolfe algorithm is modified by heuristic method using different step sizes to rectify its slow convergence problem while approaching the optimal point. Even though the heuristic turned out to be effective with carefully selected step sizes, it needs further research on finding the appropriate step size.
- *Considering non separable link cost functions*. Current model assumes the link cost functions are separable. That is, the cost function of a link is only affected by the traffic flow on the link. However, in real highway network, a flow of a link could affect the cost of the other highway links. In this case, more sophisticated cost function needs to be formulated. Also, even in the separable cost function case, the fixed parameters in BPR link cost function may need to be adjusted to accommodate different link conditions in real field application.

- *Considering when the origin-destination traffic demands matrix is not available*. The O-D traffic demand matrix is a major input to the user equilibrium assignment in the proposed approach. Hence, the credibility of the given O-D traffic demands is very important to the resulting link flow assignments. However, there could be some cases when O-D traffic demand matrix is not available for all O-D pairs or is not accurate enough to be used as an input to the problem. In such cases, the constraints of the user equilibrium assignment are not valid. Therefore, another type of assignment methodology, such as stochastic assignment model, will be needed to find a user equilibrium solution.

REFERENCES

Bazaraa, Mokhtar S., Jarvis, John J., and Sherali, Hanif D. 1990. Linear Programming and Network Flows, John Wiley & Sons Inc., New York, NY.

Beckmann M. J., McGuire C.B., and Winsten C. B. 1956. Studies in the Economics of Transportation, Yale University Press, New Haven, CT.

Bell, M. G., Iida, Y. 1997. Transportation Network Analysis, John Wiley & Sons Inc., New York, NY.

Bell, M., Shield, C. M., Busch, F., and Kruse, G. 1997. A stochastic user equilibrium path flow estimator, Transportation Research C 5, 197-210.

Bureau of Public Roads 1964. Traffic Assignment Manual, U.S. Department of Commerce, Urban Planning Division, Washington, DC.

Cascetta, E. 1984. Estimation of trip matrices from traffic counts and survey data: A generalized least squares estimator. Transportation Research B 18, 289-299.

Cascetta E. 2001. Transportation Systems Engineering: Theory and Methods, Kluwer Academic Publishers, Boston, MA.

Chen, A., Jayakrishnan, R., and Tsai, W. K. 2002. Faster Frank-Wolfe traffic assignment with new flow update scheme. Journal of Transportation Engineering, 31-39.

Dafermos, S. C. 1972. The traffic assignment problem for multiclass-used transportation networks. Transportation Science 6, 73-87.

Federal Highway Administration 2002. Highway Statistics 2001. Federal Highway Administration, Washington, DC.

Fukushima, Masao 1984. A modified Frank-Wolfe algorithm for solving the traffic assignment problem. Transportation Research B 1, 169-177.

Gazis, D. and Liu, C. 2003. Kalman filtering estimation of traffic counts for two network links in tandem. Transportation Research B 37, 737-745.

Hearn, D. W., and Ribera, J. 1981. Convergence of the Frank-Wolfe method for certain bounded variable traffic assignment problems. Transportation Research B 15, 437-442.

Ivan, J. N., and Allaire, S. A. 2001. Regional and area-type modeling of peak spreading on Connecticut freeways. Journal of Transportation Engineering, 223-229.

LeBlanc, L. J., Morlol, E. K., and Pierskalla, W. P. 1975. An efficient approach to solving the road network equilibrium traffic assignment. Transportation Research 9, 309-318.

Marcotte, P., Wynter, L. 2004. A new look at the multiclass network equilibrium problem. Transportation Science Vol. 38, No. 3, 282-292.

Moon, T. K., and Stirling, W. C. 2000. Mathematical Methods and Algorithms for Signal Processing, Prentice Hall, Upper Saddle River, NJ.

Searle, Shayle R. 1982. Matrix Algebra Useful for Statistics, John Wiley & Sons Inc., New York, NY.

Sharma, S. C., Gulati, B. M., Rizak, S. N. 1996. Statewide traffic volume studies and precision of AADT estimates, Journal of Transportation Engineering, 430-439.

Shen, LD., Zhao, F. and Ospina, DI. 1999. Estimation of annual average daily traffic for off-system roads in Florida. Florida Department of Transportation, Tallahassee, FL.

Sherali, H., Sivanandan, R., and Hobeika, A.G. 1994. A linear programming approach for synthesizing origin-destination trip tables from link traffic volumes. Transportation Research B 28, 213-233.

Soroush, H. and Mirchandani, P. B. 1990. The stochastic multicommodity flow problem. Networks Vol. 20, 121-155.

Toint, P. L. 1997. Transportation modeling and operations research: A fruitful connection. Proceedings of the NATO Advanced Study Institute on Operations Research and Decision Aid Methodologies in Traffic and Transportation Management, 1-27.

Weintraub, A., Ortiz, C., and González, J. 1985. Accelerating convergence of the Frank-Wolfe algorithm. Transportation Research B 19, 113-122.

Wells, C.E. and Evans, J.R. 1989. Statistical procedure for estimating branch flows and total network flow. Networks Vol. 19, 481-491.

Yang, H. 1995. Heuristic algorithms for the bilevel origin-destination matrix estimation problem. Transportation Research B 29, 231-242.

Yang, H., Ilda, Y., and Sasaki, T. 1994. The equilibrium-based origin-destination matrix estimation problem. Transportation Research B 28, 23-33.

Yang, H., Sasaki, T., Ilda Y., and Asakura Y. 1992. Estimation of origin-destination matrices from link traffic counts on congested networks. Transportation Research B 26, 417-434.

APPENDIX A

FRANK-WOLFE ALGORITHM

Frank-Wolfe algorithm is widely used algorithm for traffic assignment problems as well as many other applications. The use of Frank-Wolfe algorithm for solving traffic assignment problem is especially beneficial because it does not need to enumerate all possible paths between origins and destinations which could be very cumbersome work as the size of network grows. Also, it allows using very efficient shortest path algorithm to solve the traffic assignment problem with nonlinear objective function which is inevitable to consider traffic congestion on highway links.

The algorithm is also called *conditional gradient method* since it is devised to use the gradient of the original objective function to determine a search direction. Lets consider the problem:

$$\text{Minimize} \quad f(x) \tag{A1}$$

$$\text{s.t.} \quad x \in F\,. \tag{A2}$$

where convex set F is a feasible region and where function f is continuously differentiable and is convex over F. Then, the minimization problem can be solved by linear approximation using the Taylor expansion of the function f. Detailed steps of the algorithm are as follows:

0. Choose a starting point (initial solution): x_o

Any feasible solution, $x_o \in F$, could be a starting point.

Set $k = 0$.

1. Find a search direction: d_k

The Frank-Wolfe algorithm uses a first-order Taylor expansion of the function f around x_k to determine a *feasible direction* which improves the current evaluation of the function. That is,

$$\text{Minimize} \quad g_k(y) = f(x_k) + \nabla f(x_k)'(y - x_k) \tag{A3}$$

$$\text{s.t.} \quad y \in F . \tag{A4}$$

After x_k is fixed as a constant, the above problem reduces to:

$$\text{Minimize} \quad g_k(y) = \nabla f(x_k)' y \tag{A5}$$

$$\text{s.t.} \quad y \in F . \tag{A6}$$

This is a linear programming problem. When the feasible region F is subject to network constraints the problem becomes a network flow optimization problem which could be solved more efficiently with specialized algorithm such as a shortest path algorithm. A *feasible direction* is $d_k = y_k - x_k$.

2. Calculate next iteration point: x_{k+1}

Next iteration point is determined by finding appropriate step length, α_k, which satisfies the following condition:

$$f(x_k + \alpha_k d_k) < f(x_k). \tag{A7}$$

This is a simple line search problem;

$$\text{Minimize} \quad f(x_k + \alpha d_k) \tag{A8}$$

$$\text{s.t.} \quad 0 \le \alpha \le 1. \tag{A9}$$

After finding α, next iteration point is:

$$x_{k+1} = x_k + \alpha d_k. \tag{A10}$$

3. Test a stopping criterion.

If a stopping criterion is satisfied, then stop with current iteration point x_{k+1} as the solution. Otherwise, let iteration counter $k = k + 1$ and go to step 1.

(**Example A.1**) In order to illustrate the Frank-Wolfe algorithm, let's consider the following minimization problem:

$$\text{Minimize } f(x) = 3x_1^2 - 24x_1 + 2x_2^4 - 64x_2 \tag{A11}$$

$$\text{s.t. } 5x_1 + 4x_2 \le 20 \tag{A12}$$

$$x_1 - x_2 \le 2 \tag{A13}$$

$$x_1 \ge 0,\ x_2 \ge 0. \tag{A14}$$

Figure A.1 shows the graphical representation of the above example. The shaded area indicates the feasible region and the dot represents the minimum point without constrains.

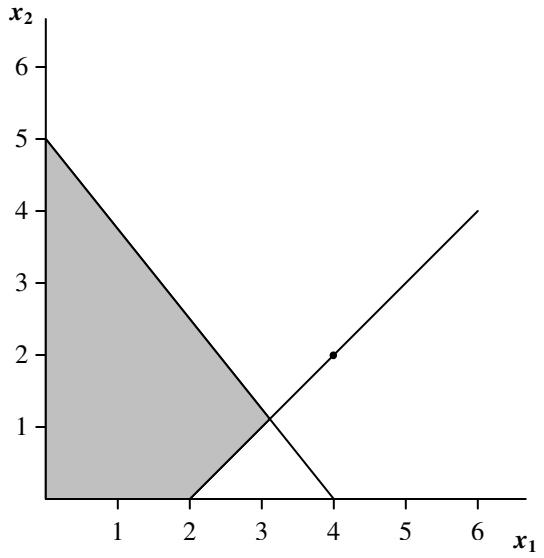


Figure A.1. Graph of a Frank-Wolfe Algorithm Example

From the partial derivatives of the objective function the unconstrained minimum of $f(x)$ is easily found at $x^* = (x_1, x_2) = (4, 2)$. However, the optimal point of the example, which is limited by the constraints (A10) – (A13), should be placed in the shaded area.

Now, let's follow the steps of the Frank-Wolfe algorithm to find the solution.

(**Iteration 1**)

0. Choose a starting point (initial solution): $\boldsymbol{x_0}$

Set $x = (x_1, x_2) = (0, 0)$.

Set $k = 0$.

1. Find a search direction: $\boldsymbol{d_0}$

Partial derivatives of the function f at $\boldsymbol{x}_0 = (0, 0)$ are:

$$\frac{\partial f}{\partial x_1} = 6x_1 - 24 = -24, \text{ and}$$

$$\frac{\partial f}{\partial x_2} = 8\,x_2^3 - 64 = -64.$$

Using the derivatives evaluated at $\boldsymbol{x}_0$ as cost coefficients solve the following minimization problem:

$$\begin{aligned} \text{Minimize } & g_0(y) = -24\,y_1 - 64\,y_2 \\ \text{s.t. } & 5y_1 + 4\,y_2 \le 20 \\ & y_1 - \;\; y_2 \le 2 \\ & y_1 \ge 0,\;\; y_2 \ge 0. \end{aligned}$$

The above LP minimization problem is easily solved and its solution is $\boldsymbol{y}_0 = (y_1, y_2) = (0, 5)$ as shown in **Figure A.2**.

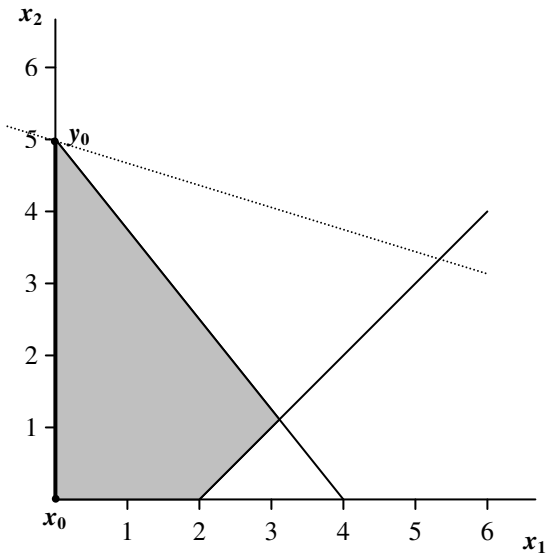


Figure A.2. Feasible Direction of Example A.1

A *feasible direction* is a vector $\boldsymbol{d}_0 = \boldsymbol{y}_0 - \boldsymbol{x}_0 = (0, 5) - (0, 0) = (0, 5)$. Next step is to find a minimum point between $\boldsymbol{x}_0$ and $\boldsymbol{y}_0$.

2. Calculate next iteration point: $\boldsymbol{x}_1$

Perform the line search to find a step size α which gives minimum value of the function f.

$$\begin{aligned} f(\boldsymbol{x}_0 + \alpha\, \boldsymbol{d}_0) &= f(0, 5\alpha) \\ &= 2(5\alpha)^4 - 64(5\alpha) \\ &= 1250\alpha^4 - 320\alpha \end{aligned}$$

Then,

$$\begin{aligned} &\text{Minimize } f(\boldsymbol{x}_0 + \alpha\, \boldsymbol{d}_0) = 1250\alpha^4 - 320\alpha \\ &\text{s.t. } 0 \le \alpha \le 1. \end{aligned}$$

The solution is $\alpha^* = 0.4$. Hence, by equation (A.10), the new iteration point is

$$\boldsymbol{x}_1 = (0, 0) + 0.4(0, 5) = (0, 2).$$

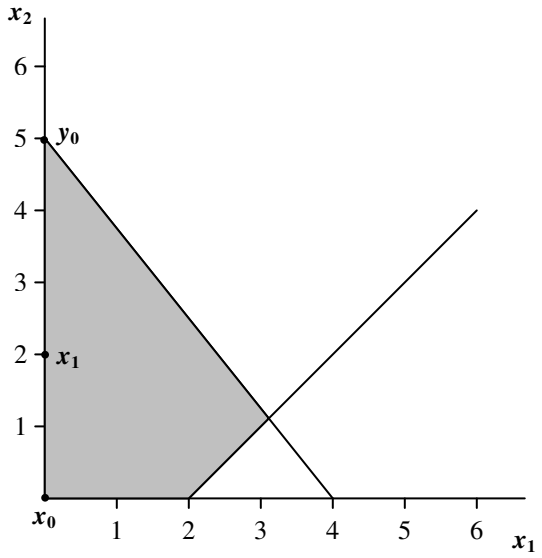


Figure A.3. New Iteration Point $\boldsymbol{x}_1$

3. Test a stopping criterion.

Calculate the following termination rule which do the ratio test on the relative improvement by new iteration point:

$$(f(\boldsymbol{x}_k) - f_k(\boldsymbol{x}_{k+1})) / | f_k(\boldsymbol{x}_{k+1})| \leq \varepsilon, \text{ where } \varepsilon \text{ is a small positive value.}$$

Then,

$$(f(\boldsymbol{x}_0) - f(\boldsymbol{x}_1)) / | f(\boldsymbol{x}_1)| = (0 - (-96))/|-96| = 1.$$

For the purpose of the example set $\varepsilon = 0.01$. Since the ratio is larger than $\varepsilon = 0.01$ continue the algorithm with the new iteration point $\boldsymbol{x}_1$ and let $k = k + 1 = 1$. Go to step 1.

(**Iteration 2**)

1. Find a search direction: $\boldsymbol{d}_1$

Partial derivatives of the function f at $\boldsymbol{x}_1 = (0, 2)$ are:

$$\frac{\partial f}{\partial x_1} = 6x_1 - 24 = -24, \text{ and}$$

$$\frac{\partial f}{\partial x_2} = 8\,x_2^3 - 64 = 0.$$

Using the derivatives evaluated at $\boldsymbol{x}_1$ as cost coefficients solve the following minimization problem:

$$\begin{aligned} \text{Minimize } & g_1(y) = -24\,y_1 - 0\,y_2 \\ \text{s.t. } & 5y_1 + 4\,y_2 \le 20 \\ & y_1 - \;\; y_2 \le 2 \\ & y_1 \ge 0,\;\; y_2 \ge 0. \end{aligned}$$

The above LP minimization problem is easily solved and its solution is $\boldsymbol{y}_1 = (y_1, y_2) = (3.111, 1.111)$ as shown in **Figure A.4**.

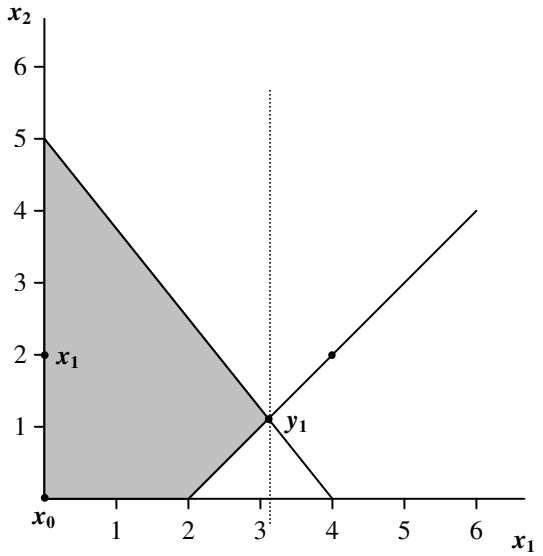


Figure A.4. Finding Feasible Direction $\boldsymbol{y}_1$ from $\boldsymbol{x}_1$

A *feasible direction* is a vector $\boldsymbol{d}_1 = \boldsymbol{y}_1 - \boldsymbol{x}_1 = (3.111, 1.111) - (0, 2) = (3.111, -.889)$. Next step is to find a minimum point between $\boldsymbol{x}_1$ and $\boldsymbol{y}_1$.

2. Calculate next iteration point: x_2

Perform the line search to find a step size α which gives minimum value of the function f.

$$\begin{aligned} \boldsymbol{x}_1 + \alpha\, \boldsymbol{d}_1 &= (0, 2) + \alpha\,(3.111, -.889) \\ &= (3.111\alpha, 2 - .889\alpha) \end{aligned}$$

Then,

$$\begin{aligned} f(\boldsymbol{x}_1 + \alpha\, \boldsymbol{d}_1) &= f\,(3.111\alpha, 2 - .889\alpha) \\ &= 3(3.111\alpha)^2 - 24(3.111\alpha) + 2(2 - .889\alpha)^4 \\ &\quad - 64(2 - .889\alpha) \\ &= 32 - 131.56\alpha + 66.9684\alpha^2 - 11.2416\alpha^3 + 1.2492\alpha^4 \end{aligned}$$

Now solve

$$\begin{aligned} \text{Minimize}\ \ & 32 - 131.56\alpha + 66.9684\alpha^2 - 11.2416\alpha^3 + 1.2492\alpha^4 \\ \text{s.t.}\ \ & 0 \le \alpha \le 1. \end{aligned}$$

The solution is $\alpha^* = 0.655$. Hence the new iteration point is

$$\boldsymbol{x}_2 = (0, 2) + .655(3.111, -.889) = (2.0377, 1.4177).$$

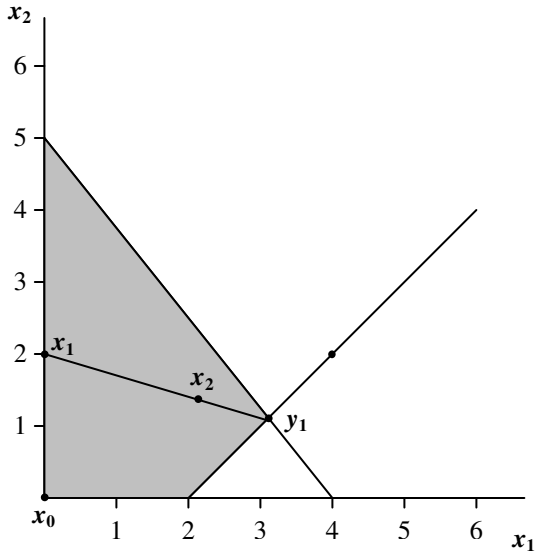


Figure A.5. New Iteration Point $\boldsymbol{x}_2$

3. Test a stopping criterion.

$$(f(x_1) - f(x_2)) / |f(x_2)| = (-96 - (-119.1018))/| -119.1018|$$
$$= .194 \geq \varepsilon = .01$$

Since the ratio is larger than ε = 0.01, continue the algorithm.

Let $k = k + 1 = 2$ and go to step 1 for next iteration.

(**Iteration 3**)

1. Find a search direction: $\boldsymbol{d_2}$

Partial derivatives of the function f at $\boldsymbol{x}_2 = (x_1, x_2) = (2.0377, 1.4177)$ are:

$$\frac{\partial f}{\partial x_1} = 6x_1 - 24 = -11.7738, \text{ and}$$

$$\frac{\partial f}{\partial x_2} = 8x_2^3 - 64 = -41.2048.$$

Using the derivatives evaluated at $\boldsymbol{x}_2$ as cost coefficients solve the following minimization problem:

$$\begin{aligned} \text{Minimize } & g_2(y) = -11.7738\, y_1 - 41.2048\, y_2 \\ \text{s.t. } & 5y_1 + 4\, y_2 \leq 20 \\ & y_1 - \ y_2 \leq 2 \\ & y_1 \geq 0,\ y_2 \geq 0. \end{aligned}$$

The solution of the above LP minimization problem is $\boldsymbol{y}_2 = (y_1, y_2) = (0, 5)$ as shown in **Figure A.6**.

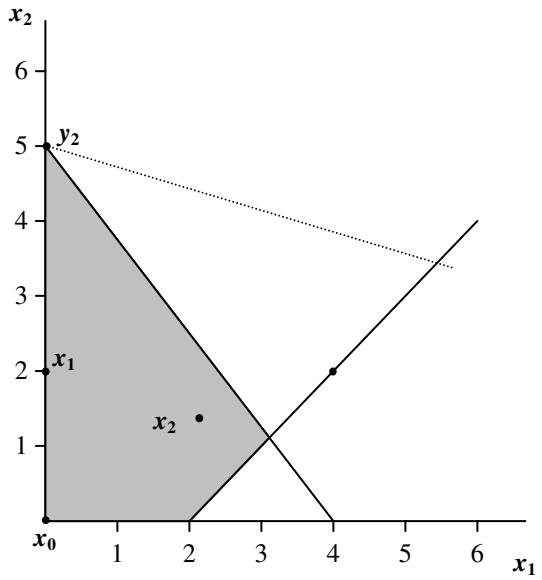


Figure A.6. Finding Feasible Direction $\boldsymbol{y}_2$ from $\boldsymbol{x}_2$

A *feasible direction* is a vector $\boldsymbol{d}_2 = \boldsymbol{y}_2 - \boldsymbol{x}_2 = (0, 5) - (2.0377, 1.4177) = (-2.0377, 3.5823)$. Next step is to find a minimum point between $\boldsymbol{x}_2$ and $\boldsymbol{y}_2$.

2. Calculate next iteration point: $\boldsymbol{x}_3$

Perform the line search to find a step size α which gives minimum value of the function f. Then,

$$f(\boldsymbol{x}_2 + \alpha\, \boldsymbol{d}_2) = f\,(2.0377 - 2.0377\alpha,\ 1.4177 + 3.5823\alpha)$$

Now solve

$$\text{Minimize } f\,(2.0377 - 2.0377\alpha,\ 1.4177 + 3.5823\alpha)$$
$$\text{s.t. } 0 \le \alpha \le 1.$$

The solution is $\alpha^* = .1393$ Hence the new iteration point is

$$\boldsymbol{x}_3 = \boldsymbol{x}_2 + \alpha\, \boldsymbol{d}_2$$
$$= (2.0377, 1.4177) + .1393(-2.0377, 3.5823)$$
$$= (1.7538, 1.9168).$$

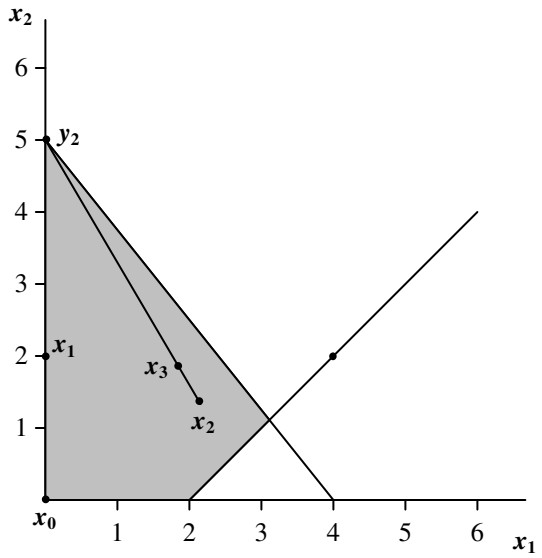


Figure A.7. New Iteration Point $\boldsymbol{x}_3$

3. Test a stopping criterion.

$$(f(x_2) - f(x_3)) / |f(x_3)| = (-119.1018 - (-128.5406))/|-128.5406|$$
$$= .0734 \geq \varepsilon = .01.$$

Since the ratio is still larger than ε = 0.01, continue the algorithm.

Let $k = k + 1 = 3$ and go to step 1 for next iteration.

(Iteration 4)

1. Find a search direction: $\boldsymbol{d}_3$

Partial derivatives of the function f at $\boldsymbol{x}_3 = (1.7538, 1.9168)$ are:

$$\frac{\partial f}{\partial x_1} = 6\,x_1 - 24 = -13.4771, \text{ and}$$

$$\frac{\partial f}{\partial x_2} = 8\,x_2^3 - 64 = -7.6624.$$

Using the derivatives evaluated at $\boldsymbol{x}_3$ as cost coefficients solve the following minimization problem:

$$\text{Minimize } g_2(y) = -13.4771\, y_1 - 7.6624\, y_2$$
$$\text{s.t. } 5y_1 + 4\, y_2 \le 20$$
$$y_1 - y_2 \le 2$$
$$y_1 \ge 0,\ y_2 \ge 0.$$

The solution of the above LP minimization problem is $\boldsymbol{y}_3 = (y_1, y_2) = (3.111, 1.111)$ as shown in **Figure A.8**.

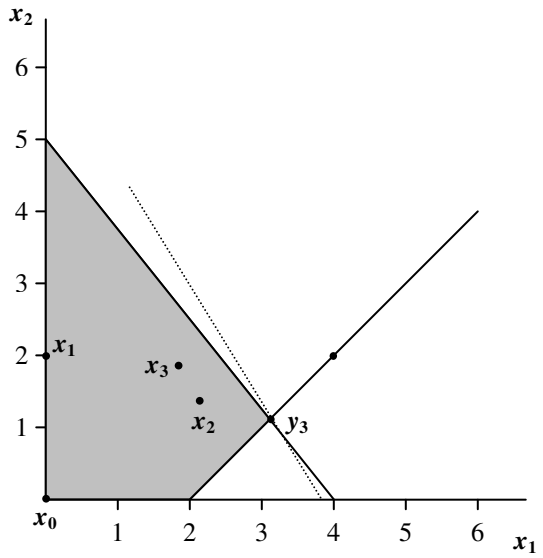


Figure A.8. Finding Feasible Direction $\boldsymbol{y}_3$ from $\boldsymbol{x}_3$

A *feasible direction* is a vector $\boldsymbol{d}_3 = \boldsymbol{y}_3 - \boldsymbol{x}_3 = (3.111, 1.111) - (1.7538, 1.9168) = (1.3573, -0.8057)$. Next step is to find a minimum point between $\boldsymbol{x}_3$ and $\boldsymbol{y}_3$.

2. Calculate next iteration point: $\boldsymbol{x}_4$

Perform the line search to find a step size α which gives minimum value of the function f. Then,

$$f(\boldsymbol{x}_3 + \alpha\, \boldsymbol{d}_3) = f\,((1.7538, 1.9168) + \alpha\, (1.3573, -0.8057))$$

Now solve

$$\text{Minimize } f(1.7538 + 1.3573\alpha,\ 1.9168 - .8057\alpha)$$
$$\text{s.t. } 0 \le \alpha \le 1.$$

The solution is $\alpha^* = .1898$ Hence the new iteration point is

$$\boldsymbol{x}_4 = \boldsymbol{x}_3 + \alpha\, \boldsymbol{d}_3$$
$$= (1.7538, 1.9168) + .1898\ (1.3573, -0.8057)$$
$$= (2.0115, 1.7638).$$

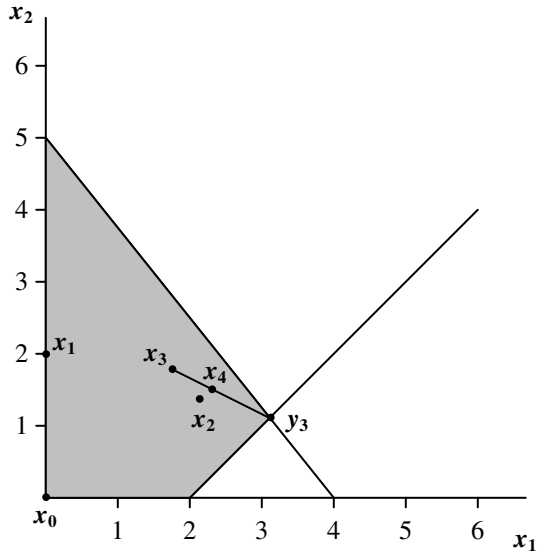


Figure A.9. New Iteration Point $\boldsymbol{x_4}$

3. Test a stopping criterion.

$$(f(x_3) - f(x_4)) / |f(x_4)| = (-128.5406 - (-129.6646))/|- 129.6646|$$
$$= .0087 \le \varepsilon = .01$$

Since the stopping rule is satisfied the algorithm stops with the current solution, x_4 and the function value $f(x_4) = -129.6646$.

APPENDIX B

COMPUTER IMPLEMENTATION

B.1. Introduction

Due to the size of the network in real life situation the proposed algorithm needs to be implemented by a computer. This appendix shows the general description of the computer code along with the input requirement and the structure of subroutines.

B.2. General Description of the Code

The computer program was coded in the MATLAB 6.5 scripts. It was developed and tested on a computer with an Intel Pentium IV 3.06 GHz processor and 512 megabytes of main memory. Currently the program is customized to fit the need of individual problem with different network configurations and input parameters. However it can be modified to handle more general problems. The maximum size of the problem, but not necessarily limited by that number, solved by the program is a network with 108 nodes, 162 links 6 sources, 8 destinations, and 2 vehicle classes.

During a computer program is being developed there is always a trade-off between memory usage and a CPU time. Currently the code does not thoroughly consider the trade-off since it is developed in the MATLAB environment which governs the major performance. In order to save time and effort to develop MATLAB script files, whenever it is possible, built-in MATLAB function was used in the code. In the implementation of the modified Frank-Wolfe algorithm, a shortest path problem is solved as a sub-problem of the given nonlinear problem. The shortest path problem is coded with the well-known Dijkstra's algorithm.

B.3. Flow of Program and Relationship between MATLAB Script Files

All subroutines and functions used in MATLAB are called *script* and saved with file extension of ".m". The multi-commodity traffic flow estimation program consists of many *scripts* which returns the variables with their values changed after execution of the codes in the *scripts*. Figure B.1 shows the relationship between MATLAB scrip files.

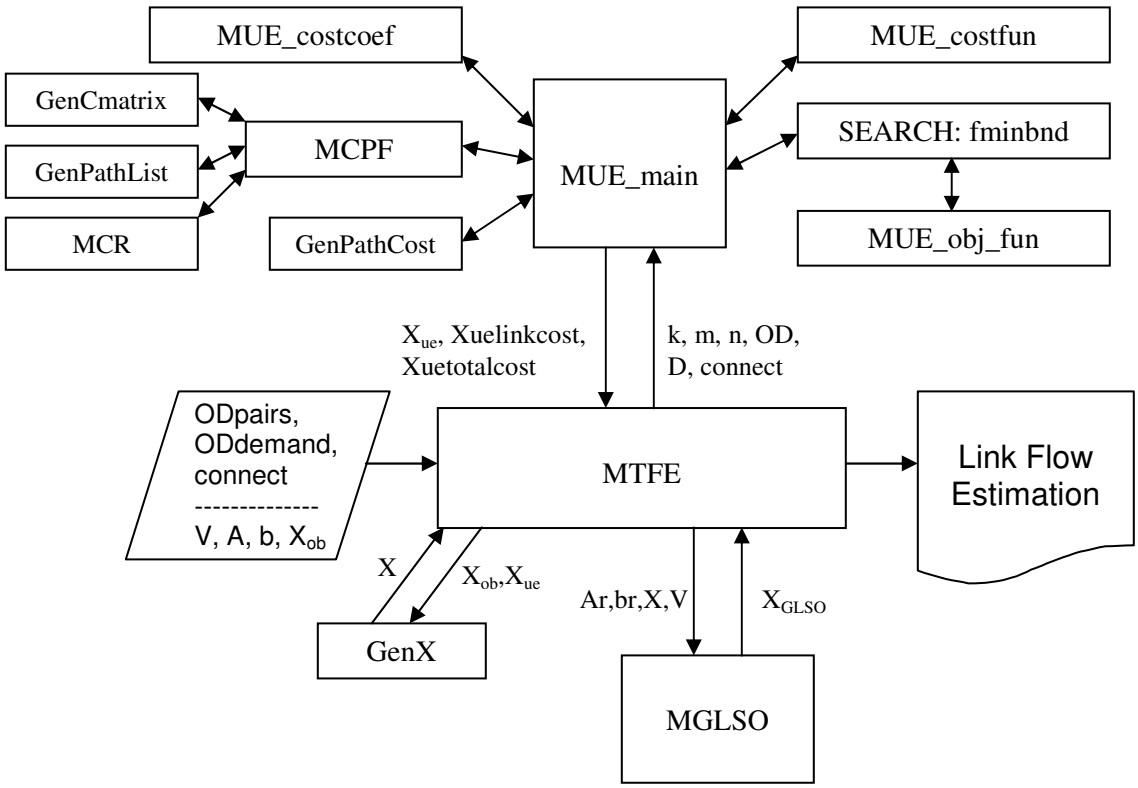


Figure B.1. Diagram of Matlab Program Modules

[MTFE] is a main control unit which accepts input variables, call [MUE_main] to obtain MDUE solution, call [GenX] to combine X_{ue} and X_{ob} as an input to [MGLSO] and generates final flow estimates. Variables exchanged between the major function modules are shown next to the arrows in Figure B.1

Each module contains brief explanation of the codes in it. The actual codes are listed in section B.4.

B.4. Script Files of MATLAB Code

This section shows the actual MATLAB code to run a sample program with 108 nodes, 162 links, 6 origins, 8 destinations, and 2 vehicle classes (commodities) which is illustrated in Chapter V. Some of the lengthy repetitive codes are omitted from the routines to enhance the readability of the program.

[MTFE.m]

```
% Master script file which runs the overall algorithm.
% 108 nodes, 162 links, 6 origins, 8 destinations, 2 commodities
%============================
% Index & Initial definition of variables
%============================
%   n      : number of nodes
%   m      : number of links
%   k      : number of commodities
%   l      : number of O-D pairs for commodity k
%   f      : Original non-linear cost function
%   g      : Gradient of f used in min-cost path problem
%   Ar     : Reduced node-arc incident matrix. To be full rank, one row
%            of each commodity will be removed from the original matrix
%   br     : Reduced RHS of the constraints.
%   x0     : initial starting value
%   xob    : Observed link flows on selected links
%   xue    : User equilibrium solution from MCUE function
%   x      : Combined matrix of 'xue' and 'xob'. x=[x1,x2,...,xk];
%            Input to MGLSO module.  x(k,i)=x(commodity,arc)
%   y      : Final MGLSO solution
%   h(k,i)=h(commodity,path#) : Path flow
%   V      : Variance-covariance matrix between links
%   p      : index of O-D pair
%   OD(:,:,k)  : 3-dimensional O-D node vectors
%   D(p, k): Demand vector of O-D pairs of commodity k
%              D = [d(OD1);d(OD2);...;d(ODp)]
%   connect(:,:,k): matrix of interconnecting links between nodes
```

```
%   y(k, i) = y(commodity, link) : Link flow
%   h(k, i) = h(commodity, path#) : Path flow
%   xprime(:,1) : x' (or x) in the algorithm (Iteration Point)
%   xprime(:,2) : x" (or y, or xx) in the algorithm (Search Direction)

%=============
% Set initial inputs
%=============
global  xprime xiter
global  pathlist1...(omitted for editing purpose)…pathlistt48
global  pathcost1…(omitted for editing purpose)…pathcostt48

%-----------------------------------------------------
disp ('<> Start Finding UE Solution, xue <>')
%-----------------------------------------------------
k=2; % # of commodities
m=162; % # of links
n=108; % # of nodes

%Set all path lists for 48 O-D pairs to zero
pathlist1=zeros(0); %path list for OD(1,58) for comm 1
pathlist2=zeros(0); %path list for OD(1,59) for comm 1
pathlist3=zeros(0); %path list for OD(1,60) for comm 1
pathlist4=zeros(0); %path list for OD(1,64) for comm 1
pathlist5=zeros(0); %path list for OD(1,65) for comm 1
pathlist6=zeros(0); %path list for OD(1,66) for comm 1
pathlist7=zeros(0); %path list for OD(1,68) for comm 1
pathlist8=zeros(0); %path list for OD(1,9) for comm 1
pathlist9=zeros(0); %path list for OD(2,58) for comm 1
pathlist10=zeros(0); %path list for OD(2,59) for comm 1
pathlist11=zeros(0); %path list for OD(2,60) for comm 1
pathlist12=zeros(0); %path list for OD(2,64) for comm 1
pathlist13=zeros(0); %path list for OD(2,65) for comm 1
pathlist14=zeros(0); %path list for OD(2,66) for comm 1
pathlist15=zeros(0); %path list for OD(2,68) for comm 1
pathlist16=zeros(0); %path list for OD(2,9) for comm 1
pathlist17=zeros(0); %path list for OD(10,58) for comm 1
pathlist18=zeros(0); %path list for OD(10,59) for comm 1
pathlist19=zeros(0); %path list for OD(10,60) for comm 1
pathlist20=zeros(0); %path list for OD(10,64) for comm 1
pathlist21=zeros(0); %path list for OD(10,65) for comm 1
pathlist22=zeros(0); %path list for OD(10,66) for comm 1
pathlist23=zeros(0); %path list for OD(10,68) for comm 1
```

```
pathlist24=zeros(0); %path list for OD(10,9) for comm 1
pathlist25=zeros(0); %path list for OD(13,58) for comm 1
pathlist26=zeros(0); %path list for OD(13,59) for comm 1
pathlist27=zeros(0); %path list for OD(13,60) for comm 1
pathlist28=zeros(0); %path list for OD(13,64) for comm 1
pathlist29=zeros(0); %path list for OD(13,65) for comm 1
pathlist30=zeros(0); %path list for OD(13,66) for comm 1
pathlist31=zeros(0); %path list for OD(13,68) for comm 1
pathlist32=zeros(0); %path list for OD(13,9) for comm 1
pathlist33=zeros(0); %path list for OD(28,58) for comm 1
pathlist34=zeros(0); %path list for OD(28,59) for comm 1
pathlist35=zeros(0); %path list for OD(28,60) for comm 1
pathlist36=zeros(0); %path list for OD(28,64) for comm 1
pathlist37=zeros(0); %path list for OD(28,65) for comm 1
pathlist38=zeros(0); %path list for OD(28,66) for comm 1
pathlist39=zeros(0); %path list for OD(28,68) for comm 1
pathlist40=zeros(0); %path list for OD(28,9) for comm 1
pathlist41=zeros(0); %path list for OD(42,58) for comm 1
pathlist42=zeros(0); %path list for OD(42,59) for comm 1
pathlist43=zeros(0); %path list for OD(42,60) for comm 1
pathlist44=zeros(0); %path list for OD(42,64) for comm 1
pathlist45=zeros(0); %path list for OD(42,65) for comm 1
pathlist46=zeros(0); %path list for OD(42,66) for comm 1
pathlist47=zeros(0); %path list for OD(42,68) for comm 1
pathlist48=zeros(0); %path list for OD(42,9) for comm 1
%---------------------------
pathlistt1=zeros(0); %path list for OD(1,58) for comm 2
pathlistt2=zeros(0); %path list for OD(1,59) for comm 2
pathlistt3=zeros(0); %path list for OD(1,60) for comm 2
pathlistt4=zeros(0); %path list for OD(1,64) for comm 2
pathlistt5=zeros(0); %path list for OD(1,65) for comm 2
pathlistt6=zeros(0); %path list for OD(1,66) for comm 2
pathlistt7=zeros(0); %path list for OD(1,68) for comm 2
pathlistt8=zeros(0); %path list for OD(1,9) for comm 2
pathlistt9=zeros(0); %path list for OD(2,58) for comm 2
pathlistt10=zeros(0); %path list for OD(2,59) for comm 2
pathlistt11=zeros(0); %path list for OD(2,60) for comm 2
pathlistt12=zeros(0); %path list for OD(2,64) for comm 2
pathlistt13=zeros(0); %path list for OD(2,65) for comm 2
pathlistt14=zeros(0); %path list for OD(2,66) for comm 2
pathlistt15=zeros(0); %path list for OD(2,68) for comm 2
pathlistt16=zeros(0); %path list for OD(2,9) for comm 2
pathlistt17=zeros(0); %path list for OD(10,58) for comm 2
```

```
pathlistt18=zeros(0); %path list for OD(10,59) for comm 2
pathlistt19=zeros(0); %path list for OD(10,60) for comm 2
pathlistt20=zeros(0); %path list for OD(10,64) for comm 2
pathlistt21=zeros(0); %path list for OD(10,65) for comm 2
pathlistt22=zeros(0); %path list for OD(10,66) for comm 2
pathlistt23=zeros(0); %path list for OD(10,68) for comm 2
pathlistt24=zeros(0); %path list for OD(10,9) for comm 2
pathlistt25=zeros(0); %path list for OD(13,58) for comm 2
pathlistt26=zeros(0); %path list for OD(13,59) for comm 2
pathlistt27=zeros(0); %path list for OD(13,60) for comm 2
pathlistt28=zeros(0); %path list for OD(13,64) for comm 2
pathlistt29=zeros(0); %path list for OD(13,65) for comm 2
pathlistt30=zeros(0); %path list for OD(13,66) for comm 2
pathlistt31=zeros(0); %path list for OD(13,68) for comm 2
pathlistt32=zeros(0); %path list for OD(13,9) for comm 2
pathlistt33=zeros(0); %path list for OD(28,58) for comm 2
pathlistt34=zeros(0); %path list for OD(28,59) for comm 2
pathlistt35=zeros(0); %path list for OD(28,60) for comm 2
pathlistt36=zeros(0); %path list for OD(28,64) for comm 2
pathlistt37=zeros(0); %path list for OD(28,65) for comm 2
pathlistt38=zeros(0); %path list for OD(28,66) for comm 2
pathlistt39=zeros(0); %path list for OD(28,68) for comm 2
pathlistt40=zeros(0); %path list for OD(28,9) for comm 2
pathlistt41=zeros(0); %path list for OD(42,58) for comm 2
pathlistt42=zeros(0); %path list for OD(42,59) for comm 2
pathlistt43=zeros(0); %path list for OD(42,60) for comm 2
pathlistt44=zeros(0); %path list for OD(42,64) for comm 2
pathlistt45=zeros(0); %path list for OD(42,65) for comm 2
pathlistt46=zeros(0); %path list for OD(42,66) for comm 2
pathlistt47=zeros(0); %path list for OD(42,68) for comm 2
pathlistt48=zeros(0); %path list for OD(42,9) for comm 2

load .\InputData\ODpairs; %|ODpairs|=(48x1x2)
load .\InputData\ODdemand; %|ODdemand|=(48x1x2)
load .\InputData\connect; %|connect|=(108x108x1)

%-------------------------------------------------------------------------------
%| Initial input data file could be acquired by the following commands.  |
%|-----------------------------------------------------------------------------|
%| disp('<> Select input data file for UE <>');                                |
%| [filename,path]=uigetfile('*.mat','Pick data file for UE problem');         |
%| UEdata=[path,filename];                                                     |
%| load UEdata;                                                                |
```

```
%| disp('== Input data, A,Aeq,b,beq,lb,x0, are acquired ==');                |
%-------------------------------------------------------------------------------

%====================
%Generate UE_solution, xue
%====================

 disp('<> Start MUE_main.m <>');
[xue,xuelinkcost,xuetotalcost,lambda,stepsize]=MUE_main(k,m,n,OD,D,connect);
%|xue|=(162x1x2)
%|xuelinkcost|=(162x1x2)
disp ('<> Returned to MTFE.m <>')
disp('Saved: xue Link Cost Coeff --> xueLinkCostCoeff.mat')
save .\Outputs\xueLinkCostCoeff.mat xuelinkcost
disp('Saved: xue Total Cost --> xueTotalCost.mat')
save .\Outputs\xueTotalCost.mat xuetotalcost
disp('Saved: UE Link Flows --> xue.mat')
save .\Outputs\xue.mat xue

%==============================
% MTFE_GLSO.m (Traffic Flow Estimation)
%==============================
disp ('<> Start MTFE_GLSO (GLS Optimization) <>')

load .\InputData\V              %V : variance-covariance matrix
load .\InputData\A.mat          %A : node-arc incident matrix
load .\InputData\b              %b=RHS vector
load .\InputData\xob            %observed link flows;
load .\Outputs\xue.mat          %user equilibrium assignment

xuered=[xue(1:162,:,1);xue(1:162,:,2)];  %reduced matrix of xue
Ar = [A(1:107,:);A(109:215,:)];     %reduced matrix of A
br = [b(1:107,:);b(109:215,:)];       %reduced vector of b

%================================================
% Combine xue with xob to make input variable x for GLS optimization
%================================================
x=GenX(xuered,xob);
disp('Saved: Combined x for GLSO input --> xCombined.mat')
save .\Outputs\xCombined.mat x

%==============================
%Generate GLSO solution using QUADPROG
```

```
%================================
H=2*inv(V);
C=-2*x'*inv(V);
lb=zeros(324,1);
Aineq=zeros(1,324);
Aineq(1,11)=1;
Aineq(1,173)=1;
bineq=[2300];
%additional constraints for multi-commodity flows x(1,11)+x(2,11)<=2300 is added.

%call "quadprog" which is a built-in MATLAB function.
xGLSO=quadprog(H,C,Aineq,bineq,Ar,br,lb);

disp('<> Start GLSO_solution.m <>');
disp ('<> Returned to STFE_TTI1_2ndHalf.m <>')
disp('Saved: Final GLSO solution --> xGLSO.mat')
save .\Outputs\xGLSO_quad_multiconstraints.mat xGLSO
disp('Saved: Whole Variables --> WholeVariables.mat')
save .\Outputs\WholeVariables_quad_multiconstraints.mat
disp('<><><> End of the program <><><>');
```

[GenCmatrix.m]

```
function costmatrix=GenCmatrix(n,connect,cc);
%Generate cost matrix between the nodes
%if two nodes are not directly connected then put infinite cost
%otherwise, the cost is the cost of the connecting link
for (r=1:1:n);
        for (c=1:1:n);
                if connect(r,c)==0;
                        costmatrix(r,c,1)=inf;
                        costmatrix(r,c,2)=inf;
                else
                        costmatrix(r,c,1)=cc(connect(r,c),1,1);
                        costmatrix(r,c,2)=cc(connect(r,c),1,2);
                end
        end
end
```

[GenPathcost.m]

```
function Genpathcost(xueCostCoef,connect)
% Generate Path costs for each O-D pair using xueCostCoef and connect
```

```
global  pathlist1...(omitted for editing purpose)…pathlistt48
global  pathcost1…(omitted for editing purpose)…pathcostt48

%Calculation for Commodity 1
%OD pair No.1
npaths=size(pathlist1,1);
nnodes=size(pathlist1,2)-1;;
pathcost1=zeros(npaths,1);
for (row=1:1:npaths)
        pathcost1(row,1)=0;
        col=1;
        for (col=1:1:nnodes);
                if pathlist1(row,col+1)==0
                        break
                end
                linknow=connect(pathlist1(row,col),pathlist1(row,col+1));
                pathcost1(row,1)=pathcost1(row,1)+xueCostCoef(linknow,1,1);
                col=col+1;
        end
end
        …
        (omitted for editing purpose: do the above routine for the remaining O-D pairs)
        …
%-----------------------------------------
%Calculation for Commodity 2
%OD pair No.1
npaths=size(pathlistt1,1);
nnodes=size(pathlistt1,2)-1;;
pathcostt1=zeros(npaths,1);
for (row=1:1:npaths)
        pathcostt1(row,1)=0;
        col=1;
        for (col=1:1:nnodes);
                if pathlistt1(row,col+1)==0
                        break
                end
                linknow=connect(pathlistt1(row,col),pathlistt1(row,col+1));
                pathcostt1(row,1)=pathcostt1(row,1)+xueCostCoef(linknow,1,2);
                col=col+1;
        end
end
        …
        (omitted for editing purpose: do the above routine for the remaining O-D pairs)
```

…

[GenPathlist.m]

```
function GenPathlist(s,d,minpath,kk)
%============================================================
% Generate the paths during the algorithm and adds the new paths to the current list
% Check if current minpath is in the pathlist.
% If current minpath is newly generated then add the minpath in the pathlist
%============================================================

global  pathlist1...(omitted for editing purpose)…pathlistt48

% origin from node 1
if (kk= =1)      % for commodity 1
        if (s= =1)&(d= =58)
                size0=size(pathlist1,1);
                size1=size(pathlist1,2);
                size2=size(minpath,2);
                if size1>size2
                        minpath=[minpath,zeros(1,size1-size2)];
                elseif size1<size2
                        pathlist1=[pathlist1,zeros(size0,size2-size1)];
                end
                exist=0;
                for (row=1:1:size0)
                        if isequal(pathlist1(row,:),minpath)
                        %check whether current minpath exist in the list
                                exist=1;
                                break
                        end
                end
                if exist~=1 %if current minpath is new one, add it to the pathlist
                        pathlist1=[pathlist1;minpath];
                end
        end
        …
        (omitted for editing purpose: do the above routine for the remaining O-D pairs)
        …
%-------------------------------
else     % for commodity 2
%-------------------------------
        if (s==1)&(d==58)
```

```
                size0=size(pathlistt1,1);
                size1=size(pathlistt1,2);
                size2=size(minpath,2);
                if size1>size2
                        minpath=[minpath,zeros(1,size1-size2)];
                elseif size1<size2
                        pathlistt1=[pathlistt1,zeros(size0,size2-size1)];
                end
                exist=0;
                for (row=1:1:size0)
                        if isequal(pathlistt1(row,:),minpath)
                                exist=1;
                                break
                        end
                end
                if exist~=1
                        pathlistt1=[pathlistt1;minpath];
                end
        end
        …
        (omitted for editing purpose: do the above routine for the remaining O-D pairs)
        …
end
```

[GenX.m]

```
function [x] = GenX(xuered,xob)
%=========================================
% Combine xue with xob to make input variable x for GLSO
%=========================================
sxob=size(xob);
x=zeros(sxob);
for(ii=1:1:sxob(1))
        if(xob(ii,1)<=0)
                x(ii,1)=xuered(ii,1);
        else
                x(ii,1)=xob(ii,1);
        end
end
```

[GLSO.m]

```
function [GLSO_LinkFlows] = GLSO(Ar,br,x,V)
```

```
%================================
% Min  (y-x)inv(V)(y-x)
% s.t.  Ay=b
%       y>=0
%================================
% y   : Decision variable of the GLSO problem, xGLSO
% Ar  : Reduced node-arc incident matrix.
% br  : Reduced RHS of the constraints.
% x   : Combined matrix of links flows of all commodities.
% V   : Variance-covariance matrix of all commodities.

szAr = size(Ar);
i = eye(szAr(2));
p = V*Ar';
q = Ar*V*Ar';
GLSO_LinkFlows = [i-p*(q\Ar)]*x + p*(q\br);
```

[MCPF.m]

```
function [y]=MCPF(k,m,n,cc,OD,D,connect)
% returns Min Cost Link flows [y]
%=======================================================
% Problem formulation (Min-Cost Path Flow Formulation: Subproblem:p2)
%=======================================================
% Min g(x)=c1(x0)x(1)+...
% s.t. x(1) = h...
%              :
%     x>=0, h>=0
% ---------------------------------------------------------------
% Notation:
%   x(k,i)=x(commodity,link) : Link flow
%   h(k,i)=h(commodity,path#) : Path flow
%   cc  : original cost coefficient
%   y   : link flows (return value to caller module)
%   h   : path flows (return value to caller module)
%   connect(:,:,k): shows interconnecting links between nodes
%   D(pp,k): Demand vector of O-D pairs of commodity k.
%   pair=size(OD,1) : # of O-D pairs of a commodity
%   OD(pp,st,k) : 3-dimensional O-D node vectors
%                     (pp=pair#, st=1:source, 2:terminal, k=commodity)
%   costmatrix(f,t,k) : link cost matrix from current link flows
%                           (f=from node,t=to node,k=commodity)
```

```
global  pathlist1...(omitted for editing purpose)…pathlistt48

%==============================
% initial matrices used in the MCR routine
%==============================
costmatrix=GenCmatrix(n,connect,cc);
y=zeros(m,1,k);         % initialize link flow y to 0

% find minimum path of O-D pair (s,d)
%-------------------------------------------
kk=1;                   %counter for WHILE loop for commodity change
pair=size(OD,1);        %OD=(48,2,2)
while (kk<=k)
        cmatrix=costmatrix(:,:,kk);     %costmatrix for kk-th commodity.
        pp=1;
        while (pp<=pair)
                s=OD(pp,1,kk);          % source node of 1st pair of kk-th commodity
                d=OD(pp,2,kk);          % destination node of 1st pair of kk-th commodity
                [minpath]=MCR(n, cmatrix, s, d);

                %Generate Path List
                Genpathlist(s,d,minpath,kk)   % add current one, if it's new path.
                counter=size(minpath,2)-1; %counter=# of min path links

                %       assign demand to min-path links
                for (index=1:1:counter);
                        linkno=connect(minpath(index),minpath(index+1));
                        y(linkno,1,kk)=y(linkno,1,kk)+D(pp,1,kk);
                end
                pp=pp+1;
        end
        kk=kk+1;
end
```

[MCR.m]

```
function [path, totalCost] = MCR(n, Distance, s, d)
% This MCR codes was found MATLAB user library and modified for the current
problem.
% Returns min-path node list
% Minimum Cost Route (shortest path) algorithm: Dijkstra's algorithm
% path: the list of nodes in the path from source to destination;
```

```
% totalCost: the total cost of the path;
% n: the number of nodes in the network;
% s: source node;
% d: destination node;
% DistMatrix: distance matrix between nodes

%======================
%    Initialization
%======================
visited(1:n) = 0;          % set all the nodes un-visited;
distance(1:n) = inf;       % set distance between s to i to infinity;
predec(1:n) = 0;           % set predecessor nodes to all zero
distance(s) = 0;           % set distance from s to itself to zero

for ii = 1:(n-1),
        temp = [];
        for jj = 1:n,
                if visited(jj) == 0;
                        temp=[temp distance(jj)];
                else
                        temp=[temp inf];
                end
        end;
        [t, u] = min(temp);
        visited(u) = 1;
        for v = 1:n,
                if ( ( DistMatrix(u,v) + distance(u)) < distance(v) )
                % update the shortest distance when a shorter path is found;
                        distance(v) = distance(u) + DistMatrix(u,v);
                        predec(v) = u; % update its predecessor;
                end;
        end;
end;
% generate node list in the shortest path
path = [];
if predec(d) ~= 0
        t = d;
        path = [d];
        while t ~= s
                p = predec(t);
                path = [p path];
                t = p;
        end;
```

```
end;
totalCost = distance(d);
return;
```

[MUE_costcoef.m]

```
function costcoef = MUE_costcoef(z)
% returns costcoef matrix
c1=3*(1+0.15*(z(1,1,1)/60)^4);
…
(omitted for editing purpose)
…
c289=21*(1+.15*(z(127,1,2)/36)^4);

costcoef=zeros(162,1,2);        % initialize
%-------------------------------------------------------
% cost coef. for 1st commodity
costcoef(1:127,1,1)=[c1;c2;c3;c4;c5;c6;c7;c8;c9;c10;
      c11;c12;c13;c14;c15;c16;c17;c18;c19;c20;
      c21;c22;c23;c24;c25;c26;c27;c28;c29;c30;
      c31;c32;c33;c34;c35;c36;c37;c38;c39;c40;
      c41;c42;c43;c44;c45;c46;c47;c48;c49;c50;
      c51;c52;c53;c54;c55;c56;c57;c58;c59;c60;
      c61;c62;c63;c64;c65;c66;c67;c68;c69;c70;
      c71;c72;c73;c74;c75;c76;c77;c78;c79;c80;
      c81;c82;c83;c84;c85;c86;c87;c88;c89;c90;
      c91;c92;c93;c94;c95;c96;c97;c98;c99;c100;
      c101;c102;c103;c104;c105;c106;c107;c108;c109;c110;
      c111;c112;c113;c114;c115;c116;c117;c118;c119;c120;
      c121;c122;c123;c124;c125;c126;c127];

costcoef(128:162,1,1)=0;        % set zero costs for imaginary links

%-------------------------------------------------------
% cost coef. for 2nd commodity
costcoef(1:127,1,2)=[c163;c164;c165;c166;c167;c168;c169;
      c170;c171;c172;c173;c174;c175;c176;c177;c178;c179;
      c180;c181;c182;c183;c184;c185;c186;c187;c188;c189;
      c190;c191;c192;c193;c194;c195;c196;c197;c198;c199;
      c200;c201;c202;c203;c204;c205;c206;c207;c208;c209;
      c210;c211;c212;c213;c214;c215;c216;c217;c218;c219;
      c220;c221;c222;c223;c224;c225;c226;c227;c228;c229;
      c230;c231;c232;c233;c234;c235;c236;c237;c238;c239;
```

```
        c240;c241;c242;c243;c244;c245;c246;c247;c248;c249;
        c250;c251;c252;c253;c254;c255;c256;c257;c258;c259;
        c260;c261;c262;c263;c264;c265;c266;c267;c268;c269;
        c270;c271;c272;c273;c274;c275;c276;c277;c278;c279;
        c280;c281;c282;c283;c284;c285;c286;c287;c288;c289];
costcoef(128:162,1,2)=0;        % set zero costs for imaginary links
```

[MUE_costfun.m]

```
function f = MUE_costfun(z)
% Link cost function gives total link cost.
% Returns value of the function
% Using 2-dimensional variable: z

f= z(1,1,1) * (3*(1+0.15*(z(1,1,1)/60)^4))+
    …
    (omitted for editing purpose)
    …
    + z(127,1,2) * (21*(1+.15*(z(127,1,2)/36)^4));
```

[MUE_main]

```
function [xueLinkFlow,xueCostCoef,xueTcost,lambda,stepsize] = …
MUE_main(k,m,n,OD,D,connect)
% UE main module to solve UE problem
%=============================================
% Deterministic User Equilibrium Problem: Master Problem, P1
%
% Min f(x)
% s.t. x = h...
%          :
%      x>=0, h>=0
%=============================================
% Min-Cost Path Flow Formulation: Sub problem, P2
%
% Min g(x)=c1(x0)x(1)+...
% s.t.  same constraints as in P1

global  xprime xiter
global  pathlist1...(omitted for editing purpose)…pathlistt48
global  pathcost1…(omitted for editing purpose)… pathcost48
global  pathcostt1…(omitted for editing purpose)…pathcostt48
%================================
```

```
% Finding initial starting feasible solution x'
%================================
disp('[Start Finding Initial Feasible Solution]')
cc=[MUE_costcoef(zeros(m,1,2))];
% find initial cost coefficient based on initial zero link flows
[x]=MCPF(k,m,n,cc,OD,D,connect); %Initial Feasible Solution: min cost path solution,
x
%----------------------
% main iteration loop
%----------------------
ratio=1;
iteration=1; % iteration counter
kappa=1;
alpha=1;
stepsize=1;
iterations4old=0;
xiter=x;
while (ratio>0.0001)    %stopping criterion
        disp(' ITERATION')
        disp(iteration)
        %=====================
        % finding search direction: x"
        %=====================
        %disp('[Find New Search Direction x"]')
        cc=[MUE_costcoef(x)];                   % cost coef. from x' to find x"
        [xx]=MCPF(k,m,n,cc,OD,D,connect);       % find search direction, x"

        % Find the Lambda by golden section search algorithm embedded in MATLAB
% and also find next iteration point
        xprime(:, 1, :)=x;      % current iteration point
        xprime(:, 2, :)=xx;     % search direction
lambda(iteration)=fminbnd(@MUE_Obj_fun,0,1);
%find lambda maximizing f(x) between x' and x"

        kappa=2.0;  % this kappa can be changed by modified F-W scheme
        alpha=lambda(iteration)*kappa;
        stepsize(iteration)=min(alpha,1);
        newiterpoint=xprime(:,1,:)+stepsize(iteration)*(xprime(:,2,:)-xprime(:,1,:));
        tcost0=MUE_costfun(xprime(:,1,:)); % single value
        tcost1=MUE_costfun(newiterpoint);
        if(tcost1>=tcost0) %If new solution is worse than the old one
newiterpoint=xprime(:,1,:)+lambda(iteration)*(xprime(:,2,:)- xprime(:,1,:));
                tcost1=UE_MTFE3_costfun(newiterpoint);
```

```
            iterations4old=iterations4old+1;
        end
        %=====================
        % Evaluate the stopping criteria
        %=====================
        ratio=(tcost0-tcost1)*100/abs(tcost1);
        if (tcost1<tcost0) %If new solution is better than the old one, continue algorithm
            x=newiterpoint;
        else % If new solution is worse than the old one, STOP.
            disp('<>Since the ratio is negative<>')
            disp('<>Improvement can not be made further more<>')
            newiterpoint=x;
        end
        %
        % If ratio is small enough then current newiterpoint is optimal solution and stop.
        %
        xxx(:,iteration,:)=x; %Matrix to save x values during iterations
        TotalCost(iteration,:)=tcost1; %Vector to save total costs during iterations
        iteration=iteration+1;
        xiter(:,iteration,:)=newiterpoint;
end

%============================
% display and save the outputs
%============================
iteration=iteration-1
disp('=====================')
disp('Total Number of Iterations =')
disp(iteration)
save .\Outputs\iteration.mat iteration

disp('Using Modified Frank-Wolfe Strategy')
disp('----------------------------------------')
disp('kappa=2.0')
disp('----------------------------------------')
disp('Number of iterations using conventional Method')
disp(iterations4old)
save .\Outputs\iteration4old.mat iterations4old

xueLinkFlow=newiterpoint;
disp('Saved: UE Link Flows --> "xueLinkFlow" ')
save .\Outputs\xueLinkFlow.mat xueLinkFlow
xueCostCoef=UE_MTFE3_costcoef(newiterpoint);
```

```
disp('Saved: UE Link Cost Coefficients --> "xueCostCoef"')
save .\Outputs\xueCostCoef.mat xueCostCoef

%================================
% Calculate path costs
%================================
Genpathcost(xueCostCoef,connect);
disp('Saved: UE Path lists --> "UEPathList"')
save .\Outputs\UEPathList.mat pathlist*
disp('Saved: UE Path Costs --> "UEPathCost"')
save .\Outputs\UEPathCost.mat pathcost*
xueTcost=tcost1;
```

[MUE_Obj_fun.m]

```
function f = MUE_Obj_fun(w)
% Objective function of Traffic Assignment Problem (P1)
% Using 1-dimensional variable: w

global xprime;

z = (xprime(:,2,:)-xprime(:,1,:))*w+xprime(:,1,:);
f = MUE_costfun(z);
```

APPENDIX C

COMMON SOLUTION APPROACHES FOR MULTI-COMMODITY NETWORK PROBLEMS

A multi-commodity problem can be represented by the following equations:

$$\text{minimize } \sum_k \mathbf{c}^k \mathbf{x}^k \tag{C.1}$$

$$\text{subject to } \mathbf{A}\mathbf{x}^k = \mathbf{b}^k, \; k = 1,\ldots,K \tag{C.2}$$

$$\sum_k \mathbf{D}^k \mathbf{x}^k \leq \mathbf{u} \tag{C.3}$$

$$\mathbf{0} \leq \mathbf{x}^k \leq \mathbf{u}^k, \; k = 1,\ldots,K \tag{C.4}$$

where $\mathbf{A}$ is a node-arc incidence matrix for the network and $\mathbf{D}^k$ for k=1,…,K are diagonal matrices. Usually the diagonal matrices $\mathbf{D}^k$ are identity matrices. The *j*th component of $\mathbf{u}$ is called *mutual (common) arc capacity* of arc *j*. The flow of all commodities on the arc *j* is constrained by the arc capacity. The constraint (C.2) is called *flow conservation constraint* and the constraint (C.3) is called *bundle (capacity) constraint.* Different commodities interact with each other by the set of bundle constraints. In order to eliminate the inequality constraint in (C.3) we add nonnegative slack variables. When there is no weighting factor for an arc the diagonal matrices $\mathbf{D}^k$ reduce to identity matrices. Now we can rewrite the equation (C.3) into the equations (C.5) and (C.6).

$$\sum_k \mathbf{x}^k + \mathbf{s} = \mathbf{u} \tag{C.5}$$

$$\mathbf{s} \geq \mathbf{0} \tag{C.6}$$

Even though multi-commodity flow problems do not have the same nice properties as single-commodity flow problems they still have some special structure that we can exploit to solve the problem efficiently. There are several common approaches for solving the multi-commodity flow problem. The solution methods generally attempt to exploit the network flow structure of the individual single commodity flow problems. In the following sections we will briefly explain the underlying concepts and algorithms of the decomposition methods.

C.1. Price-Directive Decomposition Method

Decomposition method places multi-commodity network flow problem in a form where a master optimization problem coordinates the solution of subproblems and each subproblem is a minimal cost network flow problem. This approach is divided into *price-directive* and *resource-directive* methods. **Price-directive decomposition** approach places the *prices* (dual variables) on the bundle constraints and brings these into the objective function. It removes (relaxes) the complicating capacity constraint and charges each commodity for the use of the arc. The objective is to obtain a set of prices such that the combined solution for all subproblems yields an optimum for the original problem.

For each commodity, $k = 1,\ldots,K$, let $X_k = \{\mathbf{x}^k : \mathbf{A}\mathbf{x}^k = \mathbf{b}^k,\ \mathbf{0} \le \mathbf{x}^k \le \mathbf{u}^k\}$ and let $\mathbf{x}_1^k,\ldots, \mathbf{x}_q^k$ denote the extreme points of X_k. If X_k is the null set for any k, then the original problem has no solution. Suppose λ is dual variable associated with the bundle constraints (C.5) and X_k is not the null set and bounded, then any x_k can be expressed as a convex combination of the extreme points of $\mathbf{x}_j^k$ as follows,

$$\begin{aligned} &\text{Minimize } \mathbf{x}^k = \sum_j x_j^k \lambda_j^k \\ &\text{where} \quad \sum_j \lambda_j^k = 1, \text{ all k} \\ &\qquad \lambda_j^k \ge 0, \text{ all j, k} \end{aligned} \tag{C.7}$$

Substituting (C.7) in the multi-commodity minimal cost flow problem, (C.1)-(C.6), we obtain the following:

$$\text{minimize} \sum_{j,k} (\mathbf{c}^k x_j^k) \lambda_j^k \tag{C.8}$$

$$\text{subject to} \sum_{j,k} x_j^k \lambda_j^k + s = b \qquad (\mathbf{w}) \tag{C.9}$$

$$\sum_j \lambda_j^k = 1, \text{ all } k \qquad (\boldsymbol{\alpha}) \tag{C.10}$$

$$\lambda_j^k \geq 0, \text{ all } j, k. \tag{C.11}$$

where $\mathbf{w}$ and $\boldsymbol{\alpha}_k$ are dual variables.

Suppose we have a basic feasible solution to the multi-commodity minimal cost flow problem in terms of the $\boldsymbol{\lambda}_j^k$ and w and α_k are dual variables of (C.9) and (C.10), respectively. Then dual feasibility associated with the above problem is:

(i) $w_m \leq 0$ corresponding to each slack variable, s_m, and

(ii) $(w - c^k)\, y_j^k + \alpha_k \leq 0$ corresponding to each λ_j^k.

Any variable violating any of these conditions is a candidate to enter the master basis. Finding the most violating variable for any λ_j^k involves solving following subproblems.

$$\text{minimize } (\mathbf{c}^k - \mathbf{w})\, \mathbf{x}^k + \alpha^k \tag{C.12}$$

$$\text{subject to } \mathbf{A}\mathbf{x}^k = \mathbf{b}^k \tag{C.13}$$

$$\mathbf{0} \leq \mathbf{x}^k \leq \mathbf{u}^k, \; k = 1,\ldots,K \tag{C.14}$$

Problem (C.8)-(C.11) is called the master problem and the problems (C.12)-(C.14) are called the subproblems. The master problem is solved by the revised simplex method with the subproblems, which are used to test for optimality and select candidates for entering the master problem basis. The subproblems are single-commodity problems and can be solved efficiently by any well-known techniques such as out-of-kilter algorithm or primal simplex algorithm for network optimization.

C.2. Price-Directive Decomposition Algorithm.

Initialization: Find an initial feasible basis for the master problem and the corresponding dual variables. If a feasible basis is not available, then one may use artificial variables and a two-phase method to find a starting feasible solution for the master problem.

Step 1. Pricing: Let $\mathbf{x}^k$ denote an optimum extreme point for $\mathbf{z}_k = \min \{(\mathbf{c}^k - \mathbf{w})\, \mathbf{x}^k : \mathbf{A}\mathbf{x}^k = \mathbf{b}^k,\ \mathbf{0} \le \mathbf{x}^k \le \mathbf{u}^k\}$. If there is no solution, then the master problem has no solution. If $(\mathbf{w} - \mathbf{c}^k)\, \mathbf{x}_j^k + \alpha_k > 0$ (or $\alpha_k - \mathbf{z}_k > 0$), then λ^k is a candidate to enter the basis of the master. Otherwise, no extreme point of X_k is a candidate for basis entry. If $w_m > 0$, then the corresponding slack, s_m, is a candidate for basis entry. Does there exist at least one candidate for basis entry? If so, continue with step 2. If not, terminate; optimality has been obtained.

Step 2. Pivot in Master : Select an eligible variable for basis entry. Update the chosen column, pivot in the master program, and return to step 1 with a new set of dual variables.

(Example C.1) Price-directive decomposition method

We will consider the numerical example shown in Figure C.1. The network problem has two source nodes, 1 and 2, and two terminal nodes, 3 and 4, for commodity 1. The

second commodity has a source node 2 and a terminal node 3. Notice that the optimal solution of this problem can not be solved separately for each commodity. This is because the capacity of a certain arc may be insufficient to serve the flow of two commodities.

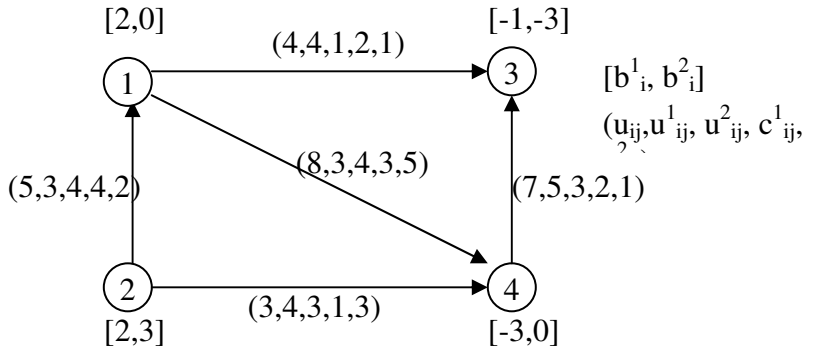


Figure C.1. Two Commodity Minimum Cost Flow Problem

The above problem is assumed to have zero lower bounds for their arc flows and fixed flows for their nodes without loss of generality. The matrix representation of the problem is shown in Figure C.2. By network transformation procedure we add one imaginary arc, x^1_1 and x^2_1 in Figure C.2., to the original network to make the node-arc matrix full rank. For simplicity, this matrix representation does not show the upper bound constraints.

	x^1_{13}	x^1_{14}	x^1_{21}	x^1_{24}	x^1_{43}	x^1_1	x^2_{13}	x^2_{14}	x^2_{21}	x^2_{24}	x^2_{43}	x^2_1	s_{13}	s_{14}	s_{21}	s_{24}	s_{43}	RHS
	1	1	-1	0	0	-1												2
node-arc matrix	0	0	1	1	0	0				∅						∅		2
for commodity 1	-1	0	0	0	-1	0												-1
	0	-1	0	-1	1	0												-3
							1	1	-1	0	0	-1						0
node-arc matrix				∅			0	0	1	1	0	0				∅		3
for commodity 1							-1	0	0	0	-1	0						-3
							0	-1	0	-1	1	0						0
	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	4
	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	8
bundle constraints	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	5
	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	3
	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	7

Figure C.2. The Matrix Representation of Example C.1

Initialization

In this stage, we build the revised simplex tableau of the master problem. It starts with the finding a feasible solution of the problem. In order to do that, we can use conventional LP methods such as two-phase method to find a starting feasible solution.

In our example we use the following solutions: $\mathbf{x}^1{}_1=(0, 2, 0, 2, 1)^T$, and $\mathbf{x}^2{}_1=(0, 2, 2, 1, 3)^T$ where $\mathbf{x}^k_{iteration}= (x^k_{13}, x^k_{14}, x^k_{21}, x^k_{24}, x^k_{43})$. Cost vectors are $\mathbf{c}^1=(2, 3, 4, 1, 2)$, $\mathbf{c}^2=(1, 5, 2, 3, 1)$ where $\mathbf{c}^k=(c^k_{13}, c^k_{14}, c^k_{21}, c^k_{24}, c^k_{43})$.

From the master problem formulation we can have the following basis and the inverse.

$$B = \left(\begin{array}{ccccc:cc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 1 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 3 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) \qquad B^{-1} = \left(\begin{array}{ccccc:cc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -2 & -2 \\ 0 & 0 & 1 & 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & 1 & 0 & -2 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & -3 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

From the given problem condition we have

$\mathbf{c}^1 \mathbf{x}^1{}_1 = 10, \ \mathbf{c}^2 \mathbf{x}^2{}_1 = 20$

$(\mathbf{w},\boldsymbol{\alpha}) = \mathbf{c}_B' \mathbf{B}^{-1} = (0, 0, 0, 0, 0, 10, 20)$

$\mathbf{b}' = (4, 8, 5, 3, 7, 1, 1)$

$\mathbf{B}^{-1} \mathbf{b}' = (4, 4, 3, 0, 3, 1 ,1)^T$

$\mathbf{c}_B' \mathbf{B}^{-1} \mathbf{b}' = 30.$

We are now ready to make a revised simplex tableau of the master problem shown in Table C.1.

Table C.1. Simplex Tableau of the Master Problem

	w_{13}	w_{14}	w_{21}	w_{24}	w_{43}	α_1	α_2	RHS
z	0	0	0	0	0	10	20	30
s_{13}	1	0	0	0	0	0	0	4
s_{14}	0	1	0	0	0	-2	-2	4
s_{21}	0	0	1	0	0	0	-2	3
s_{24}	0	0	0	1	0	-2	-1	0
s_{43}	0	0	0	0	1	-1	-3	3
$\lambda^1{}_1$	0	0	0	0	0	1	0	1
$\lambda^2{}_1$	0	0	0	0	0	0	1	1

Iteration 1

In the above revised simplex tableau, all $w_{ij} \leq 0$. That is, all w_{ij} satisfy dual feasibility. Therefore we need, now, to solve subproblems for commodity 1 and 2, which are two individual single-commodity flow problems. If we find any positive $z^k_j - c^k_j$ value, then λ^k_j can be a candidate to enter the basis.

Subproblem 1

$$\mathbf{w} - \mathbf{c}_1 = (0, 0, 0, 0, 0) - (2, 3, 4, 1, 2) = (-2, -3, -4, -1, -2).$$

This cost term consists of the following single-commodity minimum cost flow problem as showun

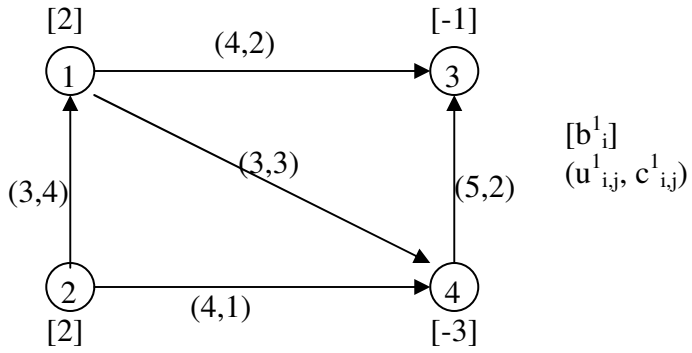


Figure C.3. Network Representation of Subproblem 1 of Iteration 1

This problem can be easily solved by out-of-kilter algorithm or network simplex algorithm. The minimum cost optimal solution is $\mathbf{x}^1{}_2 = (1, 1, 0, 2, 0)^T$.

$$z^1{}_2 - c^1{}_2 = (\mathbf{w} - \mathbf{c}_1)\, \mathbf{x}^1{}_2 + \alpha_1 = -7 + 10 = 3.$$

Since $z^1{}_2 - c^1{}_2$ is greater than zero, $\lambda^1{}_2$ could be a candidate to enter. Let's continue to subproblem 2.

Subproblem 2

$$\mathbf{w} - \mathbf{c}_2 = (0, 0, 0, 0, 0) - (1, 5, 2, 3, 1) = (-1, -5, -2, -3, -1).$$

We have the following single-commodity minimum cost flow problem for commodity 2.

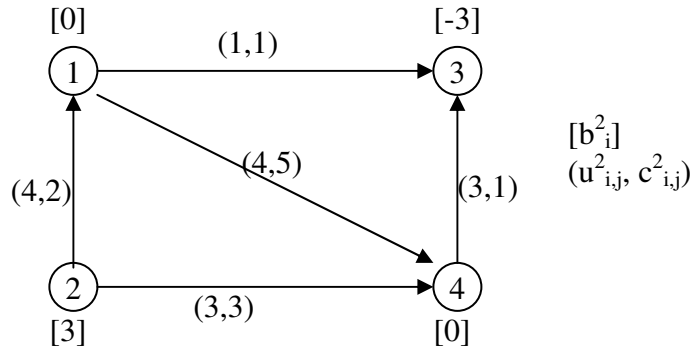


Figure C.4. Network Representation of Subproblem 2 of Iteration 1

The minimum cost optimal solution is $\mathbf{x}^2_2 = (1, 0, 1, 2, 2)^T$.

$$z^2_2 - c^2_2 = (\mathbf{w} - \mathbf{c}_2)\, \mathbf{x}^2_2 + \alpha_2 = -11 + 20 = 9.$$

Since $z^2_2 - c^2_2$ is greater than $z^1_2 - c^1_2$, we choose λ^2_2 as an entering variable to the master problem basis. Next, we generate the column and perform pivoting process. The column for λ^2_2 is calculated by

$$\mathbf{B}^{-1} (\mathbf{x}^2_2, \mathbf{e}^2)^T = \mathbf{B}^{-1} (1, 0, 1, 2, 2, 0, 1)^T = (1, -2, -1, 1, -1, 0, 1)^T.$$

Table C.2. Simplex Tableau of Iteration 1

	w_{13}	w_{14}	w_{21}	w_{24}	w_{43}	α_1	α_2	RHS	λ^2_2
z	0	0	0	0	0	10	20	30	9
s_{13}	1	0	0	0	0	0	0	4	1
s_{14}	0	1	0	0	0	-2	-2	4	-2
s_{21}	0	0	1	0	0	0	-2	3	-1
s_{24}	0	0	0	1	0	-2	-1	0	(1)
s_{43}	0	0	0	0	1	-1	-3	3	-1
λ^1_1	0	0	0	0	0	1	0	1	0
λ^2_1	0	0	0	0	0	0	1	1	1

	w_{13}	w_{14}	w_{21}	w_{24}	w_{43}	α_1	α_2	RHS	λ^2_2
z	0	0	0	-9	0	28	29	30	0
s_{13}	1	0	0	-1	0	2	1	4	0
s_{14}	0	1	0	2	0	-6	-4	4	0
s_{21}	0	0	1	1	0	-2	-3	3	0
λ^2_2	0	0	0	1	0	-2	-1	0	1
s_{43}	0	0	0	1	1	-3	-4	3	0
λ^1_1	0	0	0	0	0	1	0	1	0
λ^2_1	0	0	0	-1	0	2	2	1	0

Iteration 2

Still all $w_{ij} \leq 0$. Thus proceed to solve subproblems to find a candidate to enter the master basis.

Subproblem 1

$$\mathbf{w} - \mathbf{c}_1 = (0, 0, 0, -9, 0) - (2, 3, 4, 1, 2) = (-2, -3, -4, -10, -2).$$

This cost term consists of the following single-commodity minimum cost flow problem.

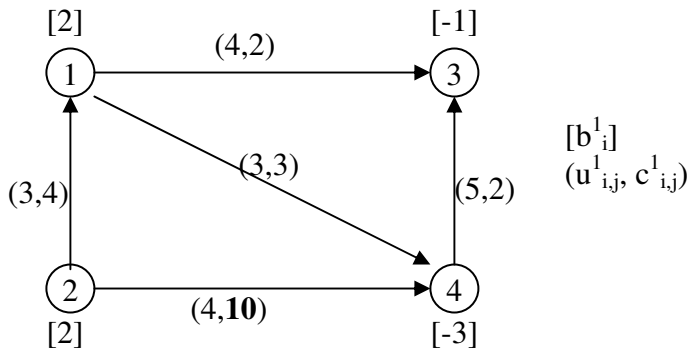


Figure C.5. Network Representation of Subproblem 1 of Iteration 2

The optimal solution is $\mathbf{x}^1_3 = (1, 3, 2, 0, 0)^T$.

$$z^1_3 - c^1_3 = (\mathbf{w} - \mathbf{c}_1)\, \mathbf{x}^1_3 + \alpha_1 = -19 + 28 = 9.$$

Since $z^1_3 - c^1_3$ is greater than zero, λ^1_3 could be a candidate to enter. Continue to subproblem 2.

Subproblem 2

$$\mathbf{w} - \mathbf{c}_2 = (0, 0, 0, -9, 0) - (1, 5, 2, 3, 1) = (-1, -5, -2, -12, -1).$$

We have the following single-commodity minimum cost flow problem.

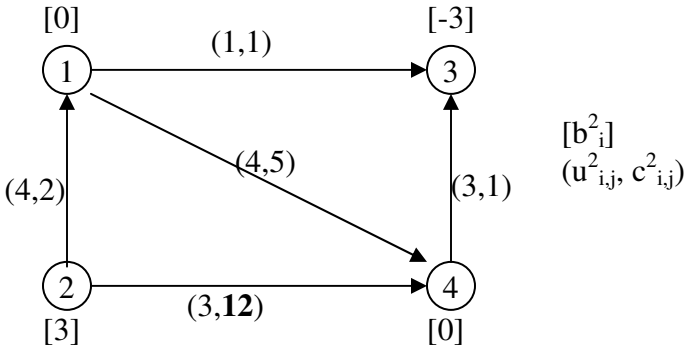


Figure C.6. Network Representation of Subproblem 2 of Iteration 2

The minimum cost optimal solution is $\mathbf{x}^2_3 = (1, 2, 3, 0, 2)^T$.

$$z^2_3 - c^2_3 = (\mathbf{w} - \mathbf{c}_2)\, \mathbf{x}^2_3 + \alpha_2 = -19 + 29 = 10.$$

Since $z^2_3 - c^2_3$ is greater than $z^1_3 - c^1_3$, we choose λ^2_3 as an entering variable to the master problem basis. Next, we generate the column and perform pivoting process. The column for λ^2_3 is calculated by

$$\mathbf{B}^{-1} (\mathbf{x}^2_3, \mathbf{e}^2)^T = \mathbf{B}^{-1} (1, 2, 3, 0, 2, 0, 1)^T = (2, -2, 0, -1, -2, 0, 2)^T.$$

Table C.3. Simplex Tableau of Iteration 2

	w_{13}	w_{14}	w_{21}	w_{24}	w_{43}	α_1	α_2	RHS	λ^2_3
z	0	0	0	-9	0	28	29	30	10
s_{13}	1	0	0	-1	0	2	1	4	2
s_{14}	0	1	0	2	0	-6	-4	4	-2
s_{21}	0	0	1	1	0	-2	-3	3	0
λ^2_2	0	0	0	1	0	-2	-1	0	-1
s_{43}	0	0	0	1	1	-3	-4	3	-2
λ^1_1	0	0	0	0	0	1	0	1	0
λ^2_1	0	0	0	-1	0	2	2	1	(2)

	w_{13}	w_{14}	w_{21}	w_{24}	w_{43}	α_1	α_2	RHS	λ^2_3
z	0	0	0	-4	0	18	19	25	0
s_{13}	1	0	0	0	0	0	-1	3	0
s_{14}	0	1	0	1	0	-4	-2	5	0
s_{21}	0	0	1	1	0	-2	-3	3	0
λ^2_2	0	0	0	1/2	0	-1	0	1/2	0
s_{43}	0	0	0	0	1	-1	-2	4	0
λ^1_1	0	0	0	0	0	1	0	1	0
λ^2_3	0	0	0	-1/2	0	1	1	1/2	1

Iteration 3

Still all $w_{ij} \leq 0$. Thus proceed to solve subproblems to find a candidate to enter the master basis.

Subproblem 1

$$\mathbf{w} - \mathbf{c}_1 = (0, 0, 0, -4, 0) - (2, 3, 4, 1, 2) = (-2, -3, -4, -5, -2).$$

This cost term consists of the following single-commodity minimum cost flow problem.

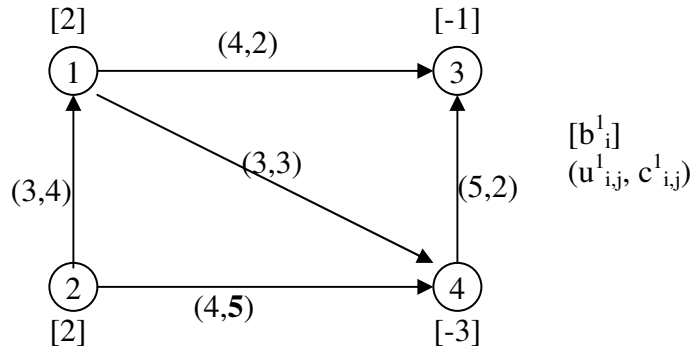


Figure C.7. Network Representation of Subproblem 1 of Iteration 3

The optimal solution is $\mathbf{x}^1_4 = (1, 1, 0, 2, 0)^T$.

$$z^1_4 - c^1_4 = (\mathbf{w} - \mathbf{c}_1)\, \mathbf{x}^1_4 + \alpha_1 = -15 + 18 = 3.$$

Since $z^1_4 - c^1_4$ is greater than zero, λ^1_4 could be a candidate to enter. Continue to subproblem 2.

Subproblem 2

$$\mathbf{w} - \mathbf{c}_2 = (0, 0, 0, -4, 0) - (1, 5, 2, 3, 1) = (-1, -5, -2, -7, -1).$$

We have the following single-commodity minimum cost flow problem.

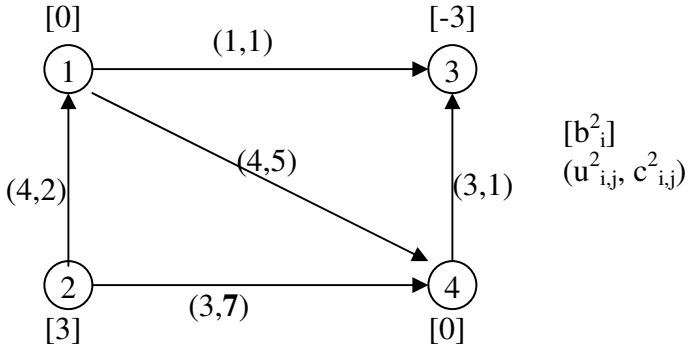


Figure C.8. Network Representation of Subproblem 2 of Iteration 3

The minimum cost optimal solution is $\mathbf{x}^2_4 = (1, 0, 1, 2, 2)^T$.

$$z^2_4 - c^2_4 = (\mathbf{w} - \mathbf{c}_2)\, \mathbf{x}^2_4 + \alpha_2 = -19 + 19 = 0.$$

Since $z^2_4 - c^2_4$ is zero, λ^2_4 can not be a candidate to enter the master problem basis. Thus λ^1_4 is selected. Calculated column for λ^1_4 is

$$\mathbf{B}^{-1} (\mathbf{x}^1_4, \mathbf{e}^1)^T = \mathbf{B}^{-1} (1, 1, 0, 2, 0, 1, 0)^T = (1, -2, 0, -1, -2, 0, 2)^T.$$

Table C.4. Simplex Tableau of Iteration 3

	w_{13}	w_{14}	w_{21}	w_{24}	w_{43}	a_1	a_2	RHS	λ^1_4
z	0	0	0	-4	0	18	19	25	3
s_{13}	1	0	0	0	0	0	-1	3	1
s_{14}	0	1	0	1	0	-4	-2	5	-1
s_{21}	0	0	1	1	0	-2	-3	3	0
λ^2_2	0	0	0	1/2	0	-1	0	1/2	0
s_{43}	0	0	0	0	1	-1	-2	4	-1
λ^1_1	0	0	0	0	0	1	0	1	(1)
λ^2_3	0	0	0	-1/2	0	1	1	1/2	0

	w_{13}	w_{14}	w_{21}	w_{24}	w_{43}	a_1	a_2	RHS	λ^1_4
z	0	0	0	-4	0	15	19	22	0
s_{13}	1	0	0	0	0	-1	-1	2	0
s_{14}	0	1	0	1	0	-3	-2	6	0
s_{21}	0	0	1	1	0	-2	-3	3	0
λ^2_2	0	0	0	1/2	0	-1	0	1/2	0
s_{43}	0	0	0	0	1	0	-2	5	0
λ^1_4	0	0	0	0	0	1	0	1	1
λ^2_3	0	0	0	-1/2	0	1	1	1/2	0

Iteration 4

Still all $w_{ij} \leq 0$. Thus proceed to solve subproblems to find a candidate to enter the master basis.

Subproblem 1

$\mathbf{w} - \mathbf{c}_1 = (0, 0, 0, -4, 0) - (2, 3, 4, 1, 2) = (-2, -3, -4, -5, -2).$

This cost term consists of the following single-commodity minimum cost flow problem.

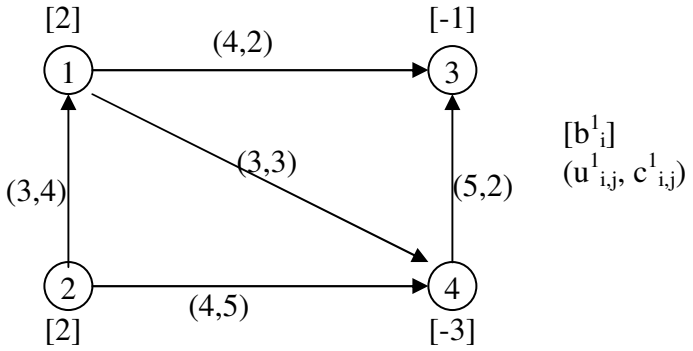


Figure C.9. Network Representation of Subproblem 1 of Iteration 4

The optimal solution is $\mathbf{x}^1_5 = (1, 1, 0, 2, 0)^T$. Notice that there is no change in the optimal solution from the previous iteration. However we have different $z^k_j - c^k_j$ due to the change in α_1 value.

$z^1_5 - c^1_5 = (\mathbf{w} - \mathbf{c}_1)\, \mathbf{x}^1_5 + \mathbf{a}_1 = -15 + 15 = 0.$

Since $z^1_5 - c^1_5$ is zero, λ^1_5 can not be a candidate to enter. Continue to subproblem 2.

Subproblem 2

$\mathbf{w} - \mathbf{c}_2 = (0, 0, 0, -4, 0) - (1, 5, 2, 3, 1) = (-1, -5, -2, -7, -1).$

We have the following single-commodity minimum cost flow problem.

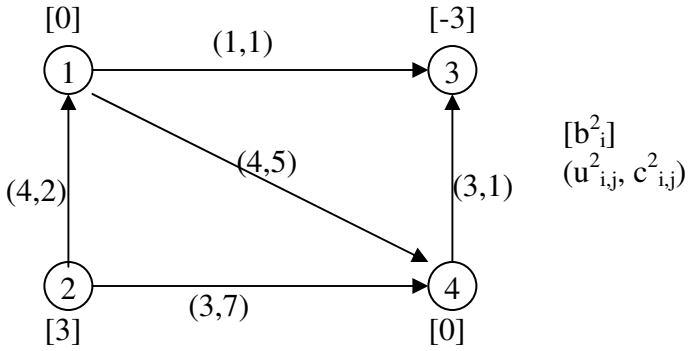


Figure C.10. Network Representation of Subproblem 2 of Iteration 4

The minimum cost optimal solution is $\mathbf{x}^2_5 = (1, 0, 1, 2, 2)^T$.

$$z^2_5 - c^2_5 = (\mathbf{w} - \mathbf{c}_2)\, \mathbf{x}^2_5 + \alpha_2 = -19 + 19 = 0.$$

Since $z^2_5 - c^2_5$ is zero, λ^2_5 can not be a candidate to enter the master problem basis. Thus there is no candidate for further pivoting. Finally, we have the following optimal solution:

$$z^* = 22$$
$$\mathbf{x}^*_1 = \lambda^1_4 \mathbf{x}^1_4 = (1, 1, 0, 2, 0)^T$$
$$\mathbf{x}^*_2 = \lambda^2_2 \mathbf{x}^2_2 + \lambda^2_3 \mathbf{x}^2_3 = (1/2)(1, 0, 1, 2, 2)^T + (1/2)(1, 2, 3, 0, 2)^T$$
$$= (1, 1, 2, 1, 2)^T.$$

C.3. Resource-Directive Decomposition Method

Resources-directive decomposition method views the problem as a capacity allocation problem, in which all commodities are competing for fixed capacity of every arc of the network. It initially allocates capacity to commodities and decomposes the

problem into a set of K independent single commodity problems. At each iteration an allocation is made and K single-commodity minimal cost flow problems are solved. The sum of the capacities allocated to an arc over all commodities is less than or equal to the arc capacity in the original problem. Hence the combined flow from the solutions of the subproblems provides a feasible flow for the original problem. Optimality is tested and the procedure either terminates or a new arc-capacity allocation is developed.

After the artificial variables are added, (C.1)-(C.6) becomes

$$\begin{aligned} \text{minimize} \quad & \sum_k \mathbf{c}^k \mathbf{x}^k + \gamma \sum_k 1 a^k \\ \text{subject to} \quad & \mathbf{A}\mathbf{x}^k + \mathbf{a}^k = \mathbf{b}^k, \ \ k = 1,\ldots,K \\ & \sum_k \mathbf{x}^k + \mathbf{s} = \mathbf{u} \\ & \mathbf{0} \le \mathbf{x}^k \le \mathbf{u}^k, \ k = 1,\ldots,K \\ & \mathbf{a}^k, \mathbf{s} \ge 0 \end{aligned} \tag{C.15}$$

where γ is a large positive scalar, $\mathbf{a}^k$ is a vector of artificial variables, and $\mathbf{1}$ is a vector of ones. An equivalent statement of (C.15) is

$$\begin{aligned} \text{minimize} \quad & \sum_k V_k(y^k) \\ \text{subject to} \quad & \sum_k y^k + s = \mathbf{u} \\ & \mathbf{0} \le y^k \le \mathbf{u}^k, \ k = 1,\ldots,K \\ & s \ge 0 \end{aligned} \tag{C.16}$$

where $V_k(y^k) = \min \{ c^k x^k + \gamma 1 a^k : Ax^k + a^k = b^k, 0 \le x^k \le y^k \} = \max \{ b^k \mu^K - y^k \nu^k : \mu^K A - \nu^k \le c^k, \mu^K \le \gamma 1, \nu^k \ge 0 \}$ by duality theory. We can show the function $V(y^1,\ldots,y^k)$ is

convex. Different resource-directive techniques differ in the manner in which (C.16) is solved. We will see the *tangential approximation method*.

Method of Tangential Approximation. [1]

Let $R_k = \{ (\mu^k, \nu^k): \mu^k A - \nu^k \le c^k, \mu^k \le \gamma 1, \nu^k \ge 0,$ and (μ^k, ν^k) an extreme point $\}$. Then (C.16) may stated as

$$\text{minimize } \sum_k \sigma_k \tag{C.17}$$

$$\text{subject to } \sigma_k \ge b^k\mu^k - y^k\nu^k, \quad \text{all } (\mu^k, \nu^k) \in R_k \text{ and all k} \tag{C.18}$$

$$\sum_k y^k + s = \mathbf{u} \tag{C.19}$$

$$\mathbf{0} \le y^k \le \mathbf{u}^k, \; k = 1,\ldots,K \tag{C.20}$$

$$s \ge 0 \tag{C.21}$$

Suppose $Q_k \subset R_k$. Let z(r) denote the optimal objective value of (C.17)-(C.21) and let z(Q) denote the optimal objective value of (C.17) with Q_k substituted for R^k in (C.18). Then $z(Q) \le z(R)$ provides a lower bound for (C.15).

Resource-Directive Decomposition Algorithm Using Tangential Approximation

Initialization. Set i = 0 and let $y_0 = (y_0^1, \ldots, y_0^k)$ be any element of $\{(y_0^1, \ldots, y_0^k): \Sigma_k y^k \le u, 0 \le y^k \le u^k\}$. Set $Q_k = \Phi$ and $\sigma_k = -\infty$ for each k.

Step 1. Solve Subproblems (Determine upper bound). Solve

$$V_k(y_i^k) = \min c^k x^k + \gamma 1 a^k$$

$$\text{s.t.} \quad Ax^k + a^k = b^k \qquad (\mu_i^k, \text{ dual var.})$$

$$x^k \le y^k \qquad (\nu_i^k, \text{ dual var.})$$

$$0 \le x^k$$

for each k=1,...K.

Step 2. Check Optimality (check Lower bound = Upper bound). $\sum_k \sigma_k = \sum_k V_k(y_i^k)$?

If so, terminate. The optimum is given by $(x_i^1, ..., x_i^k)$ and $(a_i^k,..., a_i^k)$. If not, add (μ_i^k, ν_i^k) to Q_k for each k and continue with step 3.

Step 3. Solve Master Program. Set i = i + 1. Solve

$$\text{Minimize} \quad \sum_k \sigma_k$$

$$\text{subject to} \quad \sigma_k \ge b^k \bar{\mu}^k - y_i^k \bar{\nu}^k \text{ , for each k and all } (\mu^k, \nu^k) \in Q_k$$

$$\sum_k y_i^k + s = \mathbf{u}$$

$$\mathbf{0} \le y_i^k \le \mathbf{u}^k, \; k = 1,\ldots,K$$

$$s \ge 0$$

and return to step 1.

The K subproblems to be solved in step 1 are single-commodity problems and can be solved efficiently by any popular algorithm.

VITA

Dong Hun Kang, son of Sun Kyu Kang and Myoung Soon Kim, was born on July 2, 1965, in Daejon, Korea. He obtained a B.S. and an M.S. degree in industrial engineering from Hanyang University, Korea in 1987 and 1989, respectively. After graduation, he served as an administrative sergeant in the Republic of Korea Army for more than 2 years, and worked as an independent contractor in the database engineering field before he began his Ph.D. program at the University of Michigan in August 1993. While majoring in Human Factors at the University of Michigan, his interest changed to the operations research area, and he transferred to Texas A&M University. In August, 1997, he began his Ph.D. work in operations research in the Department of Industrial Engineering at Texas A&M University. He worked for the Economics & Policy Division of Texas Transportation Institute (TTI) for two years and for the Multimodal Freight Transportation Program of TTI for three years as a graduate assistant. His email address is dhkang@tamu.edu

His permanent mailing address is:

Hyundai Hometown 1-Cha 105-404, Dongchun-Dong,
Yong-In-Si, Kyung-Ki-Do, 449-514
South Korea