

**MULTI-PERIOD MULTI-PRODUCT DISTRIBUTION PLANNING PROBLEMS:  
MODEL-BASED AND NETWORK-BASED APPROACHES**

by

**S. Ahmad Hosseini**

**Submitted to the Graduate School of Engineering and Natural Sciences**

**in partial fulfillment of the requirements for the degree of**

**Doctor of Philosophy**

**in**

**Industrial Engineering**

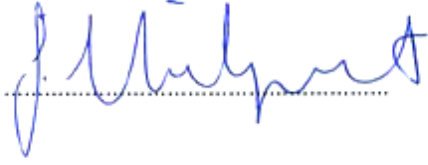
**Sabancı Üniversitesi**

**Fall 2014**

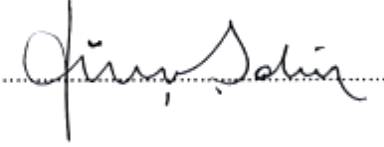
**Multi-Period Multi-Product Distribution Planning Problems:  
Model-Based and Network-Based Approaches**

APPROVED BY:

Assoc. Prof. Dr. Tongu Ünlüyurt  
(Thesis Advisor)



Assoc. Prof. Dr. Güven Şahin  
(Thesis Co-Advisor)



Prof. Dr. Şevket İlker Birbil



Asst. Prof. Dr. Hüsnü Yenigün



Asst. Prof. Dr. Erhun Kundakođlu



DATE OF APPROVAL: **December 30<sup>th</sup>, 2014**

## ACKNOWLEDGEMENTS

About three years ago, I came to Industrial Engineering Department at Sabanci University for PhD studies. Now, my thesis has taken shape and I would like to thank several people for their support.

In the first place, I would like to express my deepest gratitude to Tongu Ünlüyurt and Güven Şahin for supervising my thesis. They have been always so very patient and kind with me and supported my research. At the same time, they allowed independent work and development. I am truly indebted to them for giving me directions by asking the right questions and finding the right answers, and I am also thankful for their trust.

My heartfelt thanks go to Rina Schneur, the 17<sup>th</sup> president of Informs, without whom this thesis would never even have started. Rina aroused my interest in research on this thesis's topic while I was still a master student. Her elegant and inspiring approaches on applying operations research to industry problems in the areas of logistics, network planning, and telecommunication with the emphasis on identifying the combinatorial structure in the problem have all left a deep and eternal impact on me.

I am also grateful to my thesis committee, İlker Birbil, Hüs­ü Yenigün, and Erhun Kundakçiođlu, for their careful proof-reading of different parts of the thesis and for their helpful suggestions for improvement.

I am very fortunate to have been with a set of wonderful people who have made my stay in Istanbul very lively with numerous interesting meetings. These include Ömer, Elif, Berfu, Ceyda, Sezin, Ali, Gökhan, Faran, Öykü, Eqra, Mahir, İpek and others at the department. Things would have been very different without them and I thank all of them for all those great times. Special thanks to Elif and Ali for all those fruitful and funny dinner-coffee discussions. I will always have many fond memories of wonderful food, coffee, cooking, wine tasting, and lame jokes. ☺

Ahmad H.  
December 30<sup>th</sup>, 2014

**© S. Ahmad HOSSEINI**  
**All Rights Reserved**

# **Multi-Period Multi-Product Distribution Planning Problems: Model-Based and Network-Based Approaches**

S. Ahmad Hosseini

Industrial Engineering, PhD Thesis

Thesis Supervisors: Assoc. Prof. Tonguç Ünlüyurt  
Assoc. Prof. Güvenç Şahin

**Keywords:** Linear and Non-Linear Optimization, Block Decomposition, Network Programming, Approximation Algorithms, Scaling Algorithms

## **Abstract**

Dynamic and multiperiod flow problems arise frequently in management applications, communication systems, and process systems engineering with important applications in large-scale production scheduling and time-varying distribution planning. This thesis investigates various multiperiod distribution planning problems, where all problem parameters may change over time and or products.

First, matrix decomposition is exploited through index sets of the models to delineate block structures and to develop some methods that lead to linear programming problems comprising a set of sparse polyhedrals. Considering the sparsity and repeating structure of the polyhedrals algorithmic approaches based on decomposition techniques of block angular and block staircase are proposed aiming to reduce the computational resources required and/or getting rapid near-optimal solutions. The efficiency of the proposed approaches is demonstrated through numerical experiments. Then, we use scaling and approximate optimality together with penalty function method to develop some network-based scaling approximation algorithms.

Our algorithms exploit different ideas including matrix transformation from linear algebra, graph partitioning from graph theory, penalty methods from nonlinear optimization, and scaling and approximation algorithms from network flow theory. Moreover, we analyze the algorithms from both theoretical and practical perspectives. The practical performances corresponding to some electricity transmission distribution networks support the theoretical properties.

# **Çok zaman dilimli ve çok ürünlü dağıtım planlama problemleri: Model bazlı ve ağ bazlı yaklaşımlar**

S. Ahmad Hosseini

Endüstri Mühendisliği, Doktora Tezi

Tez Danışmanı: Doç. Dr. Tonguç Ünlüyurt  
Doç. Dr. Güvenç Şahin

**Anahtar Kelimeler:** Doğrusal ve doğrusal olmayan programlama, blok ayrıştırma, ağ programlama, yaklaşık algoritmalar, ölçekleme algoritmaları

## **Özet**

Dinamik ve çok zamanlı akış problemleri büyük ölçekli üretim planlaması, zamanla değişen dağıtım planlaması, yönetim uygulamaları, iletişim sistemleri ve süreç sistemleri mühendisliği alanlarında sıklıkla ortaya çıkar. Bu tez, tüm problem parametrelerinin zaman veya ürünler üzerinde değişebilir olduğu çok zamanlı dağıtım planlaması problemlerine çözüm yöntemleri geliştirmek üzerinedir.

İlk olarak, endeks setleri üzerinden matris ayrıştırması kullanılarak bloklar elde edilir. Bu merdiven şeklinde ve açısız bloklar kendileri tekrar ettiği için ve boyutları küçük olduğu için etkin bir şekilde ayrıştırma yöntemiyle en iyi veya en iyiye yakın çözümler bulunabilir. Yöntemin etkinliği bazı nümerik örnekler üzerinde sınanmıştır. Daha sonra ölçekleme ve yaklaşık en iyi olma şartları kullanılarak ağ bazlı bir ölçekleme algoritması geliştirilmiştir.

Geliştirilen algoritmalar lineer cebirde kullanılan matris dönüşümlerinden, ağ parçalama algoritmalarına, doğrusal olmayan programlamada kullanılan ceza yöntemlerinden yaklaşık ölçekleme algoritmalarına kadar birçok kavram kullanır. Geliştirilen algoritmaların hem teorik hem pratik özellikleri çalışılmıştır. Nümerik sonuçlar için gerçek hayattan alınan elektrik dağıtım ağı topolojileri kullanılmıştır.

# TABLE OF CONTENTS

<b>1</b>	<b>Introduction, Applications, and Problem Description</b> .....	1
1.1	Introduction and Background of the Study .....	1
1.2	Multiperiod (Multiproduct) Network Flows and Some Applications .....	8
1.2.1	Local Low-Voltage Electricity Distribution Network .....	10
1.2.2	Scheduling, Manufacturing and Planning.....	11
1.2.3	Multi-Site Sequence-Dependent Production Planning Problem .....	12
1.2.4	Crude Oil Distribution Network Problem.....	12
1.2.5	Network Design and Communication Network .....	13
<b>2</b>	<b>A Model-Based Approach and Analysis for Multiperiod Networks</b> .....	15
2.1	Minimum Cost Dynamic Flow Problem in a Multiperiod Network.....	16
2.2	Continuous-Time Model versus Discrete-Time Model.....	19
2.3	Angular and Staircase Structures in Multiperiod Networks.....	20
2.3.1	Block-Angular Structured Systems.....	24
2.3.2	Staircase-Structured Systems .....	24
2.4	A Model-Based Approach for Multiperiod Networks with Storage.....	25
2.5	Slice Modelling .....	30
2.6	Generalized Multiperiod Network Flows (MPDNF with Spoilage).....	31
2.7	Multiperiod Networks with Storage and Spoilage (SS Networks) .....	40
2.8	Examples, Applications, and Testing .....	42
2.9	Summary and Concluding Remarks .....	47
<b>3</b>	<b>A Decomposition-Based Approach for the Multiple-Product Distribution Problems over Time</b> .....	49
3.1	The Problem of Min-Cost Flow on Multiperiod Multiproduct Networks.....	50
3.2	Multiperiod Multiproduct Network Flows with Spoilage (SMMN) .....	52
3.3	A Solution Approach for MM Networks with Storage .....	67

3.4	MM Networks with Storage and Spoilage (SSMM Network Flows).....	79
3.5	An Alternative Approach for MM Networks Having no Period Capacity.....	81
3.5.1	Case 1: MM Networks with Spoilage .....	81
3.5.2	Case 2: MM Networks with Storage.....	84
3.5.3	Case 3: MM Networks with Storage and Spoilage (SS Networks) .....	88
3.6	On Applications and Computational Tuning-Testing.....	88
3.7	Summary and Concluding Remarks .....	97
<b>4</b>	<b>A Penalty-Based Scaling Algorithm for Multiperiod Multiproduct Distribution Planning Problem .....</b>	<b>98</b>
4.1	Penalty Function Method - Transformation Approach.....	99
4.2	Network Flow - Scaling Approach .....	104
4.3	Min-Cost Multiperiod Multiproduct Distribution Problem.....	108
4.3.1	Canceling the Time-Commodity Varying Lower Bounds.....	109
4.3.2	Discrete Time Multiperiod Multiproduct Distribution Problem .....	111
4.4	The Penalty-Based Scaling Approach for MCDF .....	112
4.4.1	Some Notes on the Theoretical Properties.....	120
4.5	Analysis of the Algorithm and Two Specific Implementations.....	129
4.5.1	The Penalty-Based Scaling Approach with Linear Penalty Function .....	131
4.5.2	The Multiperiod Multiproduct Feasibility Problem (MMFP) .....	133
4.6	Summary and Concluding Remarks .....	149
<b>5</b>	<b>Open Problems and Future Research .....</b>	<b>150</b>
<b>6</b>	<b>Bibliography .....</b>	<b>151</b>



# LIST OF FIGURES

Figure 1.1	A typical problem with its original structure and decomposed structure .....	7
Figure 1.2	A simple electrical distribution network .....	11
Figure 1.3	An instance of the crude oil distribution network.....	13
Figure 2.1	Most common structural forms of large-scale problems.....	21
Figure 2.2	A typical GMPDNF at a fixed time period .....	34
Figure 2.3	Sensitivity to the number of time increments.....	44
Figure 2.4	Sensitivity to density.....	44
Figure 2.5	Objective and upper bound progress of decomposition application for a random problem .....	47
Figure 3.1	A typical SMMN at a fixed period $t$ for a fixed product $q$ .....	55
Figure 3.2	A typical MMN with flow storage at nodes .....	68
Figure 3.3	Sensitivity to time increments .....	90
Figure 3.4	Sensitivity to density.....	91
Figure 3.5	Objective and dual bound progress of two case study instances .....	92
Figure 3.6	The relative duality gap observed solving the sample problem (density =8.5, T=17, K=17).....	94
Figure 4.1	Transformation of a multi-source multi-sink single-product multi-period network problem into a single-source single-sink single-product multi-period network problem .....	111
Figure 4.2	Exact optimality for $(i, j)$ .....	120
Figure 4.3	Approximate optimality for $(i, j)$ .....	121
Figure 4.4	A typical diagram of an AC electricity distribution from generation stations to consumers.....	129
Figure 4.5	A general layout of an electricity network .....	130
Figure 4.6	A typical self-similar pattern distribution network.....	137
Figure 4.7	A typical maximum excess progress at the end of each scaling phase of some application of MMNF algorithm (setting: $\delta=1, \rho_u=11111, \varepsilon=0.00001$ ).....	140

# LIST OF TABLES

Table 2.1	Sizes and computational results.....	45
Table 3.1	Sizes and Computational Results for Some MMNs with $ K =1$ ( $T=13, 20, 23, 31, 37, 41, 50$ ).....	91
Table 3.2	Sizes and Computational Results for Some MMNs with $ K =7$ ( $T=13, 20, 23, 31, 37, 41, 50$ ).....	92
Table 3.3	Sizes and Computational Results for Some MMNs with $ N =2 K =2T$ ( $T=3, 5, 7, 11, 13, 15, 17$ ) .....	92
Table 3.4	Computational Resources for Some Large Feasible MMNs .....	93
Table 4.1	Tuning-Testing for Algorithm MMFP (checking) .....	138
Table 4.2	Some Various Settings for MMFP Algorithm.....	139
Table 4.3	Parameter Settings and Tuning-Testing for Some MMNs .....	141
Table 4.4	Computational Results for Some Large Feasible MMNs .....	142
Table 4.5	Computational Results for Some Infeasible MMNs .....	143
Table 4.6	Parameter Settings for Algorithms ‘MMNF’ and ‘MMNF-Linear’.....	144
Table 4.7	MMNF Sensitivity w.r.t. Penalty Parameter Upper Bound ( $\rho_u$ ).....	145
Table 4.8	MMNF-Linear Sensitivity w.r.t. Penalty Parameter Upper Bound ( $\rho_u$ ) .....	145
Table 4.9	MMNF Sensitivity w.r.t. Initial Penalty Parameter Value ( $\rho_0$ ) .....	146
Table 4.10	MMNF-Linear Sensitivity w.r.t. Penalty Modification Rate ( $R$ ) .....	147
Table 4.11	MMNF-Linear Sensitivity w.r.t. Penalty Modification Rate ( $R$ ) .....	148

# Chapter 1

## 1 Introduction, Applications, and Problem Description

### 1.1 Introduction and Background of the Study

Network flow optimization problems arise in a wide variety of important fields, such as transportation, telecommunication, computer networking, financial planning, logistics and supply chain management, energy systems. Significant results have been achieved in both theory and applications of network flow optimization in the past few decades.

Flow variation over time is a very important feature in network flow problems arising in various applications such as traffic control, production systems, communication networks, and financial flows [6][16][29][31][32][34][35][65][80][81]. However, most of studies consider static versions of network planning problems in the sense that parameters do not change over time. Another prominent assumption in network flows is the conservation of flow over the arcs. However, this assumption may also make it hard for many real life applications to model their characteristics. Many network planning problems in real world are time-varying and do not follow such structures; the network structure and problem parameters may be time-dependent [6][16][29][30][32][34][35][36][37][38][39][40][41][80][81]. In addition, it may take a certain amount of time for the flow to traverse an arc [30][31][32][34][57][65][80] and there is no guarantee for the flow to be conserved [3][36]. Therefore, static (traditional) network flow fails to capture the time-varying property. The need for more realistic network models led to the development of *multiperiod* and *dynamic network flow*. They have been applied to a wide variety of applications. In such applications, flow values on arcs are not constant but may change over time and not only the amount of flow to be transmitted but also the time needed for the transmission plays an essential role.

Ford and Fulkerson, for the first time, dealt with the maximum flow problem in discrete time setting and developed a technique that is still widely used [31][32]. The main outcome of their work is the *time-expanded networks*. They show that a maximum flow

over time with a finite *time horizon* can be obtained by computing a maximum static flow in the time-expanded network (or by solving a minimum cost circulation problem on the original network) [6][34]. Since then, several further problems have been analyzed, such as the *quickest*, *minimum cost* or *earliest arrival flows* [6][30][65][80][81]. Ford and Fulkerson add a time dimension to the static network flows to include the *transition times* of the flow along arcs. The network flow with flow transition time is called "*flow over time*". The term "*dynamic flow*" is also used for such kind of network flows. Subsequently several models of dynamic flow problems have been studied by Fleischer [30], Hoppe [34], Aronson [6] and Lozovanu [57]. Aronson [6] concentrates on the *maximum flow* and *transshipment* problems in discrete time with an extensive coverage of applications.

Since multiperiod and dynamic network flows are generalizations of static network flows, it should come as no surprise that multiperiod and dynamic networks also have many interesting practical applications. In these instances, to account properly for the evolution of the underlying system over time, we need to use dynamic/multiperiod network flow models. Multiperiod flow problems find applications in various areas, such as process systems engineering (with essential applications in large production scheduling and multiperiod production planning), information communication technology (ICT), network design and communication network, electricity distribution network, multisite production planning system [5][17][18][28][32][47][48][49][52][55][59][60][61][70][72][73][79][81][82].

This thesis addresses non-simultaneous shipment of commodity (or commodities) from production sites (*sources*) to markets (*sinks*) in time-dependent (or multiperiod) production-distribution networks with deterministic production and demand capacities at the minimum cost over a finite planning horizon while all shipment cost and arc capacities are varying over time and/or commodities. We study, model, and investigate this class of problems by various network models and various solution approaches by including horizon capacities, time-commodity varying capacities (and/or time varying capacities), time-commodity varying costs, and time-commodity varying loss/gain factors over a finite planning horizon. The main focus is on the *minimum cost dynamic flows* (MCDF) on multiperiod networks (and multiperiod multicommodity networks) in which spoilage/storage in arcs/nodes is expected/allowed over time (and or commodity).

In such applications, flow values on arcs are not constant but may change over time because of the dynamic nature of the network. Moreover, flow may not travel through the distribution network with a conservative amount but it may be decreased or increased while circulating in the network on arcs. The third dimension is associated with the storage/waiting capability at nodes. In particular, when routing decisions are being made in one time window in the network, the effects can be seen in other time periods only after a certain time delay. The above mentioned aspects of network flows are not captured by the classic (static) network flow models. This is where dynamic multiperiod network flows come into play. They include those dimensions and therefore provide a more realistic modeling tool for numerous real-world applications. Theorems and efficient algorithms have been developed for static network flow problems as they have been in the focus of interest for many years. The differences from the classical problems make it necessary to devise new techniques, although most of the solution methods eventually reduce the dynamic and multiperiod problems to static ones and then employ existing algorithms.

There are some approaches to address somewhat similar kind of problems, like State-Task Network and Resource-Task Network [59] with important differences with respect to the assumption of continuity or discreteness of the time horizon. Terrazas et al. [84] use temporal and spatial Lagrangean decompositions to solve the multi-site, multiperiod planning problems. In a similar problem, Chen et al. [17] use Lagrangean-based decomposition techniques for solving the temporal decomposition of a continuous flexible process network. They use subgradient methods to solve the decomposed problem. Neiro et al. [66] use temporal Lagrangean decomposition to solve a multiperiod mixed-integer non-linear programming planning problem under uncertainty concerning a petroleum refinery. Mouret et al. [63] present a unified representation and modeling approach for process scheduling problems and introduce four different time representations (by using of priority-slots and order of executions of operations/tasks), and apply to single-stage and multi-stage batch scheduling problems, as well as crude-oil operations scheduling problems.

Those works mostly focus on scheduling problems (or makespan minimization in multipurpose batch plants) or multiproduct planning problem with sequence-dependent changeovers-with emphasis on modelling issues-which are modeled

as MILP problems. The main objective of those works is generally to develop some modelling approaches for scheduling problems in order to facilitate the evaluation of several time representations, or minimize the makespan of multipurpose batch plants, or present temporal and spatial Lagrangean decompositions that allow the independent solution of time periods, production sites, and markets [17][18][28][47][48][52][59][63][64][66][72][82].

This section of the thesis gives a very brief description of dynamic and multiperiod (and multiproduct multiperiod) distribution network flows along with some applications. To this aim, we briefly review all chapters of the thesis, and then point out the main problems of interest and solution approaches. We do not describe the solution methods in detail here and do not give precise mathematical models, but we always give references to the relevant chapters where the interested reader can find the details. All chapters are self-contained.

A very general setting of the problem of interest is presented as

$$\mathbf{Min} \quad \sum_{q \in \mathbf{K}} \sum_{(i,j) \in \mathbf{A}} \int_0^T c_{ijq}(t) x_{ijq}(t) dt + \sum_{q \in \mathbf{K}} \sum_i \int_0^T c_{iq}(t) \sum_j [\lambda_{jiq}(t) x_{jiq}(t) - x_{ijq}(t)] dt, \quad (1.1)$$

$$\sum_j \int_0^T x_{ijq}(t) dt - \sum_j \int_0^T \lambda_{jiq}(t) x_{jiq}(t) dt = v_{iq} \quad \forall i \in \mathbf{V}, \forall q \in \mathbf{K}, \quad (1.2)$$

$$\sum_j \int_0^\theta x_{ijq}(t) dt - \sum_j \int_0^\theta \lambda_{jiq}(t) x_{jiq}(t) dt \leq 0 \quad \forall \theta \in [0, T], \forall i \in \mathbf{V}, \forall q \in \mathbf{K}, \quad (1.3)$$

$$\sum_{q \in \mathbf{K}} \int_0^T x_{ijq}(t) dt \leq u_{ij} \quad \forall (i, j) \in \mathbf{A}, \quad (1.4)$$

$$\sum_{q \in \mathbf{K}} x_{ijq}(t) \leq u_{ij}(t) \quad \forall (i, j) \in \mathbf{A}, \forall t \in [0, T], \quad (1.5)$$

$$l_{ijq}(t) \leq x_{ijq}(t) \leq u_{ijq}(t) \quad \forall (i, j) \in \mathbf{A}, \forall t \in [0, T], \forall q \in \mathbf{K}, \quad (1.6)$$

$$x_{ijq}(t) = 0 \quad \forall (i, j) \in \mathbf{A}, \forall t > T, \forall q \in \mathbf{K}. \quad (1.7)$$

In this setting,  $c_{ijq} : [0, T] \rightarrow \mathbb{R}^+$  and  $c_{iq} : [0, T] \rightarrow \mathbb{R}^+$  represent the non-negative distribution cost function with respect to product  $q$ , and storage/waiting cost function, respectively. Constraints (1.2) involve the flow conservation constraints for each commodity in which  $v_{iq}$  denotes the pre-defined deterministic supply/demand capacities at node  $i$  over the entire time horizon. The flow storage is presented in constraints (1.3). We refer to (1.4) as *horizon capacity* constraints. Horizon capacity of an arc limits the amount of total flow (of all commodities) on that arc throughout the entire horizon planning. Constraints (1.5) represent the maximum possible amount of total flow that can enter  $(i, j)$  at time  $t$ : it is referred to as the *moment capacity* constraint. Constraints (1.6) are the time-commodity varying capacity constraint for each commodity at each time moment. Finally, constraints (1.7) ensure that there should be no flow circulating in the network after the horizon planning. Furthermore, each arc  $(i, j)$  is assigned a time-commodity varying non-negative *gain/loss factor*  $\lambda_{ijq}(t)$  with respect to each time period time  $t$  and commodity  $q$ . When  $x_{ijq}(t)$  units of flow of commodity  $q$  is sent from node  $i$  via arc  $(i, j)$  at time  $t$ ,  $\lambda_{ijq}(t)x_{ijq}(t)$  units of flow arrive at node  $j$  at the same time. If  $\lambda_{ijq}(t) < 1$ , the arc is *lossy*; if  $\lambda_{ijq}(t) \geq 1$  the arc is *gainy* on that time with respect to that commodity.

First of all, the methods used in discrete and continuous time are quite different in the context of multiperiod and dynamic flow problems. In general, there could be more practical solutions for discrete-time multiperiod/dynamic problems, whereas for continuous-time problems one may often find only theoretical results. The usual approach to give practical algorithms for continuous-time network problems is to convert it to discrete time. Chapters 2 and 3 describe a *natural transformation* with *time discretization* for both multiperiod and multiproduct multiperiod problems. In these transformations, we solve the discrete versions of the problems and then prove that the optimal continuous solution can be achieved from the discrete solution by extending the flow values to the unit intervals separating the discrete time instances. Discretization works by choosing a suitable time unit and considering the continuous time as split into discrete time periods.

Chapter 2 analyzes the optimal dynamic shipping problem with time-varying network parameters in multiperiod distribution networks when the network contains only one commodity to ship. It also discusses the generalized multiperiod dynamic network flows where time-varying spoilage on arcs (and/or time-varying storage at nodes) is a key factor for the problem. Furthermore, it introduces a set of capacities, so-called *horizon capacity*, which limits the total flow passing arcs over all periods, and proposes some approaches employing polyhedrals/blocks to obtain optimal/suboptimal solutions for a pre-specified finite planning horizon and to reduce the computational resources required.

Solving large multiperiod problems is usually very memory intensive. Decomposition techniques have some niche areas in large scale primal block angular structured problems. We describe heuristics for partitioning practical large-scale multiperiod planning problems into suitable block structures. Such heuristics are of great importance for decomposing large multiperiod problems into forms that are amenable to decomposition techniques and/or parallel processing to reduce the computational expenses and/or getting a rapid near-optimal solution.

Chapter 3 addresses the most general case of discrete-time minimum cost flow problem on multiperiod multiproduct distribution systems by allowing spoilage and or storage. All network parameters change over time and products. We investigate how suitable block structures can be inferred from the mathematical model of the practical multiperiod multiproduct network planning problems. This chapter describes some reformulation techniques to obtain sparse polyhedrals for problem to be amenable to decomposition approaches and/or parallel processing. We also discuss some special cases of such systems and propose some alternative approaches. The main step of reformulation techniques is based on matrix/graph partitioning by using the index sets.

Even though GAMS/Cplex manages memory very efficiently, the most common difficulty when solving large scale multiperiod planning problems is running out of memory. We show that a set of properly decomposed constraints into the blocks can decrease the computational effort in of solving such large scale planning problems by using decomposition techniques. We show how to reorder the variables and constraints of multiperiod multiproduct systems in order to detect underlying blocks. This is done



by adding dummy variables and nodes to the associated matrix of the multiperiod problem. These dummy elements enable the resulting blocks to have sparse matrices. Therefore, the large sets of constraints will be partitioned into a manageable number of independent blocks of constraints, linked together by relatively few linking variables and coupling constraints (e.g., see Figure 1.1). At the end, we discuss how modern computers can also take advantage of the algorithm's inherent parallelism to efficiently improve the elapsed time to motivate use of such parallel processing and block decomposing.

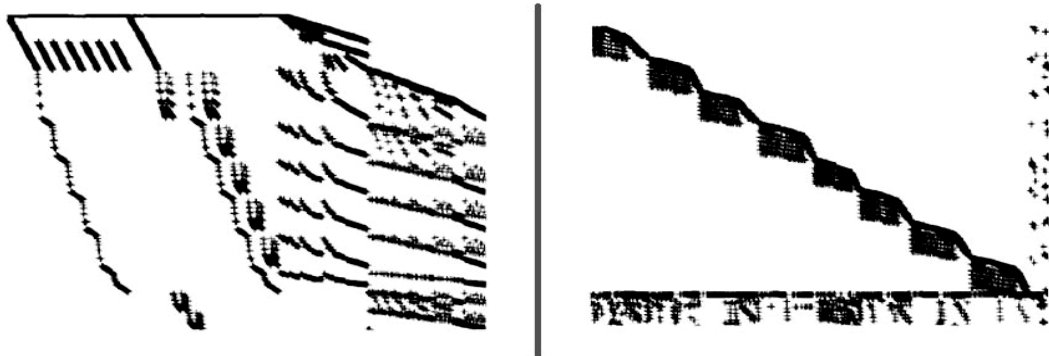


Figure 1.1 A typical problem with its original structure and decomposed structure

Chapter 4 develops a cost-scaling-based approximation algorithm to solve the minimum cost flow problem on multiperiod multiproduct distribution network flow problems with time-commodity varying network parameters. To develop the algorithm, we discuss topics from *Non-Linear Programming*, *Approximation Algorithms*, *Network Flow Theory*, and *Scaling Algorithms*. Having transformed the continuous-time multiperiod multiproduct distribution network problem into the discrete-time version, we discuss how to formulate any such problem with time-commodity varying lower/upper bounds as a problem without lower/upper bounds, as it is necessary for our solution approach.

Such problem formulations usually lead to huge LPs that cannot be handled by a direct application of an LP software. Hence, we associate different *penalty problems* to the original problem and try to solve the penalty problems through *scaling phases* aiming to get a good *approximation solution* in a reasonable amount of time and computational resources. The methods are based on the *Transformation Approach* in *Non-Linear Programming* and are designed to solve the minimum cost and feasibility multiperiod

multicommodity network flow problems. Our algorithm keeps iteratively detecting and shifting time-commodity varying flows around *cycles* at each scaling phase to improve the nonlinear objective function of the associated penalty problem; then, it jumps to the next scaling phase. In order to determine the cycles of interest (*negative cost cycles*), we introduce an auxiliary *time-commodity varying residual network*.

The basis of the methods consists of solving a sequence of penalty problems with an increasing penalty parameter  $\rho$  to find a  $\delta$ -optimal solution to the penalty problem in the sense that the solution is an approximation solution to the original problem. As a result, the influence of some constraints on the auxiliary function of the penalty problem is gradually relinquished and finally removed in the limit. Moreover, we introduce the multiperiod multiproduct feasibility distribution problem in which the objective is to determine whether it is possible to have a production circuit and shipping good within a finite time period. If there is no such dynamic feasible flow, the goal is to determine where and when this infeasibility occurs and the magnitude of the infeasibility. Based on this information, the decision maker may be able to get rid of the infeasibility by providing the necessary budget for creating more capacity.

We analyze the algorithms from both theoretical and practical perspectives using many instances corresponding to some real electricity transmission-distribution networks from our case study and many random instances. The practical performances support the theoretical properties we derive.

Chapter 5 closes this thesis with more promising areas for further directions of research.

## **1.2 Multiperiod (Multiproduct) Network Flows and Some Applications**

Although network flow theory is one of the younger branches of mathematics, it is fundamental to a number of applied fields, including operations research, computer science, and social network analysis. Networks are pervasive and arise in numerous application settings. Physical networks, which are the most readily identifiable classes of networks, arise in many applications in many different types of systems: communications, hydraulic, ecology, electronic, and transportation [3][9][11][15].

In graph theory, a *network flow* is a directed graph  $G := G(V, A)$  with vertex set  $V$ , edge set  $A$ , where each edge has a capacity and each edge receives a flow. The amount of flow on an edge cannot exceed the capacity of the edge. Often in Operations Research, a *directed graph* is called a network, the vertices are called *nodes* and the edges are called *arcs*. A flow must satisfy the restriction that the amount of flow into a node equals the amount of flow out of it, except when it is a *source*, which has more outgoing flow, or *sink*, which has more incoming flow [9][11]. The *capacity*  $u_{ij}$  of an edge  $(i, j)$  can be thought of as the maximal amount of some commodity (such as water, gas, electrical energy, number of cars, bits of information, etc.) that can be transported from station  $i$  to  $j$ , along the edge  $(i, j)$ . Flows can pertain to people or material over transportation networks or electricity over electrical distribution systems. For any such physical network, the flow coming into any intermediate node needs to equal the flow going out of that node. This conservation constraint was formalized as *Kirchhoff's current law*.

Network flows find many applications in many real life problems. Electrical and power networks bring lighting and entertainment into our homes Telephone networks permit us to communicate with each other almost effortlessly within our local communities and across regional and international borders [1][3][4][33][55]. National highway systems, rail networks, and airline service networks provide us with the means to cross great geographical distances to accomplish our work, to see our loved ones, and to visit new places and enjoy new experiences [2][3][9][23]. Manufacturing and distribution networks give us access to life's essential food stock and to consumer products [5][8][32][35][87]. Computer networks, such as airline reservation systems, have changed the way we share information and conduct our business and personal lives [3][9]. In all of these problem domains, and in many more, we wish to efficiently move some entity (electricity, a consumer product, a person or a vehicle, a message etc.) from one point to another through an underlying network both to provide good service to the users of the network and to use the underlying transmission facilities effectively.

The applications we have considered offer only a very brief glimpse of the wide-ranging practical importance of network planning problems; although our discussion of applications in this section is limited, it provides at least one example of the network

models related (or similar) to multiperiod/time-varying flow problems that we have been dealing with. There are, of course, other applications which are not mentioned here, especially, for network flow problems with a dimension of time (time-varying) whose structure is more general. Many other applications including Ecology, Fluid Dynamics, Gas Pipeline Simulation, Road Networks, Survey Design, Optimal Energy Policy, Image Segmentation, Hydraulic Engineering Systems, Electric Distribution Systems, Import and Export Models, Material Requirement Planning (MRP) can be found in the literature [1][14][19][46][49][56][60][61][62][68][69][70][71][73][79].

### **1.2.1 Local Low-Voltage Electricity Distribution Network**

This is actually the application that the author has been mostly dealing with. We have used this kind of networks and applied our solution procedures to instances of a model of an electricity-distribution (transportation) network. This sort of applications are usually a distribution network that is a local low-voltage (LV) part of the electricity system that connects the customers to the long-distance high-voltage transmission system which, in turn, connects to generating plants (see Chapter 4). The distribution network is viewed as connecting to the transmission system, via a substation, at a single point or source (it may connect to several points) [1][4][35][36]. In cities and large towns, standardized LV distribution cables form a network through link boxes. Some links are removed, so that each distributor leaving a substation forms a branched open-ended radial system. The standard 3-phase 4-wire distribution voltage level is 220/400V. However, LV systems are being converted to the latest IEC standard of 220/400V nominal (IEC 60038) [1][4]. Low-voltage and medium-voltage distribution substations, mutually spaced at approximately 500-700 meters, are typically equipped with:

- A 3-way or 4-way MV switchboard (often made up of incoming and outgoing load-break switches) and two MV circuit-breakers or combined fuse/ load-break switches for the transformer circuits.
- One or two 1,000k VA MV/LV transformers.
- One or two (coupled) 6-way or 8-way LV 3-phase 4-wire distribution fuse boards, or circuit-breaker boards, control and protect outgoing 4-core distribution cables, generally referred to as “*distributors*”.

The output from a transformer is connected to the LV busbars via a load-break switch, or simply through isolating links. In densely-loaded areas, a distributor is laid to form a network, with one cable along each pavement and 4-way link boxes located in manholes at street corners, where two cables cross [1][4][46].

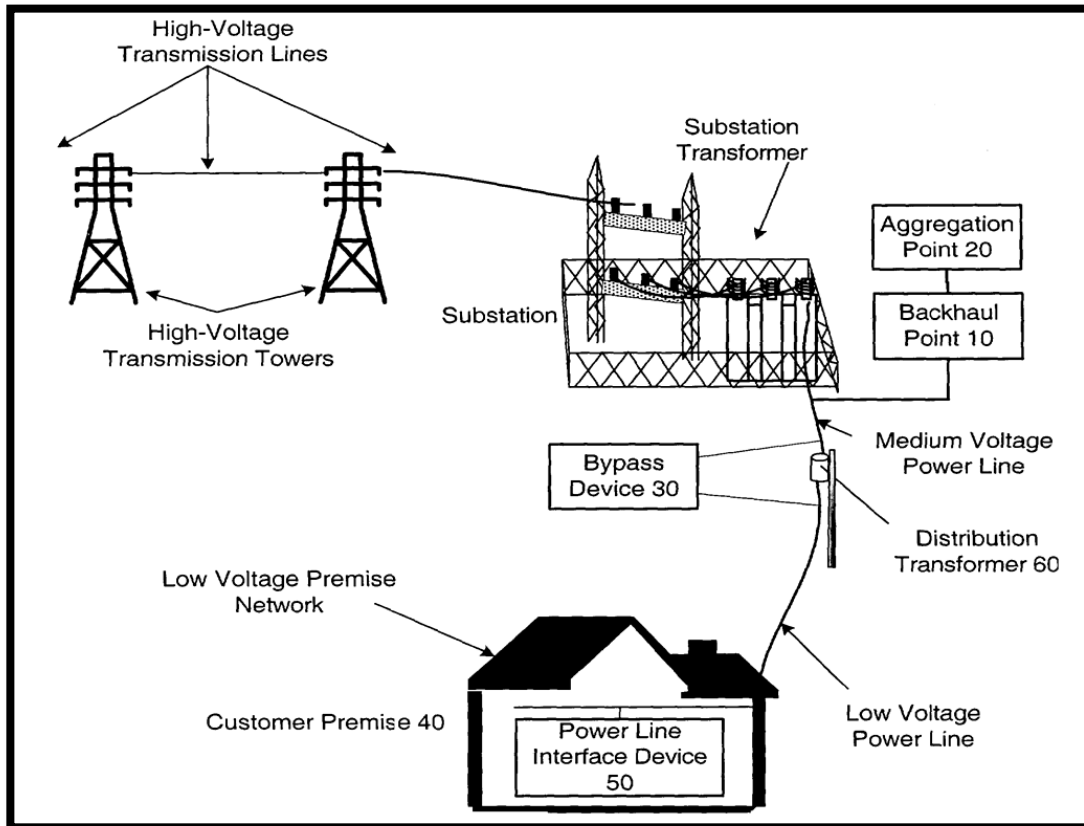


Figure 1.2 A simple electrical distribution network

### 1.2.2 Scheduling, Manufacturing and Planning

In these applications, time comes into play and the network is often partitioned into time steps. Multiperiod multi-item production scheduling problems are common in practice, and are widely considered in management science literature. The problem is to pass the products through several stages of production and shipment from raw materials to end use. At each stage and in each period, there are interrelated decisions about lot-sizing, timing, and stockpiling to be made. Zahorik et al. [87] considered a special case of this problem, in which each of many items went through the same set of production steps in the same sequence.

This problem fits industrial situations such as production, of different styles of chairs, steel pipes of different sizes, air conditioners of different sizes, etc. The nodes in this model represent different time periods and production stages, and arcs represent the possibility of a product to move directly from one node to another. Such problems can be viewed as networks. The constraints on the total production and inventory are the bundle constraints. Thus, using the method described in this study such problems may be solved as multiperiod multiproduct network flow problems.

### **1.2.3 Multi-Site Sequence-Dependent Production Planning Problem**

The optimal planning of a network of manufacturing sites and markets is a complex problem. It involves assigning which products to manufacture in each site (at each time period), how much to ship to each market and how much to keep in inventory to satisfy future demand. Each site has different production capacities and operating costs, while demand for products varies significantly across markets. Production and distribution planning is concerned with mid to long-term decisions usually involving several months, adding a temporal dimension to the spatial distribution given by the multi-site network. The production of each product can involve a setup or cleaning time that in some cases is sequence dependent. This planning problem turns out to be a mixed-integer linear programming (MILP) problem when setups and sequence-dependent transitions are to be included in the problem's assumptions. The computational expense of solving such large-scale MILP problems will be decreased by using decomposition techniques [47][52][66][84].

### **1.2.4 Crude Oil Distribution Network Problem**

As mentioned, dynamic flow problems can be used to model a variety of real world problems that arise in traffic control, production systems, communication networks (e. g., the internet), and pipeline systems for transporting. Here, a problem of pumping crude oil around a crude oil distribution network is illustrated to motivate the study of dynamic networks.

A crude oil distribution system is considered as the essential part of an oil supply chain, and the management of this part can critically affects the performance of the crude oil

supply chain. Traditionally, this system was managed without much assistance of scientific methods [14][35][65]. A large oil company operates more than refineries, which process several million barrels of crude oil every day. Due to the high transportation costs of barrels, implementing scientific methods instead of traditional ones can dramatically reduce total cost and improve customer satisfaction.

A crude oil network links a number of production units to consumption centers (refineries and export terminals) by pipelines. There also are intermediate pump stations and storage tanks next to them. A decision support system was developed for a world-wide oil supply chain by using discrete event simulation and optimal control. Although simulation is a powerful tool, it misses the optimization potential. However, the oil transportation system can be modeled as a dynamic network flow problem as time is the most important parameter in such transportation-distribution networks [65].

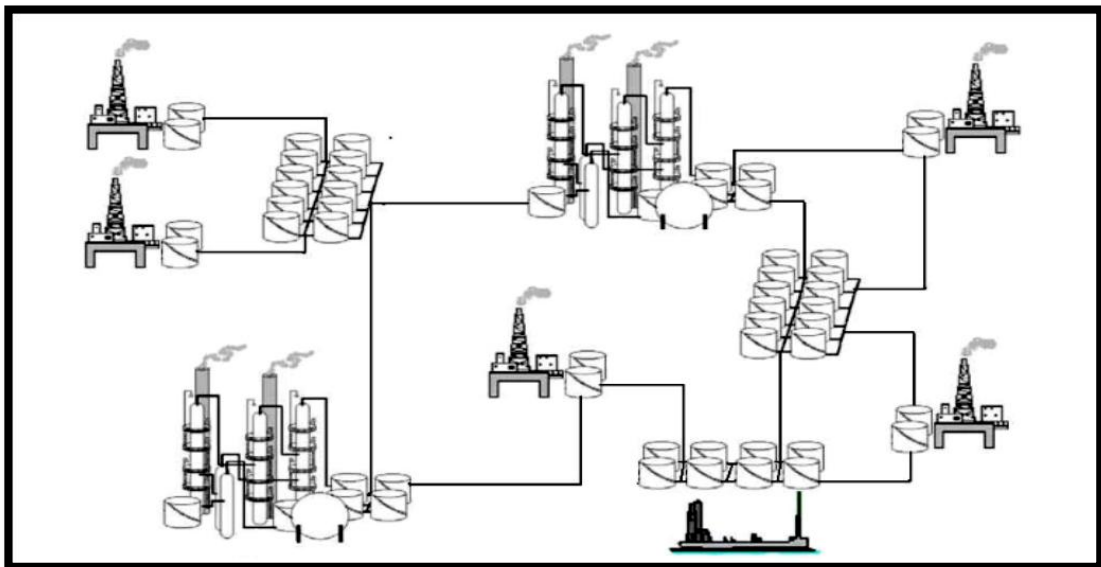


Figure 1.3 An instance of the crude oil distribution network [65]

### 1.2.5 Network Design and Communication Network

Multiproduct (multiperiod) network models can also be used in communication networks [3][55][78]. In a *communication network*, the nodes may represent ‘origin’ and destination for messages and the arcs may represent ‘transmission lines’ or ‘*communication channels*’. Similarly, in a computer communication network, the nodes may represent storage devices or computer systems. The supplies and demands

correspond to the transmission rates between nodes. Products may represent messages between pairs of nodes or messages from each origin to all of its destinations. Each transmission line has a fixed capacity over time periods which may be increased at a certain cost per unit.

There could be two basic applications for a *communication network design model*. In the first one, the objective is to determine the capacity of the network that satisfies the demand at the minimum cost over time. In the second one, a network with fixed arc capacities already exists, and a minimum cost routing is desired. Multiperiod multiproduct solution approaches can be used to determine the routing of circuits and construction of additional arc capacities in a telecommunication network satisfying forecasted circuit requirements at minimum cost.

Moreover, the *reliability communication network problem* can also be considered as a multiproduct multiperiod maximum flow problem. The motivation for this problem comes from the need to improve the reliability and flexibility of public communication networks. At the periods of failure, the communication between a number of origins and destinations are blocked. The problem of how to restore the communication over the same/future time periods would be formulated and solved as a multiperiod multiproduct maximum flow problem. A fixed amount of flow requirements between each pair of nodes is usually assumed over the time periods. The objective is to maximize the total amount of assigned flows over time without exceeding the channel capacities [3].



# Chapter 2

## 2 A Model-Based Approach and Analysis for Multiperiod Networks

The aim of this chapter is to address a general class of multiperiod distribution network problems where time-varying spoilage on arcs or and storage in nodes are inevitable. Having a set of capacities, so-called horizon capacity which limits the total flow passing arcs over all periods, the optimal dynamic shipping problem with time-varying network parameters is investigated. We propose some approaches employing polyhedrals/blocks to obtain optimal/suboptimal solutions for a pre-specified planning horizon.

Our method describes some reformulations based on polyhedrals that lead to LP problems comprising a set of sparse subproblems. Considering the sparsity and repeating structure of the polyhedrals, algorithmic approaches based on decomposition techniques of block angular and block staircase are proposed to handle the global problem aiming to reduce the computational resources required. While the original description of the algorithm was motivated by its reduced memory usage, modern computers can also take advantage of the algorithm's parallelism. This is because the Dantzig-Wolfe method is inherently parallel and can be implemented to take advantage of clusters of machines or multiple cores on a single machine.

The existence and process of identifying such block structures is a prerequisite for decomposition methods to be considered as practical optimization techniques. It will always be the step which requires the most involvement from the practitioner, as success at identifying block structures is essentially a mixture of practical experience and trial and error (if not a systematic approach is applied). In this chapter, we describe a number of block structures, show how they are defined by a partition of the entities, and shortly discuss a various ways to identify them in multiperiod planning problems to be exploited by decomposition techniques.

The results of this chapter and a specific version of the problem are published in Hosseini et al. [36][41].

## 2.1 Minimum Cost Dynamic Flow Problem in a Multiperiod Network

As discussed in previous chapter, there are plenty of relevant decision making problems in practice that can be formulated as optimization models on dynamic or multiperiod networks. Furthermore, important characteristics of real-world networks, such as arc costs and capacities, demands and supplies etc., may be subject to fluctuations over time. Consequently, also flow values on arcs can change over time. On the other hand, many applications do not obey flow conservation assumption.

In our setting in this chapter, each arc has a *positive time-varying factor* associated with it representing the fraction of flow that remains when it is sent at a specific time period. We study a certain type of dynamic networks, *multiperiod dynamic network flows*, containing *horizon capacities* and *loss/gain factors* over a pre-specified time horizon  $T$ . This study deals with *minimum cost dynamic flow* (MCDF) on *generalized multiperiod dynamic network flow* (GMPDNF), in which spoilage/storage in arc/nodes is expected/allowed. Each arc will be assigned a non-negative time-varying gain/loss factor, a non-negative time-varying capacity function, a non-negative horizon capacity, and a non-negative time-varying cost function.

Our problem is dealing with non-simultaneously time-discrete shipping commodity/energy from sources to sinks in a transportation network, such that no capacity conditions are violated, and this time-dependent shipping should optimally happen in a pre-defined planning horizon, and to this aim we propose some simple, efficient LP models aiming to develop polyhedral-based approaches. We present some model-based approaches for GMPDNF problems, which propose some reformulations based on polyhedrals that lead to LP problems comprising a set of subproblems with exceptional structure. Considering the repeating structure of the subproblems, algorithmic approaches based on decomposition techniques of block-angular and block-staircase are proposed to handle the global problem aiming to reduce the computational resources required. The structural similarity of the subproblems helps us use decomposition techniques to improve the computational efficiency. Our approaches rely on an appropriate defining of polyhedral sets. They show that the MCDF problem on a multiperiod network can be reduced to some linear programs, whose special structures permit efficient computation of its solution.

As mentioned in the previous chapters, in contrast to static flows, a dynamic flow specifies the flow rate entering an arc for each time period/moment. Let  $G = (V, A)$  be a directed graph with node set  $V$ , arc set  $A$ , and integral time horizon  $T$ . Each arc  $(i, j)$  has an associated non-negative time varying capacity  $u_{ij}(t)$  with it, which limits the rate of flow entering  $(i, j)$  at any moment, a *horizon capacity*  $u_{ij}^T$  which represents the maximum amount of flow which can be carried on arc within the entire time horizon  $T$ , and a non-negative *transit time*  $\tau_{ij}$ . Transit time measures the time a unit flow takes to get from the tail the head of an arc. A dynamic flow  $x(t)$  satisfies the supplies and demands if by time  $T$  the net flow into each sink equals the demand at the sink and the net flow out of each source equals the supply at the source:

$$\sum_j \int_0^T x_{ij}(t) dt - \sum_j \int_0^T x_{ji}(t - \tau_{ji}) dt = v_i \quad \forall i \in V, \quad (2.1)$$

where  $v_i$  is the pre-defined supply/demand of source/sink/intermediate node  $i$  over the entire time horizon. Given  $G$ ,  $x(t): A \rightarrow R^+$  is called a *dynamic feasible flow* if it satisfies (2.1) - (2.4):

$$\int_0^T x_{ij}(t) dt \leq u_{ij}^T \quad \forall (i, j) \in A, \quad (2.2)$$

$$0 \leq x_{ij}(t) \leq u_{ij}(t) \quad \forall (i, j) \in A, \quad \forall t \in [0, T], \quad (2.3)$$

$$x_{ij}(t) = 0 \quad \forall (i, j) \in A, \quad \forall t > T - \tau_{ij}, \quad (2.4)$$

where  $x_{ij}(t)$  is the amount of flow passing arc  $(i, j)$  at time moment  $t$ . Constraint (2.1) denotes the flow conservation constraint. Condition (2.2) specifies the upper limit on the total flow that can be sent from node  $i$  along arc  $(i, j)$  over  $[0, T]$ , and condition (2.3) represents the maximum possible amount of flow that can enter  $(i, j)$  at time  $t$ . The last condition emphasizes that flow can be traveling in the network until the end of pre-specified time horizon.

In the traditional min-cost flow problems, there is a capacitated network, and the aim is to send a commodity from some sources to some sinks without exceeding the arc

capacity limits at the minimal cost disregarding the time dimension. The minimum cost dynamic flow (MCDF) problem involves non-simultaneously shipping commodity/commodities from sources through intermediate nodes to sinks in a (single) network, such that the total amount of flow going through each arc (path) does not exceed its capacities (time-varying and horizon capacities), and this shipping should optimally take place in a pre-defined planning horizon  $T$ .

Hence, having a continuous cost function  $c_{ij} : [0, T] \rightarrow \mathbb{R}^+$ , MCDF problem is a decision problem where we are trying to find a feasible dynamic flow satisfying (2.7)-(2.10) minimizing the following objective function:

$$\sum_{(i,j) \in A} \int_0^T c_{ij}(t) x_{ij}(t) dt . \quad (2.5)$$

Therefore, we may formulate the MCDF problem, in continuous-time model as follows:

$$\mathbf{Min}_{x_{ij}(t)} \sum_{(i,j) \in A} \int_0^T c_{ij}(t) x_{ij}(t) dt , \quad (2.6)$$

$$\sum_j \int_0^T x_{ij}(t) dt - \sum_j \int_0^T x_{ji}(t) dt = v_i \quad \forall i \in V , \quad (2.7)$$

$$\int_0^T x_{ij}(t) dt \leq u_{ij}^T \quad \forall (i, j) \in A , \quad (2.8)$$

$$0 \leq x_{ij}(t) \leq u_{ij}(t) \quad \forall (i, j) \in A , \quad \forall t \in [0, T] , \quad (2.9)$$

$$x_{ij}(t) = 0 \quad \forall (i, j) \in A , \quad \forall t > T . \quad (2.10)$$

The model presented in (2.6)-(2.10) lets no storage at nodes. It may be necessary that the flow waits at some intermediate nodes until it can continue on an arc, as it appears in many applications such as batch process scheduling, traffic routing, evacuation planning, energy transmission, inventory, and telecommunications [3][30][31][32][34][35]. This leads to a slightly different notion of flow conservation. Storage means that the flow conservation is not satisfied at each time instance because the amount of flow arriving at a (intermediate) node at a given time can be different

from the amount of flow that leaves the node at that time. If we let the set of vertices  $V$  be divided into three subsets  $V_S, V_I, V_D$  comprising source, intermediate, and sink nodes, respectively, we can state the flow conservation constraints as following. In this case,  $x(t): A \rightarrow \mathbb{R}^+$  is a *dynamic feasible flow* if it satisfies constraints (2.7)-(2.10) and (2.11) as well

$$\sum_j \int_0^\theta x_{ij}(t) dt - \sum_j \int_0^\theta x_{ji}(t) dt \leq 0 \quad \forall i \in V \setminus V_S, \quad \forall \theta \in ]0, T[. \quad (2.11)$$

As flow travels through the network, we may allow limited (or unlimited) flow storage at nodes, but prohibit any deficit by constraint (2.11). As before, all demands must be met, flow must not remain in the network after time  $T$  and each source/sink must not exceed its supply/demand.

## 2.2 Continuous-Time Model versus Discrete-Time Model

Time may pass in discrete increments or continuously. In discrete-time models we look at the network at times  $t=0,1,2,\dots,T-1$  by choosing a suitable unit. In practical models time can be discretized, thus converting continuous flow models to discrete ones. The Continuous-time version looks for flow distributed continuously over time within period  $[0, T]$  while the discrete one is looking for the flow rates over discrete periods. On the other hand, the choice of the time unit has a considerable impact on the complexity of the problem. To be able to use general notations that are valid for both discrete and continuous models, we denote the time domain by  $T$ , thus  $\mathbb{N} = \{0,1,\dots,T-1\}$  in a discrete-time model and  $[0, T]$  in a continuous-time model. It might be of use to have a short discussion on the relationship between continuous-time and discrete-time dynamic flows in multiperiod networks. There is a natural transformation of continuous dynamic flow  $\bar{x}$  with integral time horizon  $T$  to a discrete flow  $x$  of the same horizon, and vice versa. To this end, let  $x_{ij}(t)$  be the total amount of flow sent into arc  $(i, j)$  during time interval  $[t, t+1[$ , i.e.,

$$x_{ij}(t) := \int_t^{t+1} \bar{x}_{ij}(\xi) d\xi.$$

We define capacity  $u_{ij}(t)$  and cost  $c_{ij}(t)$  by

$$u_{ij}(t) := \int_t^{t+1} \bar{u}_{ij}(\xi) d\xi, \quad \text{and} \quad c_{ij}(t) := \bar{c}_{ij}(\xi_t),$$

where  $\xi_t \in ]t, t+1[$ ,  $t \in \mathbb{N}$ , and  $\int_t^{t+1} \bar{c}_{ij}(\xi) \bar{x}_{ij}(\xi) d\xi = \bar{c}_{ij}(\xi_t) \int_t^{t+1} \bar{x}_{ij}(\xi) d\xi$ .

The last equality holds true because  $c$  and  $\bar{x}$  are non-negative continuous functions (see, for example, [10][74]). The flow  $x$  is feasible. For any integral time step  $t$  and time horizon  $T$  we can bound  $x_{ij}(t)$  as follows:

$$x_{ij}(t) := \int_t^{t+1} \bar{x}_{ij}(\xi) d\xi \leq \int_t^{t+1} \bar{u}_{ij}(\xi) d\xi = u_{ij}(t) \quad \text{and} \quad \sum_{t \in \mathbb{N}} x_{ij}(t) = \sum_{t=0}^{T-1} \int_t^{t+1} \bar{x}_{ij}(\xi) d\xi = \int_0^T \bar{x}_{ij}(\xi) d\xi \leq u_{ij}^T,$$

where the above inequalities hold because  $\bar{x}$  is feasible. It is easy to verify that flow conservation constraints hold and  $x$  satisfies all such constraints

$$\begin{aligned} \sum_j \sum_{t \in \mathbb{N}} x_{ij}(t) - \sum_j \sum_{t \in \mathbb{N}} x_{ji}(t) &= \sum_j \sum_{t=0}^{T-1} \int_t^{t+1} \bar{x}_{ij}(\xi) d\xi - \sum_j \sum_{t=0}^{T-1} \int_t^{t+1} \bar{x}_{ji}(\xi) d\xi, \\ &= \sum_j \int_0^T \bar{x}_{ij}(\xi) d\xi - \sum_j \int_0^T \bar{x}_{ji}(\xi) d\xi = \int_0^T v_i(\xi) d\xi = \sum_{t=0}^{T-1} v_i(t) = v_i. \end{aligned}$$

In this transformation, the flow cost is preserved.

$$\begin{aligned} \sum_{(i,j) \in \mathbb{A}} \int_0^T \bar{c}_{ij}(\xi) \bar{x}_{ij}(\xi) d\xi &= \sum_{(i,j) \in \mathbb{A}} \sum_{t=0}^{T-1} \int_t^{t+1} \bar{c}_{ij}(\xi) \bar{x}_{ij}(\xi) d\xi \\ &= \sum_{(i,j) \in \mathbb{A}} \sum_{t=0}^{T-1} \bar{c}_{ij}(\xi_t) \int_t^{t+1} \bar{x}_{ij}(\xi) d\xi = \sum_{(i,j) \in \mathbb{A}} \sum_{t \in \mathbb{N}} c_{ij}(t) x_{ij}(t). \end{aligned}$$

### 2.3 Angular and Staircase Structures in Multiperiod Networks

Certain structural forms of large-scale problems reappear frequently in applications, and large-scale systems theory concentrates on the analysis of these problems. In this context, structure means the pattern of zero and nonzero coefficients in the constraints;

the most important such patterns are depicted in the following figure. Several large-scale problems including any with block angular or near-network structure become much easier to solve when some of their constraints are removed. The decomposition method is one way to attack these problems. It essentially considers the problem in two parts, one with the ‘easy’ constraints and one with the ‘complicating’ constraints [21][22][25][26][76][83][86]. It uses the shadow prices of the second problem to specify resource prices to be used in the first problem. This leads to interesting economic interpretations, and the method has had an important influence upon mathematical economics. It also has provided a theoretical basis for discussing the coordination of decentralized organization units, and for addressing the issue of transfer prices among such units [83][85].

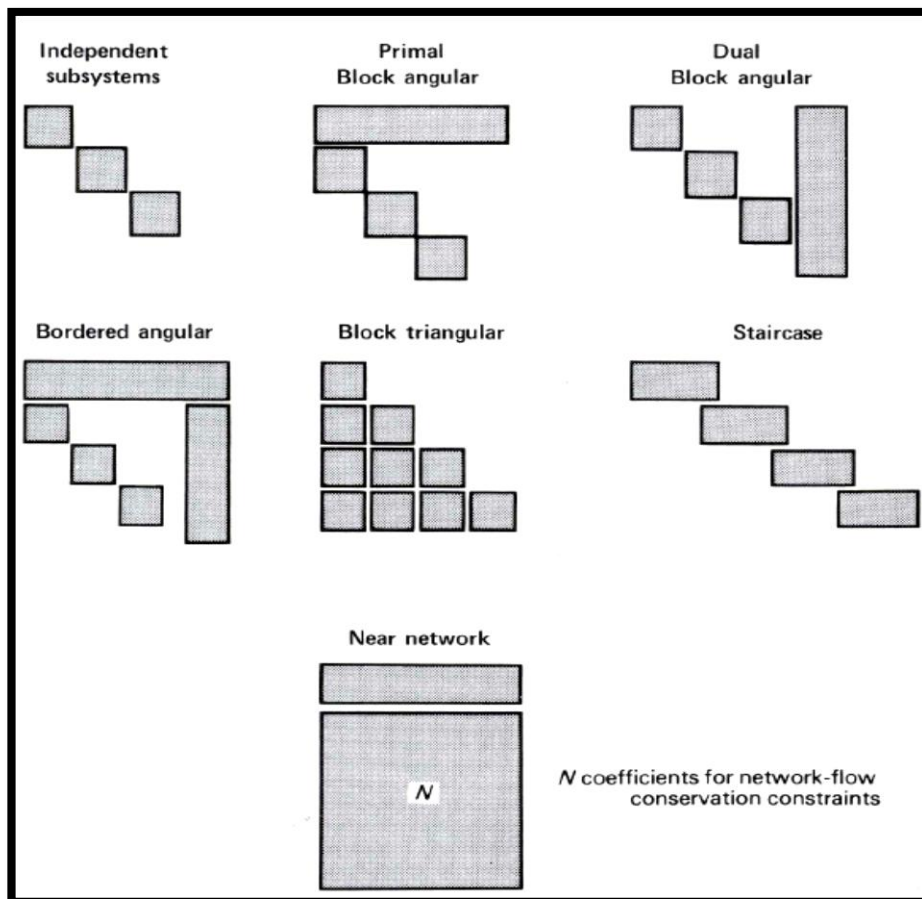


Figure 2.1 Most common structural forms of large-scale problems

DW decomposition is a solution method for the class of LP problems in which the constraint matrix,  $\mathbf{A}$ , exhibits the primal *block angular structure*. DW relies on delayed column generation for improving the tractability of large-scale linear programs. For

most linear programs solved via the revised simplex algorithm, at each step, most columns (variables) are not in the basis. In such a scheme, a master problem containing at least the currently active columns (the basis) uses a subproblem or subproblems to generate columns for entry into the basis such that their inclusion improves the objective function. To apply it, there are two major steps. The formulation step, which is carried out implicitly, is to reformulate the original problem into a form amenable to decomposing: the reformulated problem is called the master problem. The algorithmic step is to solve the master problem using a specialized algorithm, which treats a much smaller, restricted master problem and a set of small pricing problems to obtain a solution to the full master problem, and thus the original problem. The pricing problems are based on the block structure present in the original problem.

The existence and process of identifying block structures is a building block for our modelling approach to exploit decomposition methods as practical solution approaches. Here, we are going to have a short discussion on some general principles and guidelines for identifying block structures in LPs, and subsequently MDNF problems, by considering the algebraic formulation and some other insights. One method is to think about a plot of the non-zero elements in the constraint matrix of a problem instance. It might be supposed that it would be easy to identify a block structure from a constraint matrix of an arbitrary problem instance, but this is not the case [83][85][86]. The rows and columns almost invariably do need permuting in order to realize a block structure. Natural formulations of LP problems group the entities by function rather than by index, i.e., all variables associated with production for all products occur together, rather than all variables (production, storage etc.) associated with a particular product. Even with a relatively small problem instance, it is hard to pick out a block structure visually from a plot of the non-zero elements, when the rows and columns are not arranged to expose it. A second approach could be to pose the problem of identifying block structure as an optimization problem in itself, and use optimization software to identify a structure.

Various studies have been conducted into this area, but with limited success. One argument against such an approach is that any potential benefit obtained by using a decomposition method to solve the original problem is more than outweighed by the extra effort required to identify the block structure in the first place. This argument loses some of its weight if the same problem is to be solved many times with different data



sets, as the structure obtained can be used for different data sets (provided they are of the same size), but the longer it takes to solve the block structure problem, the greater the gain needed to compensate in using decomposition methods to solve the original problem [76][83][86].

A further difficulty is that if the structure obtained does not correspond to an underlying structure in the algebraic model, then it will not scale and will be inapplicable to the same model with different sized data sets. It is our opinion and that of others [53][83][85][86] that the criteria for a good block structure are too ill-defined for the result of a block structure optimization problem to be useful. Criteria such as minimizing the size of the global blocks or maximizing the number of subsystem blocks are used, but these are poor proxies for the ultimate objective of a block structure that results in a good performance of a decomposition method.

Maybe, one of the other efficient ways to obtain a block structure for an LP model for use with a decomposition method is to study the algebraic formulation and generate block structures from the index sets of the model. With a little practice it is not difficult, and with good modelling support tools one can try a number of alternatives to find one that works well. One may consult Borndorfer and Ferreira [13], Ferris et al. [25][26], Kernighan and Lin [53], and Weil and Kettler [86] to get more information on this matter. Structures that rely on the algebraic formulation, rather than a specific data instance, have the key advantage of being scalable, that is, applicable to any data instance of the problem. The key point is to consider how production/usage/storage at one period is affected by production/storage of the previous time step.

Given an arbitrary matrix  $A$ , think of a pair of partitions of the rows and columns: suppose the rows are divided into  $T+1$  sets  $\{I_0, I_1, \dots, I_T\}$  and the columns into  $T'+1$  sets  $\{J_0, J_1, \dots, J_{T'}\}$ . The rows (respectively columns) in each set  $I_t$  (resp.  $J_{t'}$ ) need not be adjacent in the matrix. The partitions of the rows and columns clearly impose a partition on the matrix elements. We say that the elements are partitioned into blocks  $A_{t,t'} = \{a_{t,t'} \mid t \in I_t \text{ and } t' \in J_{t'}\}$ , and we refer to the partition of the matrix elements as a block structure. A block structure is thus defined by a pair of partitions on the rows  $\{I_0, I_1, \dots, I_T\}$  and columns  $\{J_0, J_1, \dots, J_{T'}\}$ . Given the constraint matrix  $A$  from a large sparse

LP problem (like MCDF), it is often possible to choose a pair of partitions so that the non-zero elements of  $A$  are connected to relatively few of the blocks in the block structure (and normally these blocks will themselves be sparse). The two block structures that are amenable to decomposition methods and we discussed in previous chapters are of the block angular and staircase forms.

### 2.3.1 Block-Angular Structured Systems

The matrix has a set of rows  $I_0$  that connect with all sets of columns, and sets of rows  $I_t$  that each connect with a single set of columns  $J_t$ . We had such a constraint in previous sections and we called it master constraint. In some practical applications, there is usually an additional set of columns  $J_0$  that interacts solely with the row set  $I_0$ . The block angular structure is one of the most widely known and recognized structures in decomposition, as it is the basis for the original decomposition method developed by Dantzig and Wolfe. The structure may typically arise where the system being modeled splits naturally into a set of subsystems, e.g., a set of facilities/time periods, independent apart from a number of global constraints. The variables and constraints referring to a single subsystem  $t$  correspond to the rows and columns in sets  $I_t$  and  $J_t$ . The constraints that link the subsystems, corresponding to rows  $I_0$ , may express limits of system-wide scarce resources or ensure that materials balance correctly between the subsystems (e.g., facilities or time periods), and are referred to as global, *common or linking constraints*.

### 2.3.2 Staircase-Structured Systems

The block staircase structure is best explained as a dynamic (time stage) structure. Let column set  $J_t$  comprise the columns of variables related directly to time period  $t$ . Row set  $I_t$  comprises the rows for constraints linking the decisions made in period  $t$  with those made in the previous period  $t-1$ . It's usually said that staircase structure has a time lag of one, as activities in period  $t$  are related directly to those in period  $t-1$ , but not to those from earlier periods. However, we mentioned that the production/storage policy of the current period may only be affected by the previous step, and so this attribute (time lag of one) well describe the storage policy needed for MPDNFs. We will use it later when we have to include storage into the problem's parameters. In general,

the staircase structure may have longer time lags. These types of structure can be exploited by nested DW decomposition.

It should be noted that, although DW algorithm works in a different feasible region from that of the original problem there is a correspondence between the master problem with feasible region  $P_{DW}$ , and the original problem with feasible region  $P$ . Thus, under the *Minkowski Mapping* (Caratheodory Theorem), the master problem and original problem are equivalent, in the sense that a solution to one implies an equivalent solution to the other, where solution is taken in its broadest sense to encompass a primal or dual point or ray [9][22]. A basic DW algorithm [21][22][44][45][83] can be formulated as following and we show how to implement it by GAMS through a multiperiod problem.

```

{Initialization}
Choose initial subsets of variables
While true do
  {Master problem}
  Solve the restricted master problem
   $\mu_1 :=$  duals of master constraints
   $\mu_2^{(t)} :=$  duals of the  $t^{\text{th}}$  convexity constraints
  {Subproblems}
  for  $t = 0, \dots, T$  do
    Plug  $\mu_1$  and  $\mu_2^{(t)}$  into subproblem  $t$ 
    Solve subproblem  $t$ 
    if  $(\text{reduced cost})_t = \min (c^t - \mu_1 M)Y_t - \mu_2^{(t)} < 0$  then
      Add proposal  $Y_t$  to the restricted master problem
    end if
  end for
  if no proposals generated then
    Stop: optimality
  End if
end while

```

## 2.4 A Model-Based Approach for Multiperiod Networks with Storage

An MPDNF is determined by a directed graph  $G=(V, A, T)$ , where  $V$  is a set of vertices and  $A$  is a set of arcs. It consists of two non-negative capacity functions  $u: \{A \cup V\} \times N \rightarrow \mathbb{R}^+$  and  $u^T: A \rightarrow \mathbb{R}^+$ , and a pre-defined non-negative time-varying cost function  $c: \{A \cup V\} \times N \rightarrow \mathbb{R}^+$ , where  $N = \{0, 1, \dots, T-1\}$  is the set of discrete periods. In the meanwhile, flow is allowed to be stored in nodes at any period, and we can use the

stored flow for the next time. The discrete-time model is considered, in which all times are integral and bounded by an integer horizon. A *discrete dynamic flow* in  $G$ , which satisfies the following constraints, is said to be *feasible*. Such a flow is a non-negative function  $x: \{A \cup V\} \times \mathbb{N} \rightarrow \mathbb{R}^+$  satisfying (2.12)-(2.16).

$$\sum_j \sum_{t=0}^{T-1} x_{ij}(t) - \sum_j \sum_{t=0}^{T-1} x_{ji}(t) = v_i \quad \forall i \in V, \quad (2.12)$$

$$\sum_j x_{ij}(t) - \sum_j x_{ji}(t) \leq 0 \quad \forall i \in V \setminus VS, \quad \forall t \in \mathbb{N}, \quad (2.13)$$

$$\sum_{t \in \mathbb{N}} x_{ij}(t) \leq u_{ij}^T \quad \forall (i, j) \in A, \quad (2.14)$$

$$0 \leq x_{ij}(t) \leq u_{ij}(t) \quad \forall (i, j) \in A, \quad \forall t \in \mathbb{N}, \quad (2.15)$$

$$x_{ij}(t) = 0 \quad \forall (i, j) \in A, \quad \forall t \notin \mathbb{N}. \quad (2.16)$$

In this formulation,  $x_{ij}(t)$  is the amount of flow passing through arc  $(i, j)$  at time period  $t$ . We refer to constraints (3.14) as *horizon capacity constraints*. Flow must not remain in the network after time  $T$ , and this is ensured by conditions (2.12) and (2.16).

Let  $c_{ii}(t)$  be the storage cost in node  $i$  at period  $t$ . Thus, the total cost of a discrete dynamic flow  $x$  is defined by

$$\sum_{t \in \mathbb{N}} \sum_{(i,j) \in A} c_{ij}(t) x_{ij}(t) + \sum_{i \in V} \sum_{t \in \mathbb{N}} c_{ii}(t) \left( \sum_j x_{ji}(t) - \sum_j x_{ij}(t) \right). \quad (2.17)$$

Having introduced the unrestricted variables  $v_i(t)$ , we may reformulate the problem as

$$\mathbf{Min}_{x_{ij}(t), x_{ii}(t), v_i(t)} \sum_{t \in \mathbb{N}} \sum_{(i,j) \in A} c_{ij}(t) x_{ij}(t) + \sum_{t \in \mathbb{N}} \sum_{i \in V} c_{ii}(t) x_{ii}(t) + \sum_{t \in \mathbb{N}} \sum_{i \in V} c_i(t) v_i(t), \quad (2.18)$$

$$\sum_j x_{ij}(t) - \sum_j x_{ji}(t) + [x_{ii}(t) - x_{ii}(t-1)] - v_i(t) = 0 \quad \forall t \in \mathbb{N}, \quad \forall i \in V, \quad (2.19)$$

$$\sum_{t \in \mathbb{N}} v_i(t) = v_i \quad \forall i \in V, \quad (2.20)$$

$$0 \leq x_{ii}(t) \leq u_{ii}(t) \quad \forall t \in \mathbb{N}, \quad \forall i \in V \setminus VS, \quad (2.21)$$

$$\sum_{t \in \mathbf{N}} x_{ij}(t) \leq u_{ij}^T \quad \forall (i, j) \in \mathbf{A}, \quad (2.22)$$

$$0 \leq x_{ij}(t) \leq u_{ij}(t) \quad \forall (i, j) \in \mathbf{A}, \quad \forall t \in \mathbf{N}, \quad (2.23)$$

$$x_{ij}(t) = 0 \quad \forall (i, j) \in \mathbf{A}, \quad \forall t \notin \mathbf{N}, \quad (2.24)$$

$$x_{ii}(t) = 0 \quad \forall i \in \mathbf{V}, \quad t = -1, T-1, \quad (2.25)$$

where  $v_i(t)$  is an unrestricted variable which defines the difference between outflow and inflow at node  $i$  at time step  $t$ . Needless to say,  $v_i(t)$  differs from storage decision variables, and we leave them to the algorithm to optimally determine.  $x_{ii}(t)$  and  $c_{ii}(t)$  are the amount and cost of stored flow at node  $i$  in period  $t$ . We set  $c_i(t) = 0$  for each  $i$  and  $t$ . Clearly, there is no need to have flow storage in period  $T-1$ , and this is ensured by (2.25). One can easily check that conditions (2.19)-(2.21) are equivalent to condition (2.12) and (2.13) subject to considering (2.25). We prove that this model possesses a nice property which enables us to reduce the MCDF problem to a problem with special structure. By substituting  $v_{i+}^t - v_{i-}^t$  for  $v_i(t)$ , our problem is reduced to the following *matrix form* as an LP, whose special structure helps us exploit very efficient computation algorithm of its solution.

$$\mathbf{Min}_{[\mathbf{X}^t], [\mathbf{X}_s^t], [\mathbf{V}^t]} \sum_{t \in \mathbf{N}} [\mathbf{C}^t]^{trans} \cdot [\mathbf{X}^t] + \sum_{t \in \mathbf{N}} [\mathbf{C}_s^t]^{trans} \cdot [\mathbf{X}_s^t],$$

$$[\mathbf{X}^0]_{m \times 1} + [\mathbf{X}^1]_{m \times 1} + \dots + [\mathbf{X}^{T-1}]_{m \times 1} + [\mathbf{S}]_{m \times 1} = [\mathbf{U}^T]_{m \times 1},$$

$$([\mathbf{V}_+^0] - [\mathbf{V}_-^0]) + ([\mathbf{V}_+^1] - [\mathbf{V}_-^1]) + \dots + ([\mathbf{V}_+^{T-1}] - [\mathbf{V}_-^{T-1}]) + [\mathbf{0}]_{n \times 1} = [\mathbf{V}]_{n \times 1},$$

$$[\mathbf{A}]_{n \times m} [\mathbf{X}^0]_{m \times 1} + [\mathbf{I}]_{n \times n} [\mathbf{X}_s^0]_{n \times 1} + [-\mathbf{I}]_{n \times n} [\mathbf{X}_s^{-1}]_{n \times 1} - ([\mathbf{V}_+^0] - [\mathbf{V}_-^0]) = [\mathbf{0}]_{n \times 1},$$

$$[\mathbf{A}]_{n \times m} [\mathbf{X}^1]_{m \times 1} + [\mathbf{I}]_{n \times n} [\mathbf{X}_s^1]_{n \times 1} + [-\mathbf{I}]_{n \times n} [\mathbf{X}_s^0]_{n \times 1} - ([\mathbf{V}_+^1] - [\mathbf{V}_-^1]) = [\mathbf{0}]_{n \times 1},$$

$\vdots$

$$[\mathbf{A}]_{n \times m} [\mathbf{X}^{T-1}]_{m \times 1} + [\mathbf{I}]_{n \times n} [\mathbf{X}_s^{T-1}]_{n \times 1} + [-\mathbf{I}]_{n \times n} [\mathbf{X}_s^{T-2}]_{n \times 1} - ([\mathbf{V}_+^{T-1}] - [\mathbf{V}_-^{T-1}]) = [\mathbf{0}]_{n \times 1},$$

$$[\mathbf{0}]_{m \times 1} \leq [\mathbf{X}^t]_{m \times 1} \leq [\mathbf{U}^t]_{m \times 1} \quad \forall t \in \mathbf{N},$$

$$[\mathbf{0}]_{n \times 1} \leq [\mathbf{X}_s^t]_{n \times 1} \leq [\mathbf{U}_s^t]_{n \times 1} \quad \forall t \in \mathbf{N},$$

$$[\mathbf{V}_+^t]_{n \times 1} \geq [\mathbf{0}]_{n \times 1}, \quad [\mathbf{V}_-^t]_{n \times 1} \geq [\mathbf{0}]_{n \times 1} \quad \forall t \in \mathbf{N},$$

where  $[\mathbf{X}^t] = \{x_{ij}(t)\}_{i,j} = \{x_{ij}^t\}_{i,j}$  and  $[\mathbf{X}_s^t] = \{x_{ii}(t)\}_i = \{x_{ii}^t\}_i$  are the vectors of flow and storage at time period  $t$ ,  $[\mathbf{U}^t] = \{u_{ij}(t)\}_{i,j} = \{u_{ij}^t\}_{i,j}$  and  $[\mathbf{U}_s^t] = \{u_{ii}(t)\}_i = \{u_{ii}^t\}_i$  are the vectors of flow capacities and storage at  $t \in \mathbb{N}$ , respectively,  $[\mathbf{U}^T] = \{\mathbf{u}_{ij}^T\}_{i,j}$  the vector of pre-defined horizon capacities.  $[\mathbf{C}^t] = \{c_{ij}(t)\}_{i,j} = \{c_{ij}^t\}_{i,j}$  and  $[\mathbf{C}_s^t] = \{c_{ii}(t)\}_i = \{c_{ii}^t\}_i$  are the vectors of pre-defined flow costs on arcs and storage costs in nodes, respectively. Let  $[\mathbf{V}^t] = \{v_i(t)\}_i = \{v_i^t\}_i$  represent the vector of decision variables showing the differences between outflow and inflow of nodes at time  $t$ .  $[\mathbf{A}]_{n \times m}$  is the node-arc incidence matrix of the network.  $[\mathbf{V}^t]$  is decomposed to  $[\mathbf{V}_+^t]$  and  $[\mathbf{V}_-^t]$ .  $[\mathbf{V}] = \{v_i\}_i$  is the vector of pre-defined supply/demand numbers. Let  $[\mathbf{X}_s^{-1}]$  be the vector of initial storage which is zero, and  $[\mathbf{S}]$  be the vector of slack variables. Without any loss of generality, let  $\mathbf{S} := \mathbf{X}^T$  and  $[\mathbf{V}_+^T] = [\mathbf{V}_-^T] = [\mathbf{0}]$ . Matrix properties help us manipulate the problem to extract the following efficient matrix form.

$$\begin{aligned}
& \underset{\{\mathbf{X}^t, \mathbf{X}_s^t, \mathbf{V}_+^t, \mathbf{V}_-^t\}_t}{\text{Min}} \sum_{t \in \mathbb{N}} [\mathbf{C}^t \quad \mathbf{C}_s^t \quad \mathbf{0} \quad \mathbf{0}] [\mathbf{X}^t \quad \mathbf{X}_s^t \quad \mathbf{V}_+^t \quad \mathbf{V}_-^t]^{\text{trans}}, \\
& \begin{pmatrix} \mathbf{I}_m & \mathbf{0}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_m & \mathbf{0}_n & \mathbf{I}_n & -\mathbf{I}_n \end{pmatrix} \begin{pmatrix} \mathbf{X}^0 \\ \mathbf{X}_s^0 \\ \mathbf{V}_+^0 \\ \mathbf{V}_-^0 \end{pmatrix} + \begin{pmatrix} \mathbf{I}_m & \mathbf{0}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_m & \mathbf{0}_n & \mathbf{I}_n & -\mathbf{I}_n \end{pmatrix} \begin{pmatrix} \mathbf{X}^1 \\ \mathbf{X}_s^1 \\ \mathbf{V}_+^1 \\ \mathbf{V}_-^1 \end{pmatrix} + \dots + \begin{pmatrix} \mathbf{I}_m & \mathbf{0}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_m & \mathbf{0}_n & \mathbf{I}_n & -\mathbf{I}_n \end{pmatrix} \begin{pmatrix} \mathbf{X}^{T-1} \\ \mathbf{X}_s^{T-1} \\ \mathbf{V}_+^{T-1} \\ \mathbf{V}_-^{T-1} \end{pmatrix} + \dots \\
& \dots + \begin{pmatrix} \mathbf{I}_m & \mathbf{0}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_m & \mathbf{0}_n & \mathbf{I}_n & -\mathbf{I}_n \end{pmatrix} \begin{pmatrix} \mathbf{X}^T \\ \mathbf{X}_s^T \\ \mathbf{V}_+^T \\ \mathbf{V}_-^T \end{pmatrix} = \begin{pmatrix} \mathbf{U}^T \\ \mathbf{V} \end{pmatrix}, \\
& [\mathbf{A}_{n \times m} \quad \mathbf{I}_n \quad -\mathbf{I}_n \quad \mathbf{I}_n] \begin{pmatrix} \mathbf{X}^0 \\ \mathbf{X}_s^0 \\ \mathbf{V}_+^0 \\ \mathbf{V}_-^0 \end{pmatrix} = [\mathbf{0}]_{n \times 1}, \\
& [\mathbf{0}_m \quad -\mathbf{I}_n \quad \mathbf{0}_n \quad \mathbf{0}_n] \begin{pmatrix} \mathbf{X}^0 \\ \mathbf{X}_s^0 \\ \mathbf{V}_+^0 \\ \mathbf{V}_-^0 \end{pmatrix} + [\mathbf{A}_{n \times m} \quad \mathbf{I}_n \quad -\mathbf{I}_n \quad \mathbf{I}_n] \begin{pmatrix} \mathbf{X}^1 \\ \mathbf{X}_s^1 \\ \mathbf{V}_+^1 \\ \mathbf{V}_-^1 \end{pmatrix} = [\mathbf{0}]_{n \times 1},
\end{aligned}$$

$$\begin{aligned}
& \begin{bmatrix} \mathbf{0}_m & -\mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \end{bmatrix} \begin{pmatrix} \mathbf{X}^1 \\ \mathbf{X}_s^1 \\ \mathbf{V}_+^1 \\ \mathbf{V}_-^1 \end{pmatrix} + \begin{bmatrix} \mathbf{A}_{n \times m} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{I}_n \end{bmatrix} \begin{pmatrix} \mathbf{X}^2 \\ \mathbf{X}_s^2 \\ \mathbf{V}_+^2 \\ \mathbf{V}_-^2 \end{pmatrix} = [\mathbf{0}]_{n \times 1}, \\
& \vdots \\
& \begin{bmatrix} \mathbf{0}_m & -\mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \end{bmatrix} \begin{pmatrix} \mathbf{X}^{T-2} \\ \mathbf{X}_s^{T-2} \\ \mathbf{V}_+^{T-2} \\ \mathbf{V}_-^{T-2} \end{pmatrix} + \begin{bmatrix} \mathbf{A}_{n \times m} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{I}_n \end{bmatrix} \begin{pmatrix} \mathbf{X}^{T-1} \\ \mathbf{X}_s^{T-1} \\ \mathbf{V}_+^{T-1} \\ \mathbf{V}_-^{T-1} \end{pmatrix} = [\mathbf{0}]_{n \times 1}, \\
& \begin{bmatrix} \mathbf{0}_m & -\mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \end{bmatrix} \begin{pmatrix} \mathbf{X}^{T-1} \\ \mathbf{X}_s^{T-1} \\ \mathbf{V}_+^{T-1} \\ \mathbf{V}_-^{T-1} \end{pmatrix} + \begin{bmatrix} \mathbf{A}_{n \times m} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{I}_n \end{bmatrix} \begin{pmatrix} \mathbf{X}^T \\ \mathbf{X}_s^T \\ \mathbf{V}_+^T \\ \mathbf{V}_-^T \end{pmatrix} = [\mathbf{0}]_{n \times 1}, \\
& [\mathbf{0}] \leq \begin{pmatrix} \mathbf{X}^t \\ \mathbf{X}_s^t \\ \mathbf{V}_+^t \\ \mathbf{V}_-^t \end{pmatrix} \leq \begin{pmatrix} \mathbf{U}^t \\ \mathbf{U}_s^t \\ \mathbf{V} \\ \mathbf{V} \end{pmatrix} \quad \forall t \in \mathbb{N}.
\end{aligned}$$

The system arisen above emphasizes that the production/usage/storage at one period might be affected by production/storage of the previous time step, and affects the amount of usage/storage of the following period. The matrix form of the problem shows that we can formulate any MCDF problem on a multiperiod dynamic network (with storage) as a problem which possesses the staircase structured system. This structure permits efficient handling. One general approach might be the technique of decomposition. Here, we may refer to those constraints with general structure as master constraint and ones with special structures as subproblems while applying decomposition methods, DW decomposition or Benders decomposition. In other words, the original problem is reformulated into a series of structurally similar LP subproblems.

The structural similarity of the subproblems and the sparsity of constraints matrices allow us to use decomposition techniques to improve the computational efficiency. In addition to structural similarity, this modelling technique has converted MCDF problem to one with many sparse matrices. Generally, decomposition algorithms have inherent efficiency for large-scale problems. On the other hand, while the original description of

the algorithm is motivated by its reduced memory usage, modern computers can also take advantage of the algorithm's inherent parallelism. In Chapter 3, we will discuss the parallelism of Dantzig-Wolfe algorithm and we will report very promising results from Rios [76] for such parallel processing. However, prior to taking advantage of algorithm's parallelism we should be able to decompose the problem into suitable blocks to be attacked by DW. We also might note that the foregoing formulation of the MCDF problem has a simple constraint structure. It has got  $m+n(1+T)$  constraints and  $m+(m+2n)T+n(T-1)$  variables (including the slacks), where  $m$  is the cardinality of arc set, and  $n$  the network size.

## 2.5 Slice Modelling

One of the strategies to handle optimization problems formed by subproblems with similar structures is *slice modelling*. The modelling language used in this study may resemble slice modelling system. Although they may seem similar, there are some important differences between slice modelling system and polyhedral-based approach of this thesis. The resemblance and differences of the problems can directly arise from the definition/assumptions of slice modelling. In a broad sense, a slice model consists of a series of mathematical programs with the same structure but different data. Further, in slice models, often the data elements are related: some or most of the data may stay the same between programs, and so the programs differ only in a few rows or columns. Because of this, the basic structure in a slice model remains the same from program to program. To get a better understanding, one may consult with Ferris et al. [25][26]. For the  $k$ -th slice program, this idea can be expressed as follows:

$$\begin{aligned} \mathbf{Min}_x \quad & f_k(x) \text{ (Objective slice)} \\ \mathbf{A}_k(x) = \mathbf{b}_k \quad & \text{(Slice constraint)} \\ x \in X \quad & \text{(Core constraint)} \end{aligned}$$

Where  $\mathbf{A}_k$  represents the matrix of constraint coefficients which (along with right-hand-side  $\mathbf{b}_k$ ) are unique to the  $k$ -th program. The set  $X$  represents the (core) constraints and program structure that remain constant between programs [85]. One of the fundamental differences between the subproblems generated in the paper and slice problems is that in



slice problems none of the variables of one subproblem is involved on another (independency). But in our case, as discussed, the arisen system shows that the production/storage at any period will affect the production/storage of the next period, which means some of the decision variables of the current period will appear in the next period's decision variables set, and our decision at each step cannot be independent of that from the previous period. On the other hand, in contrast to slice modelling, our model considers a non-separable common objective function-for all subproblems-where all variables included in all subproblems are appeared. Furthermore, slice modelling system can be interpreted as a series of mathematical programs which must be solved in order to obtain the complete solution of the optimization problem, but this is not the case in our model which thinks of the MCDF problem as a whole comprising some polyhedrals.

## 2.6 Generalized Multiperiod Network Flows (MPDNF with Spoilage)

In each of the models we have considered up to now, we have made a fundamental assumption, namely, flow has to be conserved on every arc. This assumption seems reasonable in many applications, including those we expressed in the previous sections. However, many other practical applications may violate this conservation assumption. For example, in the transmission of a volatile gas, we may lose flow because of evaporation; or, in the transmission of liquids such as raw petroleum crude, we might lose flow due to leakage [3].

The *generalized multiperiod dynamic network flow* problem (GMPDNF) is a natural generalization of the problem stated in the foregoing chapter. GMPDNF problem is going to develop multiperiod network problem by allowing flow to leak as it is sent through the network. In this setting, each arc  $(i, j)$  has a time varying non-negative capacity  $u_{ij}(t)$  and horizon capacity  $u_{ij}^T$ . Additionally, each arc will be assigned a time-varying non-negative *gain/loss multiplier*  $\lambda_{ij}(t)$  associated with it. We will refer to  $\lambda_{ij}(t)$  as the *factor* of arc  $(i, j)$  at time  $t$ . When we send  $x_{ij}(t)$  units of flow from node  $i$  via arc  $(i, j)$  at time  $t$ ,  $\lambda_{ij}(t).x_{ij}(t)$  units of flow arrive at node  $j$  at the same time. If  $\lambda_{ij}(t) < 1$ , the arc is *lossy*; if  $\lambda_{ij}(t) \geq 1$  the arc is *gainy* (on that time). Therefore, for example, if there is

flow spoilage (loss) on an arc during the all periods, then model may be represented as the initial one by assigning a *loss factor* to the related arc.

$$\sum_j \int_0^T x_{ij}(t) dt - \sum_j \int_0^T \lambda_{ji}(t) x_{ji}(t) dt = v_i \quad \forall i \in V. \quad (2.26)$$

Generalized multiperiod networks can successfully model many application settings that cannot appropriately be represented as ordinary min-cost flow problems. The factors can represent physical transformations of one commodity into a less or greater amount of the same commodity. Some examples may include: spoilage, theft, evaporation, taxes, seepage, deterioration, interest, or breeding. The gain/loss factors may also model the transformation of one commodity into a different commodity. Some examples for this case could include: converting raw materials into finished goods, currency conversion, and machine scheduling [3][[62][70].

A generalized multiperiod network  $G=(V, A, T)$  consists of node set  $V$ , arc set  $A$ , two non-negative capacity functions  $u: A \times N \rightarrow \mathbb{R}^+$  and  $u^T: A \rightarrow \mathbb{R}^+$ , pre-defined non-negative cost function  $c: A \times N \rightarrow \mathbb{R}^+$ , and pre-defined non-negative time-varying gain/loss function  $\lambda: A \times N \rightarrow \mathbb{R}^+$ , where  $N = \{0, 1, \dots, T-1\}$  is the set of discrete periods. A *discrete feasible dynamic flow* in  $G$  will be a non-negative function  $x = \{x_{ij}^t\}_{i,j}^t: A \times N \rightarrow \mathbb{R}^+$  satisfying (2.27)-(2.30).

$$\sum_j \sum_{t=0}^{T-1} x_{ij}^t - \sum_j \sum_{t=0}^{T-1} \lambda_{ji}^t x_{ji}^t = v_i \quad \forall i \in V, \quad (2.27)$$

$$\sum_{t \in N} x_{ij}^t \leq u_{ij}^T \quad \forall (i, j) \in A, \quad (2.28)$$

$$0 \leq x_{ij}^t \leq u_{ij}^t \quad \forall (i, j) \in A, \quad \forall t \in N, \quad (2.29)$$

$$x_{ij}^t = 0 \quad \forall (i, j) \in A, \quad \forall t \notin N. \quad (2.30)$$

The equations stated above are the conditions for flow feasibility in a generalized MPDNF. Note that we are assuming that the arc capacity  $u_{ij}^t$  is an upper bound on the flow sent from node  $i$  at time step  $t$ , not on the flow that becomes available at node  $j$ . Similarly,  $c_{ij}^t$  should be interpreted as the cost for each unit of flow that we send from

node  $i$ , not the per unit cost of the flow that reaches node  $j$ . Now, we can formulate the minimum cost dynamic flow problem on GMPDNF

$$(Discrete-Time\ GMPDNF) \quad \mathbf{Min}_{x_{ij}^t} \quad \sum_{t \in \mathbf{N}} \sum_{(i,j) \in \mathbf{A}} c_{ij}^t x_{ij}^t, \\ \text{Subject to (2.27)-(2.39).}$$

For this case, we will also show that the problem possesses the property which enables us to reduce the MCDF problem to a problem with special structure without being have to use time-expanded network. By introducing the unrestricted variable  $v_i^t$ , we reformulate the problem as:

$$\mathbf{Min}_{x_{ij}^t, v_i^t} \quad \sum_{t \in \mathbf{N}} \sum_{(i,j) \in \mathbf{A}} c_{ij}^t x_{ij}^t + \sum_{t \in \mathbf{N}} \sum_{i \in \mathbf{V}} c_i^t v_i^t, \quad (2.31)$$

$$\left( \sum_j x_{ij}^t - \sum_j \lambda_{ji}^t x_{ji}^t \right) - v_i^t = 0 \quad \forall i \in \mathbf{V}, \forall t \in \mathbf{N}, \quad (2.32)$$

$$\sum_{t \in \mathbf{N}} v_i^t = v_i \quad \forall i \in \mathbf{V}, \quad (2.33)$$

$$\sum_{t \in \mathbf{N}} x_{ij}^t \leq u_{ij}^T \quad \forall (i,j) \in \mathbf{A}, \quad (2.34)$$

$$0 \leq x_{ij}^t \leq u_{ij}^t \quad \forall (i,j) \in \mathbf{A}, \forall t \in \mathbf{N}. \quad (2.35)$$

where  $v_i^t$  is a free decision variable which defines the difference between outflow and inflow at node  $i$  at time step  $t$ , and  $c_i^t = 0$  for each  $i$  and  $t$ . It is easy to check that conditions (2.32) and (2.33) together are equivalent to condition (2.27). To develop our polyhedral-based approach we need to define a *two-level matrix transformation* as following. To this end, we introduce secondary matrix  $[\Phi]_{n \times m}$  as:

$$[\Phi]_{n \times m} := [\mathbf{A}]_{n \times m} \times [\Sigma]_{m \times m},$$

where  $[\Sigma]$  is a  $m \times m$ -diagonal matrix whose elements are the arc factors of our MP network in the same order that arcs appear in  $m$ -vector  $[\mathbf{X}]_{m \times 1}$  ( $m$  is the number of arcs of the network). Now, for a fixed time period  $t$  we construct  $[\mathbf{B}^t]_{n \times m} = [\mathbf{B}]_{n \times m}$  as following.

$$[\mathbf{B}]_{ij} := \begin{cases} [\Phi]_{ij} & \text{if } [\Phi]_{ij} \leq 0, \\ 1 & \text{if } [\Phi]_{ij} > 0, \end{cases} \quad (2.36)$$

where  $[\mathbf{B}]_{ij}$  and  $[\Phi]_{ij}$  represent the  $(i, j)$ -th elements of the matrices  $[\mathbf{B}]$  and  $[\Phi]$ , respectively. We refer to matrix  $[\mathbf{B}]_{n \times m}$  as the *generalized node-arc incidence matrix* of the multiperiod network. Hence, we call  $[\mathbf{B}^t]_{n \times m}$  as the generalized node-arc incidence matrix of GMPDNF at time step  $t$ . Note that, due to changes in arc factors over time, incident matrices are not necessarily the same for each time step, but since the time-varying gain/loss functions are pre-defined, all matrices can be computed off-line when we have a jump in time. As an illustration of this idea, let's consider the 3-dimensional incidence matrix of the network presented in Figure 2.2 for a fixed period, and see how this idea works.

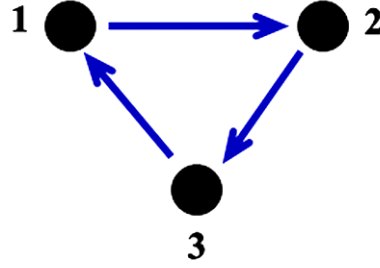


Figure 2.2 A typical GMPDNF at a fixed time period

Suppose  $[\mathbf{V}] = \{v_i\}_{i=1,2,3}$  is the vector of supply/demand numbers. We get the vector of flows  $[\mathbf{X}] = \{x_{ij}\}_{i=1,2,3}^{j=1,2,3}$ , node-arc incidence matrix  $[\mathbf{A}]$ , and diagonal matrix  $[\mathbf{\Sigma}]$  as following.

$$\begin{matrix} \mathbf{A} \\ \begin{bmatrix} 1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix} \end{matrix} \quad \text{and} \quad \begin{matrix} \mathbf{X} \\ \begin{bmatrix} x_{12} \\ x_{31} \\ x_{23} \end{bmatrix} \end{matrix} \quad \text{and} \quad \begin{matrix} \mathbf{V} \\ \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \end{matrix} \quad \text{and} \quad \begin{matrix} \mathbf{\Sigma} \\ \begin{bmatrix} \lambda_{12} & 0 & 0 \\ 0 & \lambda_{31} & 0 \\ 0 & 0 & \lambda_{23} \end{bmatrix} \end{matrix}.$$

Now, let's introduce ancillary matrix  $[\Phi]$  as discussed. It is

$$[\Phi] = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} \lambda_{12} & 0 & 0 \\ 0 & \lambda_{31} & 0 \\ 0 & 0 & \lambda_{23} \end{bmatrix} = \begin{bmatrix} \lambda_{12} & -\lambda_{31} & 0 \\ -\lambda_{12} & 0 & \lambda_{23} \\ 0 & \lambda_{31} & -\lambda_{23} \end{bmatrix}.$$

Having  $[\Phi]$  at hand, we can construct the generalized node-arc incidence matrix  $[\mathbf{B}]$ .

$$[\mathbf{B}] := \begin{bmatrix} 1 & -\lambda_{31} & 0 \\ -\lambda_{12} & 0 & 1 \\ 0 & 1 & -\lambda_{23} \end{bmatrix}_{3 \times 3}.$$

It is easy to see that  $[\mathbf{B}][\mathbf{X}]$  yields exactly required conditions for a flow in GMPDNF problem.

$$[\mathbf{B}][\mathbf{X}] = \begin{bmatrix} 1 & -\lambda_{31} & 0 \\ -\lambda_{12} & 0 & 1 \\ 0 & 1 & -\lambda_{23} \end{bmatrix} \begin{bmatrix} x_{12} \\ x_{31} \\ x_{23} \end{bmatrix} = \begin{bmatrix} x_{12} - \lambda_{31}x_{31} \\ x_{23} - \lambda_{12}x_{12} \\ x_{31} - \lambda_{23}x_{23} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = [\mathbf{V}].$$

By substituting  $v_{i+}^t - v_{i-}^t$  for  $v_i^t$  ( $v_{i+}^t \geq 0, v_{i-}^t \geq 0$ ) and considering the two-level matrix transformation, the model can be stated as:

$$\text{Min}_{\mathbf{X}^t} \sum_{t \in \mathbf{N}} [\mathbf{C}^t]^{trans} \cdot [\mathbf{X}^t],$$

$$[\mathbf{X}^0]_{m \times 1} + [\mathbf{X}^1]_{m \times 1} + \dots + [\mathbf{X}^{T-1}]_{m \times 1} \leq [\mathbf{U}^T]_{m \times 1},$$

$$([\mathbf{V}_+^0] - [\mathbf{V}_-^0]) + ([\mathbf{V}_+^1] - [\mathbf{V}_-^1]) + \dots + ([\mathbf{V}_+^{T-1}] - [\mathbf{V}_-^{T-1}]) = [\mathbf{V}]_{n \times 1},$$

$$[\mathbf{B}^0]_{n \times m} [\mathbf{X}^0]_{m \times 1} - [\mathbf{I}]_{n \times n} [\mathbf{V}_+^0]_{n \times 1} + [\mathbf{I}]_{n \times n} [\mathbf{V}_-^0]_{n \times 1} = [\mathbf{0}]_{n \times 1},$$

$$[\mathbf{B}^1]_{n \times m} [\mathbf{X}^1]_{m \times 1} - [\mathbf{I}]_{n \times n} [\mathbf{V}_+^1]_{n \times 1} + [\mathbf{I}]_{n \times n} [\mathbf{V}_-^1]_{n \times 1} = [\mathbf{0}]_{n \times 1},$$

$\vdots$

$$[\mathbf{B}^{T-1}]_{n \times m} [\mathbf{X}^{T-1}]_{m \times 1} - [\mathbf{I}]_{n \times n} [\mathbf{V}_+^{T-1}]_{n \times 1} + [\mathbf{I}]_{n \times n} [\mathbf{V}_-^{T-1}]_{n \times 1} = [\mathbf{0}]_{n \times 1},$$

$$[\mathbf{0}]_{m \times 1} \leq [\mathbf{X}^t]_{m \times 1} \leq [\mathbf{U}^t]_{m \times 1} \quad \forall t \in \mathbf{N}.$$

$$[\mathbf{V}_+^t]_{n \times 1} \geq [\mathbf{0}]_{n \times 1}, \quad [\mathbf{V}_-^t]_{n \times 1} \geq [\mathbf{0}]_{n \times 1} \quad \forall t \in \mathbf{N},$$

where  $[\mathbf{X}^t] = \{x_{ij}^t\}_{i,j}$  and  $[\mathbf{U}^t] = \{u_{ij}^t\}_{i,j}$  are the vectors of flow and capacities in  $t$ .  $[\mathbf{U}^T] = \{u_{ij}^T\}_{i,j}$  is the vector of horizon capacities. Let  $[\mathbf{C}^t] = \{c_{ij}^t\}_{i,j}$  represent the vector of arc costs at  $t \in \mathbf{N}$  and  $[\mathbf{B}]$  be the generalized node-arc incidence matrix.  $[\mathbf{V}^t] = \{v_i^t\}_i$  and  $[\mathbf{V}] = \{v_i\}_i$  are defined as before. Matrix properties allow us to express the following

matrix form as a linear program whose special structure has a great advantage to exploit efficient algorithms for its solution. let  $[\mathbf{S}] := [\mathbf{X}^T]$  and  $[\mathbf{V}_+^T] = [\mathbf{V}_-^T] = [\mathbf{0}]$ .

$$\begin{aligned}
& \underset{\mathbf{X}^t, \mathbf{V}_+^t, \mathbf{V}_-^t}{\text{Min}} \sum_{t \in \mathbf{N}} [\mathbf{C}^t \quad \mathbf{0} \quad \mathbf{0}] [\mathbf{X}^t \quad \mathbf{V}_+^t \quad \mathbf{V}_-^t]^{\text{trans}}, \\
& \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_m & \mathbf{I}_n & -\mathbf{I}_n \end{bmatrix} \begin{pmatrix} \mathbf{X}^0 \\ \mathbf{V}_+^0 \\ \mathbf{V}_-^0 \end{pmatrix} + \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_m & \mathbf{I}_n & -\mathbf{I}_n \end{bmatrix} \begin{pmatrix} \mathbf{X}^1 \\ \mathbf{V}_+^1 \\ \mathbf{V}_-^1 \end{pmatrix} + \dots + \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_m & \mathbf{I}_n & -\mathbf{I}_n \end{bmatrix} \begin{pmatrix} \mathbf{X}^{T-1} \\ \mathbf{V}_+^{T-1} \\ \mathbf{V}_-^{T-1} \end{pmatrix} + \dots + \\
& \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_m & \mathbf{I}_n & -\mathbf{I}_n \end{bmatrix} \begin{pmatrix} \mathbf{X}^T \\ \mathbf{V}_+^T \\ \mathbf{V}_-^T \end{pmatrix} = \begin{pmatrix} \mathbf{U}(T) \\ \mathbf{V} \end{pmatrix}, \\
& \begin{bmatrix} \mathbf{B}_{n \times m}^0 & -\mathbf{I}_n & \mathbf{I}_n \end{bmatrix} \begin{pmatrix} \mathbf{X}^0 \\ \mathbf{V}_+^0 \\ \mathbf{V}_-^0 \end{pmatrix} = [\mathbf{0}], \\
& \begin{bmatrix} \mathbf{B}_{n \times m}^1 & -\mathbf{I}_n & \mathbf{I}_n \end{bmatrix} \begin{pmatrix} \mathbf{X}^1 \\ \mathbf{V}_+^1 \\ \mathbf{V}_-^1 \end{pmatrix} = [\mathbf{0}], \\
& \vdots \\
& \begin{bmatrix} \mathbf{B}_{n \times m}^T & -\mathbf{I}_n & \mathbf{I}_n \end{bmatrix} \begin{pmatrix} \mathbf{X}^T \\ \mathbf{V}_+^T \\ \mathbf{V}_-^T \end{pmatrix} = [\mathbf{0}], \\
& [\mathbf{0}] \leq \begin{pmatrix} \mathbf{X}^t \\ \mathbf{V}_+^t \\ \mathbf{V}_-^t \end{pmatrix} \leq \begin{pmatrix} \mathbf{U}^t \\ \mathbf{V} \\ \mathbf{V} \end{pmatrix} \quad \forall t \in \mathbf{N} \cup \{T\}.
\end{aligned}$$

Our modelling procedure reveals that we can formulate any MCDF problem on a GMPDNF as a problem which possesses the *block diagonal* or *angular* structure. It is approached conveniently by either the decomposition procedure or a technique referred to as generalized upper bounding [25][26][77] which is available on many commercial mathematical-programming systems. Block diagonal structure is very desirable because it can speed up the solution process for our linear programming problem. This structure may be exploited by splitting the original problem into smaller *subproblems* (those which form the diagonal) while having a *coupling constraint*, master constraint. In this setting, we call the first set of constraints as *master constraint*. The structure of the block-angular system suggests that we try to break the problem down into a set of some independent smaller parts and then adjust the solution to take into account the interconnections.

DW decomposition is very well suited for problems with this kind of structure. Interestingly enough, the master constraint in the matrix form (and subsequently, in the master problem) has the same matrix for any set of variables  $(\mathbf{X}^t, \mathbf{V}_+^t, \mathbf{V}_-^t)^{trans}$ . This plus the sparsity of the matrices are the most essential ingredients of this modelling approach, which facilitates to reach the optimal/suboptimal by decomposition method. Of course, it is not necessary for either set of constraints to have special structure, but when available, improves the efficiency of the procedure, which is the case in our problem. Thus, we may apply the block diagonal decomposition techniques to solve the foregoing problem to achieve the desired effect. For further discussion, one is referred to Chapter 3.

Let's consider an application of a decomposition algorithm to the problem. For this aim, define  $T+1$  polyhedral sets  $\chi^t$  for each  $t \in \{0, 1, \dots, T\}$  as:

$$\chi^t := \left\{ \begin{pmatrix} \mathbf{X}^t \\ \mathbf{V}_+^t \\ \mathbf{V}_-^t \end{pmatrix} : \begin{bmatrix} \mathbf{B}_{n \times m}^t & -\mathbf{I}_n & \mathbf{I}_n \end{bmatrix} \begin{pmatrix} \mathbf{X}^t \\ \mathbf{V}_+^t \\ \mathbf{V}_-^t \end{pmatrix} = [\mathbf{0}], \quad [\mathbf{0}] \leq \begin{pmatrix} \mathbf{X}^t \\ \mathbf{V}_+^t \\ \mathbf{V}_-^t \end{pmatrix} \leq \begin{pmatrix} \mathbf{U}^t \\ \mathbf{V} \\ \mathbf{V} \end{pmatrix} \right\}_{t=0,1,2,\dots,T}. \quad (2.37)$$

For the first case, let's assume that each component of  $[\mathbf{U}^t]$  is finite so that the polyhedral set  $\chi^t$  for each  $t \in \mathbb{N}$  is definitely bounded (polytop). Considering Caratheodory Theorem, any  $(\mathbf{X}^t, \mathbf{V}_+^t, \mathbf{V}_-^t)^t \in \chi^t$  can be expressed as a convex combination of a finite number of extreme points of  $\chi^t$ .

$$(\mathbf{X}^t, \mathbf{V}_+^t, \mathbf{V}_-^t)^{trans} = \sum_{i=1}^{k^t} \alpha_i^t [\mathbf{x}_i^t], \quad \sum_{i=1}^{k^t} \alpha_i^t = 1, \quad \text{and} \quad \alpha_i^t \geq 0 \quad i=1, \dots, k^t,$$

where  $[\mathbf{x}_1^t], [\mathbf{x}_2^t], \dots, [\mathbf{x}_{k^t}^t]$  are the extreme points of polytop  $\chi^t$  and  $\alpha_1^t, \alpha_2^t, \dots, \alpha_{k^t}^t$  denote Lagrange multipliers. Replacing each  $(\mathbf{X}^t, \mathbf{V}_+^t, \mathbf{V}_-^t)^{trans}$  by its corresponding convex representation yields the following equivalent formulation, *master problem*, of GMPDNF in an utterly different space of variables.

$$\begin{aligned}
& \mathbf{Min} \sum_{t \in N} \sum_{i=1}^{k^t} \alpha_i^t \cdot (\mathbf{C}^t, \mathbf{0}, \mathbf{0}) \cdot [\mathbf{x}_i^t], \\
& \begin{pmatrix} \mathbf{I}_m & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_m & \mathbf{I}_n & -\mathbf{I}_n \end{pmatrix} \sum_{i=0}^{k^0} \alpha_i^0 [\mathbf{x}_i^0] + \begin{pmatrix} \mathbf{I}_m & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_m & \mathbf{I}_n & -\mathbf{I}_n \end{pmatrix} \sum_{i=1}^{k^1} \alpha_i^1 [\mathbf{x}_i^1] + \dots + \begin{pmatrix} \mathbf{I}_m & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_m & \mathbf{I}_n & -\mathbf{I}_n \end{pmatrix} \sum_{i=1}^{k^{T-1}} \alpha_i^{T-1} [\mathbf{x}_i^{T-1}] = \begin{pmatrix} \mathbf{U}^T \\ \mathbf{V} \end{pmatrix}, \\
& \sum_{i=1}^{k^t} \alpha_i^t = 1 \quad \forall t \in N \cup \{T\}, \\
& \alpha_i^t \geq 0 \quad \forall t \in N \cup \{T\}, \quad i = 1, \dots, k^t.
\end{aligned}$$

Due to having a huge number of extreme points for each polyhedron set, enumerating all the extreme points, and directly solve this problem seems impossible. Rather, we should find a reasonable solution approach of the problem without enumerating all the extreme points. This is where we suggest decomposition techniques, especially due to the special structure of our problem that affects the memory usage of the decomposition methods. Let's go a further ahead and reduce our new problem to a *condensed master* problem.

$$\begin{aligned}
& \mathbf{Min} \sum_{t \in N} \sum_{i=1}^{k^t} \alpha_i^t \cdot (\mathbf{C}^t, \mathbf{0}, \mathbf{0}) \cdot [\mathbf{x}_i^t], \\
& \begin{pmatrix} \mathbf{I}_m & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_m & \mathbf{I}_n & -\mathbf{I}_n \end{pmatrix} \sum_{t \in N} \sum_{i=1}^{k^t} \alpha_i^t \cdot [\mathbf{x}_i^t] = \begin{pmatrix} \mathbf{U}^T \\ \mathbf{V} \end{pmatrix}, \\
& \sum_{i=1}^{k^t} \alpha_i^t = 1 \quad \forall t \in N \cup \{T\}, \\
& \alpha_i^t \geq 0 \quad \forall t \in N \cup \{T\}, \quad i = 1, \dots, k^t.
\end{aligned}$$

Since the vast majority of the  $\alpha_i^t$  variables are valued zero at any given iteration, most columns are irrelevant (that is, nonbasic) to the master. This leads to the heart of the decomposition algorithm. Only potentially useful columns are added to a so-called reduced *master problem*. For each (sparse) polyhedral, an independent LP is created, which is easy to solve. Assuming there are  $T$  subproblems in a DW implementation, at any iteration of the algorithm there are up to  $T$  potential columns to add to the reduced master formulation.



Note also that the new formulation of the MCDF problem shows a much simpler constraint structure than the usual matrix form. It possesses only  $m+n+T$  constraints rather than  $m+n+nT$  in the ex-form, and this again adds a point to the efficiency of the algorithms which may be applied to solve the problem on this new form.

**Observation 2.1** Considering the relationship between the extreme points and inner points stated in Caratheodory mapping, the optimal solution (corresponding to the optimal basis) obtained for GMPDNF from the master problem determines one set of original variables of form  $(\mathbf{X}^t, \mathbf{V}_+^t, \mathbf{V}_-^t)^{trans}$  for each time step conveying a positive flow [9][83].

**Remark 2.1** Any optimal basis will detect one arc set for every time step  $t$  that transports a positive amount of flow. Moreover, the values of  $v_i^t$  for each  $i$  and  $t$  will be determined at any basis, particularly in the optimal basis.

**Remark 2.2** It immediately follows that the optimal arc sets for every time step are not necessarily the same.

Furthermore, if we eliminate the restriction over components of  $[\mathbf{U}^t]$ , we may let the polyhedrals have some extreme directions and hence, under the Caratheodory mapping the original problem can be reformulated as the following *condensed master problem* where  $[\mathbf{d}_1^t], [\mathbf{d}_2^t], \dots, [\mathbf{d}_{k^t}^t]$  are extreme directions (if any) of polyhedral  $\chi^t$ .

$$\begin{aligned} & \mathbf{Min}_{\alpha_i^t, \mu_j^t} \sum_{t \in N \cup \{T\}} (\mathbf{C}^t, \mathbf{0}, \mathbf{0}) \left( \sum_{i=1}^{k^t} \alpha_i^t [\mathbf{x}_i^t] + \sum_{j=1}^{l^t} \mu_j^t [\mathbf{d}_j^t] \right), \\ & \begin{pmatrix} \mathbf{I}_m & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_m & \mathbf{I}_n & -\mathbf{I}_n \end{pmatrix} \sum_{t \in N \cup \{T\}} \left( \sum_{i=1}^{k^t} \alpha_i^t \cdot [\mathbf{x}_i^t] + \sum_{j=1}^{l^t} \mu_j^t \cdot [\mathbf{d}_j^t] \right) = \begin{pmatrix} \mathbf{U}^T \\ \mathbf{V} \end{pmatrix}, \\ & \sum_{i=1}^{k^t} \alpha_i^t = 1 \quad \forall t \in N \cup \{T\}, \\ & \alpha_i^t \geq 0 \quad \forall t \in N \cup \{T\} \quad i = 1, \dots, k^t, \\ & \mu_j^t \geq 0 \quad \forall t \in N \cup \{T\} \quad j = 1, \dots, l^t. \end{aligned}$$

Problems of this structure might be well amenable by many decomposition algorithms and so the computational advantage of the algorithm arises from the efficiency of the decomposition methods. We propose DW decomposition or Benders algorithm to obtain optimal solutions optimizing min-cost flow employing our polyhedral based approaches. As we proved, the original problem can be reformulated into a series of structurally similar LP subproblems, which may be solved employing GAMS.

The structural similarity of the subproblems and the sparsity of the associated matrices allow us to use any decomposition technique to well improve the computational resources. One of the other interesting features of this approach is that our algorithms considers the dynamic network problem as a linear programming problem which does not need the underlying graph to be connected, or does not force the data to be integer. In contrast with many other network algorithms, this one solves the (dynamic) problem under any sort of conditions. Our method basically relies on the modelling aspects of the multiperiod network flows, and then it tries to extract the special structure hidden in this kind of problems. As the last point regarding the algorithm, it should be mentioned that the problem in the primitive form could not have been solved efficiently due to the high dimension of the problem caused by the horizon time and size of the network. It seems as if the problem has many thousands of rows and unsolvable in a reasonable amount of time, however, our approach suggests a method to convert the large-scale (high dimensional problem) into one or more appropriately coordinated smaller problems of manageable sizes.

## **2.7 Multiperiod Networks with Storage and Spoilage (SS Networks)**

In this Section, we will introduce the most general case of multiperiod networks, i.e., MPDNF with storage at nodes and spoilage in arcs. We call these networks as *SS network flows*. The SS network flow problem is going to develop the multiperiod network flow problem by allowing the flow to leak and be stored, at the same time, as it is sent through the network. Naturally, the *min-cost SS network problem* in the continuous time setting will be of the form stated in (2.38)-(2.43)

$$\mathbf{Min}_{x_{ij}^t} \sum_{(i,j) \in A} \int_0^T c_{ij}(t) x_{ij}(t) dt + \sum_i \int_0^T c_i(t) \sum_j [x_{ij}(t) - \lambda_{ji}(t) x_{ji}(t)] dt \quad (2.38)$$

$$\sum_j \int_0^T x_{ij}(t) dt - \sum_j \int_0^T \lambda_{ji}(t) x_{ji}(t) dt = v_i \quad \forall i \in V, \quad (2.39)$$

$$\sum_j \int_0^\theta x_{ij}(t) dt - \sum_j \int_0^\theta \lambda_{ji}(t) x_{ji}(t) dt \leq 0 \quad \forall \theta \in [0, T[ \quad \forall i \in V \setminus VS, \quad (2.40)$$

$$\int_0^T x_{ij}(t) dt \leq u_{ij}^T \quad \forall (i, j) \in A, \quad (2.41)$$

$$0 \leq x_{ij}(t) \leq u_{ij}(t) \quad \forall (i, j) \in A \quad \forall t \in [0, T], \quad (2.42)$$

$$x_{ij}(t) = 0 \quad \forall (i, j) \in A \quad \forall t > T. \quad (2.43)$$

If we replace (2.31) and (2.32) by (2.44) and (2.45), we get the min-cost SS network problem in the discrete time setting subject to introducing free variable  $v_i^t := v_{i+}^t - v_{i-}^t$  ( $v_{i+}^t \geq 0, v_{i-}^t \geq 0$ ).

$$\mathbf{Min}_{x_{ij}^t, x_{ii}^t, v_i^t} \sum_{t \in \mathbb{N}} \sum_{(i,j) \in A} c_{ij}^t x_{ij}^t + \sum_{t \in \mathbb{N}} \sum_{i \in V} c_{ii}^t x_{ii}^t + \sum_{t \in \mathbb{N}} \sum_{i \in V} c_i^t v_i^t, \quad (2.44)$$

$$x_{ii}^t + \sum_j x_{ij}^t - \sum_j [\lambda_{ji}^t x_{ji}^t] - x_{ii}^{t-1} - v_i^t = 0 \quad \forall t \in \mathbb{N}, \forall i \in V. \quad (2.45)$$

To develop a polyhedral-based approach, we use again the two-level matrix transformation introduced in (2.36). It yields the *node-arc incidence matrix of SS network* with respect to each time step. Let  $[\mathbf{B}^t]_{n \times m}$  be the node-arc incidence matrix at step  $t$ . Therefore, having done all the necessary changes, the min-cost SS network problem can be modeled as that in previous section. The only difference will be the subproblems' matrices. It suffices to replace

$$[\mathbf{A}_{n \times m} \quad \mathbf{I}_n \quad -\mathbf{I}_n \quad \mathbf{I}_n]$$

by

$$[\mathbf{B}_{n \times m}^t \quad \mathbf{I}_n \quad -\mathbf{I}_n \quad \mathbf{I}_n] \text{ for any } t.$$

## 2.8 Examples, Applications, and Testing

Multiperiod dynamic flow problems arise frequently in Process Systems Engineering and dynamic generative network flows with essential applications in large production scheduling, multiperiod production planning, data/energy transmissions, and information communication technology. Dynamic Flows are widely used in modeling of control processes from different technical, electrical, economic and informational systems. Electricity and data transmissions, road or air traffic control, production systems, evacuation planning, production and distribution, telecommunication, transportation, communication, and management problems can be formulated and solved as single-commodity or multi commodity problems on (multiperiod) dynamic networks [6][16][17][29][32][35][57][65][71][72][80][81]. The need for more realistic network models led to the development of the dynamic network flow theory and nowadays, they have been applied to a variety of situations including production, crude oil transportation, inventory, and workforce models. When the above models involve a parameter of time horizon, they may often be modeled as dynamic or multiperiod flows.

Below, we give a potential application as a representative sampling. Consider a number of cities with demand for a certain good (that may vary over time or not) over a specified time period, e.g., demand for electricity -which we have been dealing with in our case study. We assume that demand is satisfied by shipping electricity in a fixed number of wires from a number of supply/production sites, where the cost of production is assumed to be time varying (or fixed for each time period).

We restrict our attention to the case in which each wire/cable/line must unload all of its goods (electricity) at the demand site upon arriving. The objective is to determine the production circuit and shipping electricity over the time period, so as to minimize the daily cost (horizon time was considered a day for our case study). We are going to consider the case for which capacities and costs change over time during a day. Note that the transmission times for electricity over wires are negligible, and so can be estimated as zero, if desired. The mentioned problem may be formulated using the techniques discussed in this section as a MP network flow if loss is expected, or as that discussed earlier or storage assumption can/must be applied in sites. The problem can be formulated as follows: (1) Production site  $i$  has a fixed amount of supplies  $v_i$  over time

horizon  $T$  (Horizon time = a day); (2) Demand site  $i$  has a fixed demand  $v'_i$  for electricity over a day for; (3) The number of wires can be bounded above and below; (4) Storage can be allowed at the production and demand sites at a cost, or not; (5) Losses may be expected at wires or not (it is about zero in this case study). Constraints (3) and (4) are related in that flow is measured in goods traveling over time; if storage is allowed, then storage will be interpreted as throughput as will a wire traveling. Simultaneously, allowing both (3) and (4) results in a problem that is NP-complete even for problems with exactly one wire. The NP-completeness can be proved via a transformation from the traveling salesman problem.

The static network has node set  $\{1,2,\dots,n\}$ . For each production site  $i$  and demand site  $j$ , there are arcs  $(i, j)$  and  $(j, i)$  with zero transit times. Thus the static network is a complete directed bipartite graph. For each arc  $(i, j)$  there is an additional constraint that the flow into arc in each time period is bounded above and below. To test the applicability of the proposed models, we conducted a series of experiments using a set of real data from our case study on grid networks and on random MP networks. Several parameters must be specified in order to generate the network topology, arc capacities and costs, losses and gains, and node storage capacities (if desired). These parameters are random seed, time horizon  $T$ , number of supply/demand nodes, indegree and outdegree of each node, minimum and maximum values of arc capacities, losses, gains, and costs, which all must be nonnegative. The cost on each arc (for each time period) is randomly chosen from a uniform distribution between user defined parameter  $c_{\min}$  and  $c_{\max}$ , and gain/loss factors for each period are also chosen from a  $[0, \lambda_{\max}]$ -uniform distribution where  $\lambda_{\max}$  is given.

The user also sets the number of time periods, supply nodes, and demand nodes. Then, we may randomly select as many source-sink pairs as desired. The demand for each time step is to be randomly chosen from a uniform distribution between  $v_{\min}$  and a pre-given parameter  $v_{\max}$ , and likewise for supply, storage capacity, spoilage, arc capacities. The experiments are conducted on random networks with 20, 26, 30, 40, 46, 62, 74, 82, and 100 nodes and time horizon for each case is set to be 10, 13, 15, 20, 23, 31, 37, 41, and 50. For each choice of  $n$  nodes, we create networks with different indegree and outdegree in a range from 2 to 8. We denote by  $\delta$  the density of the

network, that is  $\delta = m/n$ . The minimum and maximum loss/gain is set to 0 and 2, respectively, the minimum and maximum capacity is set to 50 and 70, respectively, the minimum and maximum cost is set to 1 and 10. For each specific setting of  $n$  and  $m$ , we test a random MP network.

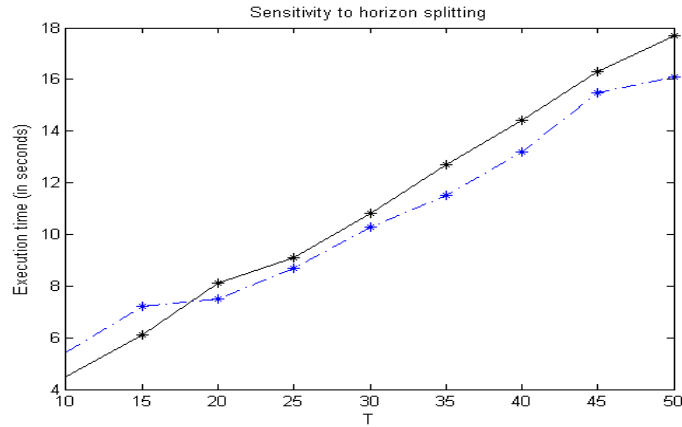


Figure 2.3 Sensitivity to the number of time increments (a case study problem(dotted line) vs. a random network (solid line))

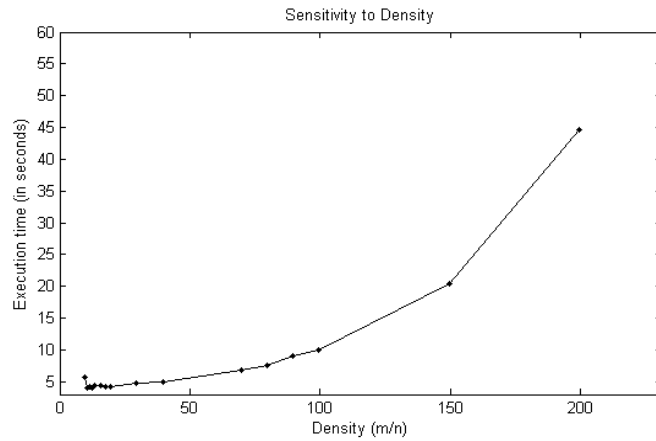


Figure 2.4 Sensitivity to density

In order to better illustrate the sensitivity of the method to various data parameters (number of time increments and network density), Figure 2.3 and Figure 2.4 give the plots of execution time with respect to each of them for increasing larger problem instances. A plot for each different network parameter,  $\delta$  and  $T$ , helps us visualize the effects of time splitting and density on the growth of the problem or the average CPU time. Figure 2.4 shows that CPU time increases exponentially in denser networks. In addition, we have performed several computational tests and analyzed decomposition

approach on a variety of MPDNF problem instances. These tests have included the investigation of different implementation ideas and the testing of the sensitivity of the method to various data parameters, such as the number of arcs, number of time increments and the congestion in the multiperiod network. We generated many MPDNF of various sizes, different number of time periods, and different levels of congestion. Each generated network has an underlying random-complete form or a grid graph. For each grid network, the length  $L$  (the number of arcs in each horizontal line) and the height  $H$  (the number of arcs in each vertical line) are user defined. Each grid network is derived from an undirected network, i.e., there are two oppositely directed arcs between each pair of connected nodes. Our linear programming models, decomposition methods, were implemented in GAMS/Cplex on a personal computer with a 3GHz processor and 4GB physical RAM. The results of a very small number of runs are summarized in Table 2.1. We let the storage be zero at any node in any moment of time. Computational experiences shown in the first four rows correspond to some grid MPDNFs from our case study having parameters. The remaining rows represent the results for some multiperiod random networks.

**Table 2.1 Sizes and computational results**

	<i>Number of Polyhedrals</i>	<i>Number of Variables</i>	<i>Number of Constraints</i>	<i>Number of Non-zeros</i>	<i>Work space Allocated (Mb)</i>	<i>Computational Time (s)</i>
<i>Data set 1</i>	11	2490	410	401	1.4	05.45
<i>Data set 2</i>	14	5226	689	677	1.4	07.31
<i>Data set 3</i>	16	7860	915	901	1.4	08.05
<i>Data set 4</i>	21	17980	1620	1601	1.9	14.98
<i>Data set 5</i>	24	26956	2047	2117	1.9	21.87
<i>Data set 6</i>	32	64356	3875	2853	3.0	20.02
<i>Data set 7</i>	38	108114	5513	5477	4.5	58.04
<i>Data set 8</i>	42	146206	6765	6725	5.6	76.09
<i>Data set 9</i>	51	262450	10050	10001	8.2	68.09

Also observe that, in practice, this approach usually develops a very good approximation quickly, but then expends considerable effort to refine it (Figure 2.5). In a practical application, there is often the need to get solutions quickly, within a given time, even if it means compromising the quality of the solutions. DW method provides feasible solutions if halted prematurely, and the quality of the solutions improves monotonically as it progresses.

**Remark 2.3** It is proved [9][83] that the DW algorithm provides an upper bound (dual bound) on the value of the objective function at each iteration, which allows the quality of the current solution to be assessed, so that the trade of between time and quality can be quantified. Tebboth [83] proved that given a dual (optimal) solution to the original problem there is an obvious corresponding dual (optimal) solution to the master problem (by a simple projection) and vice versa. Applying his argument to our case yields that if  $(\mu_1, \nu)$  is dual feasible for the original problem, then  $(\mu_1, \mu_2)$  is dual feasible for the master problem, where  $\mu_2^{(t)} = \nu_t b_t$  ( $b$ : rhs vector). Conversely, if  $(\mu_1, \mu_2)$  is dual feasible for the master problem, then  $(\mu_1, \nu)$  is dual feasible for the original problem, where  $\nu_t$  is a dual solution to subproblem problem  $t$ . Therefore,  $(\mu_1, \nu)$  is a dual feasible solution to the original problem, and the corresponding dual objective value  $\underline{z} = \mu_1 b_0 + \nu_1 b_1 + \dots + \nu_T b_T$  is a dual (upper) bound on the optimal value of the original problem, and thus also of the master problem. Hence, at each iteration of the algorithm, once the restricted master problem is feasible, a new dual solution of the restricted master problem is available and we can calculate a new dual bound. It provides a guarantee on the quality of the current solution to the restricted master problem [83]. In practice, a near optimal solution may be acceptable and so the algorithm can be terminated early once the gap (The difference between the primal value of the restricted master problem and the least dual bound obtained during the course of the algorithm) falls to within an acceptable limit.

**Remark 2.4** One particularly attractive feature of decomposition, in contrast to the simplex method, is the relatively straightforward potential for a parallel computational version. The pricing phase of the algorithm consists of solving a set of mutually independent LP problems, which can also be accomplished in parallel [76][83]. The parallelism is coarse grained, uses distributed memory, and is ideally suited to networks of serial computers that are in common everyday use. Furthermore, provided that the work to solve the pricing problems is not trivial, and, in particular, it is not dominated by the work to solve the master problem, the parallelism is likely to be efficient. To know more about parallelism and different parallelism strategies of DW, please read the last section of Chapter 3.



Generally, since the convergence of this approach has turned out to be slow in the final stages, such a termination procedure might be of great importance to employ. Consequently, when decomposition is applied on a min-cost MPDNF problem, the objective value usually decreases rapidly and then slowly tails off by approaching the optimal objective value. Figure 2.5 plots the progress of a random problem for an application of the decomposition method showing how the objective values and upper bounds converge as the number of iterations (time) increases.

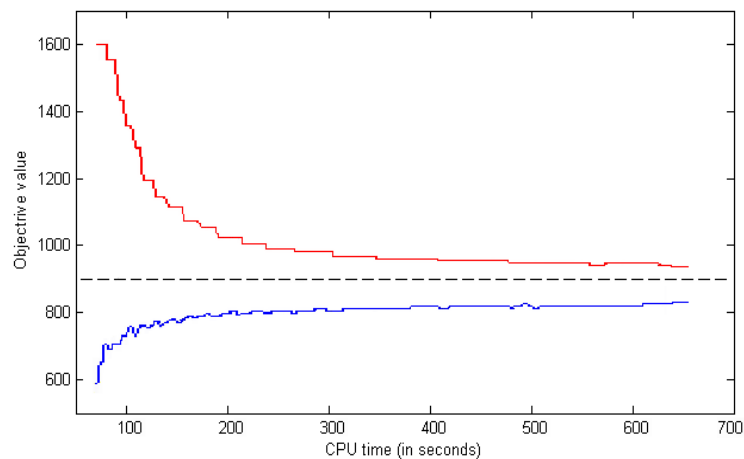


Figure 2.5 Objective and upper bound progress of decomposition application for a random problem

The problems are min-cost flow problems, so the objective value descends from above while the bound ascends from below. The plots show how much more rapid the convergence is with the set decomposition, and the length of the tail of the time decomposition. Figure 2.5 also shows the relative gap between the objective value and dual bound as the algorithm progresses.

## 2.9 Summary and Concluding Remarks

This chapter addresses discrete-time dynamic min-cost flow problem on multiperiod network flows under the generalization of node storage or/and arc spoilage. We have developed some model-based approaches to solve the dynamic min-cost flow problem employing polyhedral sets hidden in the underlying network structure. By generating block structures from the index sets of the model, the original problem(s) is reformulated into a series of structurally similar sparse LP subproblems (polyhedrals), which are solved by GAMS/CPLEX/DW. The structural similarity of the subproblems

allows us to use decomposition techniques to well improve the computational resources. We propose some algebraic approaches by plotting the non-zero and zero elements in the constraint matrix of a problem instance to generate the blocks of interest.

What is important to note is that our approach is a two-phase method. The first phase is the matrix transformation/decomposition, and the second one is an application of DW method. Needless to say, the performance of our algorithm highly depends on the two-level matrix transformation we already introduced. A simple analysis reveals that the running time needed for all transformations is  $O(Tnm^2)$ . However, for more general cases, where the underlying MP network has no loss/gain on its arcs or has fixed loss/gain during the planning time, the running time can be improved to  $O(nm^2)$  if a simple data structure is used to maintain the factors for each time period. As mentioned before, although the nature of the MP network is time varying, all the node-arc incidence matrices and matrix transformations can be updated/run off-line and parallel. Therefore, if the solutions of the first phase are calculated in parallel, we can expect to obtain the optimal solution for any min-cost MPDNF problem in a reasonable time, as shown in Table **2.1**.

# Chapter 3

## 3 A Decomposition-Based Approach for the Multiple-Product Distribution Problems over Time

In this chapter, we will attack the most general case of multiperiod multiproduct network planning problems, where we allow spoilage on arcs and/or storage at nodes. In our models, all network parameters change over time and products. The minimum-cost flow problem in the discrete-time model with varying network parameters is investigated when we allow storage and or spoilage, and some reformulation techniques employing polyhedrals are developed to obtain optimal solutions for a predefined horizon. Our methods rely on appropriate definitions of polyhedrals and identification of block structures that lead to LP problems comprising a set of sparse subproblems with repeated components.

Little has been written on identifying block structure in practical large LP problems. We show how block structure can be inferred from the algebraic model description of the problem, by recognizing that underlying structure in the problem is identified through index sets in the model. Our method is to consider a plot of the non-zero and zero elements in the constraint matrix of a problem instance. Matrix decomposition is utilized to illustrate the transformation from the original problem to the Dantzig-Wolfe master problem and to establish how solutions are obtained from the decomposition algorithm correspond to solutions of the original problem.

Solving LP problems is quite memory intensive. When memory is limited, GAMS/Cplex automatically makes adjustments which may negatively impact performance. Having properly decomposed constraints into the (sparse) blocks, computational expenses of solving such large-scale planning problems can be decreased by using decomposition techniques [9][11][15][83]. On the other hand, modern computers can also take advantage of the algorithm's inherent parallelism to efficiently improve the elapsed time.

In a very recent inspiring work, Rios [76] has reported computational results to motivate use of such parallel solvers, as this implementation outperforms state-of-the-art commercial solvers (like CPLEX) in terms of wall-clock runtime by an order of magnitude or more on several problem instances. Applying his approach (in implementation) and ours (in decomposing) simultaneously will sufficiently demonstrate the utility of our approach.

The results of this chapter are published in Hosseini et al. [37].

### **3.1 The Problem of Min-Cost Flow on Multiperiod Multiproduct Networks**

The single-commodity multiperiod distribution problem, discussed in previous chapter, can be extended to the multi-commodity multiperiod distribution network flow problem. Motivated by time-dependent multi-item distribution planning problems, we study an extension of multiperiod flow problems as a generalization of Hosseini [36][41] by including *horizon capacities*, *time-commodity varying capacities*, *time-commodity varying costs*, and *time-commodity varying loss/gain factors* over a finite horizon.

This chapter focuses on the *minimum cost dynamic flows* (MCDF) on *multiperiod multiproduct networks* (MMN), in which spoilage/storage in arcs/nodes is expected/allowed. In such applications, each arc is assigned a non-negative time-commodity varying *gain/loss factor*, two non-negative time and time-commodity varying capacity functions, a non-negative horizon capacity, and a non-negative time-commodity varying cost function. In our setting, a positive time-commodity varying factor represents the fraction of flow that remains when it is sent at a specific time period. We consider non-simultaneous shipment of commodities from production sites (sources) to markets (sinks) in a distribution network such that no capacity conditions are violated and this time-commodity dependent shipment should optimally happen in a pre-defined horizon. Hence, the problem here is a decision problem aiming to find a dynamic flow minimizing a pre-defined non-negative distribution cost function.

We assume that the size of the problem prohibits its direct solution; and we develop algorithmic approaches based on block-angular and block-staircase decomposition techniques as alternatives to overcome this challenge. Relatively little attention has been

devoted to the issue of problem formulation while considerably more attention is given to improvements in algorithms for a given solution method. Having this in mind, we show that any MCDF problem on a MMN can be formulated as an LP problem with special structures that permit efficient computation of its solution and help save storage requirements.

In the MCDF problem on MMN, given is a set of products that are manufactured in several multiproduct production sites (sources) and shipped to a set of markets (sinks) where they are sold, and the objective is to find a routing plan to non-simultaneously ship the products from source nodes to sink nodes through a distribution network without exceeding the arc capacities (time-varying, time-commodity varying, and horizon capacities) at the minimal cost during a finite-length planning horizon. Here,  $G=(V, A, T, K)$  denotes a distribution (directed) network where  $v$  is the set of production and demand sites (nodes),  $A$  is the set of all possible connections between sites (arcs),  $K=\{1,2,\dots,K\}$  is the set of products, and  $T$  represents the length of the planning horizon. Then, by describing the dynamic flow decision variable  $x_{ijq}(t)$  as the vector of flow rates of commodity  $q$  entering arc  $(i, j)$  at time period  $t$ , the formulation for the MCDF on MMN becomes as

$$\text{Continuous-time-MCDF} \quad \mathbf{Min}_{x_{ijq}(t)} \sum_{q \in K} \sum_{(i,j) \in A} \int_0^T c_{ijq}(t) x_{ijq}(t) dt, \quad (3.1)$$

$$\sum_j \int_0^T x_{ijq}(t) dt - \sum_j \int_0^T x_{jiq}(t) dt = v_{iq} \quad \forall i \in V, \forall q \in K, \quad (3.2)$$

$$\sum_{q \in K} \int_0^T x_{ijq}(t) dt \leq u_{ij} \quad \forall (i, j) \in A, \quad (3.3)$$

$$\sum_{q \in K} x_{ijq}(t) \leq u_{ij}(t) \quad \forall (i, j) \in A, \forall t \in [0, T], \quad (3.4)$$

$$0 \leq x_{ijq}(t) \leq u_{ijq}(t) \quad \forall (i, j) \in A, \forall q \in K, \forall t \in [0, T], \quad (3.5)$$

$$x_{ijq}(t) = 0 \quad \forall (i, j) \in A, \forall q \in K, \forall t > T. \quad (3.6)$$

In this setting,  $c_{ijq} : [0, T] \rightarrow \mathbb{R}^+$  is the non-negative cost function with respect to product  $q$ , and  $v_{iq}$  denotes the pre-defined supply/demand capacities at node  $i$  over the entire time horizon. Constraint (3.2) involves the flow conservation constraints for each commodity. We refer to (3.3) as *horizon capacity* constraints. Horizon capacity of an arc limits the amount of total flow (of all commodities) on the arc throughout the entire horizon. Constraint (3.4) represents the maximum possible amount of total flow that can enter  $(i, j)$  at time  $t$ : it is referred to as the *moment/period capacity* constraint. Constraint (3.5) is the time-commodity varying capacity constraint for each commodity at each moment. The domain of decision variables prescribed in (3.6) also emphasizes that commodities can flow on the network only until the end of pre-specified time horizon.

The problem formulation in (3.1)-(3.6) represents MCDF in a continuous-time setting. However, as an approximation to this setting time may be represented in discrete increments. By using a similar transformation discussed in Section 2.2, a continuous-time multiproduct flow  $x$  can be estimated by a discrete multiproduct flow  $\bar{x}$  and vice versa. Let  $\bar{x}_{ijq}(t)$  represent the total amount of flow sent into arc  $(i, j)$  during time interval  $[t, t+1[$ , then

$$\begin{aligned} \bar{x}_{ijq}(t) &:= \int_t^{t+1} x_{ijq}(\xi) d\xi \quad \text{and} \quad \bar{u}_{ijq}(t) := \int_t^{t+1} u_{ijq}(\xi) d\xi \quad \forall q \in \mathbb{K}, \text{ and} \\ \bar{c}_{ijq}(t) &:= c_{ijq}(\xi_t) \quad \forall q \in \mathbb{K}, \end{aligned} \tag{3.7}$$

where  $\xi_t \in ]t, t+1[$ ,  $t \in \mathbb{N}$ .

### 3.2 Multiperiod Multiproduct Network Flows with Spoilage (SMMN)

Any of the models discussed up to now, has a fundamental assumption, namely, flow has to be conserved on any arc with respect to any commodity. However, some practical applications do not satisfy such a conservation assumption [3]. In the transmission of a volatile gas, for example, we may lose some portion of the flow due to evaporation; or, in the transmission of liquids such as raw petroleum crude, some flow may be lost due to leakage. In the setting where spoilage on arcs is also considered, each arc  $(i, j)$  has a time-commodity varying non-negative *gain/loss factor*  $\lambda_{ijq}(t)$  with respect to each time period time  $t$  and commodity  $q$ . When  $x_{ijq}(t)$  units of flow of commodity  $q$  is sent from

node  $i$  via arc  $(i, j)$  at time  $t$ ,  $\lambda_{ijq}(t)x_{ijq}(t)$  units of flow arrive at node  $j$  at the same time. If  $\lambda_{ijq}(t) < 1$ , the arc is *lossy*; if  $\lambda_{ijq}(t) \geq 1$  the arc is *gainy* on that time with respect to that commodity. Therefore, if there is flow spoilage (loss) on an arc during all periods with respect to all commodities, then the model may be represented as the initial one by assigning a *loss factor* to the related arc.

$$\sum_j \int_0^T x_{ijq}(t) dt - \sum_j \int_0^T \lambda_{jiq}(t) x_{jiq}(t) dt = v_{iq} \quad \forall i \in V, \forall q \in K. \quad (3.8)$$

Such a production-distribution planning problem in discrete-time setting has an underlying graph  $G=(V, A, T, K)$  consisting of three sorts of capacity functions  $u_q^t: A \times N \times K \rightarrow IR^+$ ,  $u^t: A \times N \rightarrow IR^+$  and  $u: A \rightarrow IR^+$ , pre-defined non-negative cost function  $c_q^t: A \times N \times K \rightarrow IR^+$ , and pre-defined non-negative time-commodity varying gain/loss function  $\lambda_q^t: A \times N \times K \rightarrow IR^+$ . Therefore, a *discrete feasible dynamic flow* is a non-negative function  $x = \{x_{ijq}^t\}: A \times N \times K \rightarrow IR^+$  satisfying (3.10)-(3.14), and the discrete-time minimum-cost dynamic flow problem becomes as:

$$\text{Discrete-time-MCDF} \quad \text{Min}_{x_{ijq}^t} \sum_{q \in K} \sum_{t \in N} \sum_{(i,j) \in A} c_{ijq}^t x_{ijq}^t, \quad (3.9)$$

$$\sum_j \sum_{t=0}^{T-1} x_{ijq}^t - \sum_j \sum_{t=0}^{T-1} \lambda_{jiq}^t x_{jiq}^t = v_{iq} \quad \forall i \in V, \forall q \in K, \quad (3.10)$$

$$\sum_{q \in K} \sum_{t \in N} x_{ijq}^t \leq u_{ij} \quad \forall (i, j) \in A, \quad (3.11)$$

$$\sum_{q \in K} x_{ijq}^t \leq u_{ij}^t \quad \forall (i, j) \in A, \forall t \in N, \quad (3.12)$$

$$0 \leq x_{ijq}^t \leq u_{ijq}^t \quad \forall (i, j) \in A, \forall q \in K, \forall t \in N, \quad (3.13)$$

$$x_{ijq}^t = 0 \quad \forall (i, j) \in A, \forall t \notin N, \forall q \in K. \quad (3.14)$$

The equations stated in (3.10)-(3.14) represent the flow feasibility conditions in a SMMN. Here, we assume that the arc capacity  $u_{ijq}^t$  is an upper bound on the  $q$ -flow (flow of commodity  $q$ ) sent from node  $i$  at time period  $t$ , not on the flow that becomes available at node  $j$ . Similarly,  $c_{ijq}^t$  should be interpreted as the cost for each unit of flow

which is sent from node  $i$ . In order to benefit from an efficient decomposition-based solution method by transforming the formulation structure, we introduce an unrestricted variable  $v_{iq}^t$ , and the formulation becomes

$$\mathbf{Min}_{x_{ijq}^t, v_{iq}^t} \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} \sum_{(i,j) \in \mathbf{A}} c_{ijq}^t x_{ijq}^t, \quad (3.15)$$

$$\left( \sum_j x_{ijq}^t - \sum_j \lambda_{jiq}^t x_{jiq}^t \right) - v_{iq}^t = 0 \quad \forall i \in \mathbf{V}, \forall q \in \mathbf{K}, \forall t \in \mathbf{N}, \quad (3.16)$$

$$\sum_{t \in \mathbf{N}} v_{iq}^t = v_{iq} \quad \forall i \in \mathbf{V}, \forall q \in \mathbf{K}, \quad (3.17)$$

$$\sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t \leq u_{ij} \quad \forall (i, j) \in \mathbf{A}, \quad (3.18)$$

$$\sum_{q \in \mathbf{K}} x_{ijq}^t \leq u_{ij}^t \quad \forall (i, j) \in \mathbf{A}, \forall t \in \mathbf{N}, \quad (3.19)$$

$$0 \leq x_{ijq}^t \leq u_{ijq}^t \quad \forall (i, j) \in \mathbf{A}, \forall q \in \mathbf{K}, \forall t \in \mathbf{N}, \quad (3.20)$$

$$x_{ijq}^t = 0 \quad \forall (i, j) \in \mathbf{A}, \forall t \notin \mathbf{N}, \forall q \in \mathbf{K}, \quad (3.21)$$

where  $v_{iq}^t$  denotes the difference between the outflow and the inflow of commodity  $q$  at node  $i$  at period  $t$ . It is easy to check that conditions (3.16) and (3.17) together are equivalent to condition (3.10). To develop our polyhedral-based approach, we need to define the node-arc incidence matrix including the time dimension of the problem. Given an underlying SMM network and pre-defined loss/gain factors, introduce an auxiliary matrix  $[\Phi_q^t]_{n \times m}$  for a time period  $t$  and for a product  $q$  as

$$[\Phi_q^t]_{n \times m} := [\mathbf{A}]_{n \times m} \times [\Sigma_q^t]_{m \times m}, \quad (3.22)$$

where  $[\mathbf{A}]$  is the node-arc incidence matrix of the underlying distribution network (which remains unchanged during the planning horizon) and  $[\Sigma_q^t]$  is a  $m \times m$ -diagonal matrix whose elements are the pre-defined arc factors (at time  $t$  with respect to  $q$ ) in the same order that arcs appear in  $m$ -vector  $[\mathbf{X}_q^t]_{m \times 1}$  ( $m$  is the number of arcs of the network). For a time period  $t$  and a commodity  $q$ , we construct  $[\mathbf{B}_q^t]_{n \times m}$  as



$$[\mathbf{B}_q^t]_{ij} := \begin{cases} [\Phi_q^t]_{ij} & \text{if } [\Phi_q^t]_{ij} \leq 0, \\ 1 & \text{if } [\Phi_q^t]_{ij} > 0, \end{cases} \quad (3.23)$$

where  $[\mathbf{B}_q^t]_{ij}$  and  $[\Phi_q^t]_{ij}$  represent the  $(i, j)$ -th elements of the matrices  $[\mathbf{B}_q^t]$  and  $[\Phi_q^t]$ , respectively. We refer to matrix  $[\mathbf{B}_q^t]_{n \times m}$  as the  $t$ - $q$ -node-arc incidence matrix of the SMM network, and it represents the node-arc incidence matrix of SMMN at time  $t$  for commodity  $q$ . Due to the changes in arc factors over time and commodity, incident matrices are not necessarily the same during the complete planning horizon with respect to each product. However, since the time-commodity varying gain/loss functions are pre-defined, all matrices can be computed off-line. In order to illustrate, let's consider the  $t$ - $q$ -incidence matrix of the SMM network presented in Figure 3.1 for a fixed period and product.

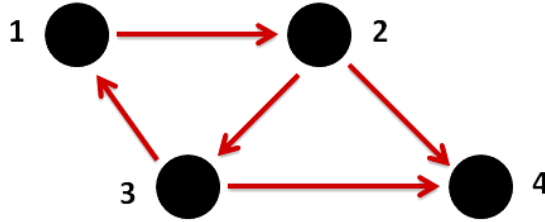


Figure 3.1 A typical SMMN at a fixed period  $t$  for a fixed product  $q$

Suppose  $[\mathbf{V}_q^t] = \{v_{iq}^t\}_{i=1,2,3,4}$  is the vector of supply/demand at time  $t$  for commodity  $q$ . Based on the network information and the distribution network's topology, we get the vector of flows  $[\mathbf{X}_q^t] = \{x_{ijq}^t\}$  and the *gain/loss* matrix  $[\Sigma_q^t]$  as following.

$$\begin{array}{c} \mathbf{X}_q^t \\ \left[ \begin{array}{c} x_{12q}^t \\ x_{31q}^t \\ x_{23q}^t \\ x_{24q}^t \\ x_{34q}^t \end{array} \right] \end{array} \quad \text{and} \quad \begin{array}{c} \mathbf{V}_q^t \\ \left[ \begin{array}{c} v_{1q}^t \\ v_{2q}^t \\ v_{3q}^t \\ v_{4q}^t \end{array} \right] \end{array} \quad \text{and} \quad \begin{array}{c} \Sigma_q^t \\ \left[ \begin{array}{ccccc} \lambda_{12q}^t & 0 & 0 & 0 & 0 \\ 0 & \lambda_{31q}^t & 0 & 0 & 0 \\ 0 & 0 & \lambda_{23q}^t & 0 & 0 \\ 0 & 0 & 0 & \lambda_{24q}^t & 0 \\ 0 & 0 & 0 & 0 & \lambda_{34q}^t \end{array} \right] \end{array} .$$

Considering (4.23), we may construct the ancillary matrix  $[\Phi_q^t]_{4 \times 5}$  and the  $t$ - $q$ -node-arc incidence matrix  $[\mathbf{B}_q^t]$  for each  $t$  and  $q$  as

$$\begin{aligned}
& [\Phi_q^t] := \\
& \begin{bmatrix} +1 & -1 & 0 & 0 & 0 \\ -1 & 0 & +1 & +1 & 0 \\ 0 & +1 & -1 & 0 & +1 \\ 0 & 0 & 0 & -1 & -1 \end{bmatrix} \begin{bmatrix} \lambda_{12q}^t & 0 & 0 & 0 & 0 \\ 0 & \lambda_{31q}^t & 0 & 0 & 0 \\ 0 & 0 & \lambda_{23q}^t & 0 & 0 \\ 0 & 0 & 0 & \lambda_{24q}^t & 0 \\ 0 & 0 & 0 & 0 & \lambda_{34q}^t \end{bmatrix} = \begin{bmatrix} \lambda_{12q}^t & -\lambda_{31q}^t & 0 & 0 & 0 \\ -\lambda_{12q}^t & 0 & \lambda_{23q}^t & \lambda_{24q}^t & 0 \\ 0 & \lambda_{31q}^t & -\lambda_{23q}^t & 0 & \lambda_{34q}^t \\ 0 & 0 & 0 & -\lambda_{24q}^t & -\lambda_{34q}^t \end{bmatrix}. \\
& [\mathbf{B}_q^t] := \begin{bmatrix} +1 & -\lambda_{31q}^t & 0 & 0 & 0 \\ -\lambda_{12q}^t & 0 & +1 & +1 & 0 \\ 0 & +1 & -\lambda_{23q}^t & 0 & +1 \\ 0 & 0 & 0 & -\lambda_{24q}^t & -\lambda_{34q}^t \end{bmatrix}.
\end{aligned}$$

It is now easy to see that

$$[\mathbf{B}_q^t][\mathbf{X}_q^t] = \begin{bmatrix} +1 & -\lambda_{31q}^t & 0 & 0 & 0 \\ -\lambda_{12q}^t & 0 & +1 & +1 & 0 \\ 0 & +1 & -\lambda_{23q}^t & 0 & +1 \\ 0 & 0 & 0 & -\lambda_{24q}^t & -\lambda_{34q}^t \end{bmatrix} \begin{bmatrix} x_{12q}^t \\ x_{31q}^t \\ x_{23q}^t \\ x_{24q}^t \\ x_{34q}^t \end{bmatrix} = \begin{bmatrix} x_{12q}^t - \lambda_{31q}^t x_{31q}^t \\ x_{23q}^t + x_{24q}^t - \lambda_{12q}^t x_{12q}^t \\ x_{31q}^t + x_{34q}^t - \lambda_{23q}^t x_{23q}^t \\ -\lambda_{24q}^t x_{24q}^t - \lambda_{34q}^t x_{34q}^t \end{bmatrix} = \begin{bmatrix} v_{1q}^t \\ v_{2q}^t \\ v_{3q}^t \\ v_{4q}^t \end{bmatrix} = [\mathbf{V}_q^t].$$

Substituting  $v_{iq+}^t - v_{iq-}^t$  for  $v_{iq}^t$  ( $v_{iq+}^t \geq 0, v_{iq-}^t \geq 0$ ), considering relations (3.22)-(3.23), and having defined some appropriate vectors and matrices, the formulation for the problem becomes:

$$\begin{aligned}
& \underset{\mathbf{x}_q^t, \mathbf{v}_{q+}^t, \mathbf{v}_{q-}^t}{\text{Min}} \sum_{t \in \mathbf{N}} [\mathbf{C}_1^t \quad \mathbf{C}_2^t \quad \dots \quad \mathbf{C}_K^t] [\mathbf{x}_1^t \quad \mathbf{x}_2^t \quad \dots \quad \mathbf{x}_K^t]^{trans}, \\
& ([\mathbf{X}_1^0] + \dots + [\mathbf{X}_K^0]) + ([\mathbf{X}_1^1] + \dots + [\mathbf{X}_K^1]) + \dots + ([\mathbf{X}_1^{T-1}] + \dots + [\mathbf{X}_K^{T-1}]) \leq [\mathbf{U}]_{m \times 1}, \\
& \begin{cases} ([\mathbf{V}_{1+}^0] - [\mathbf{V}_{1-}^0]) + ([\mathbf{V}_{1+}^1] - [\mathbf{V}_{1-}^1]) + \dots + ([\mathbf{V}_{1+}^{T-1}] - [\mathbf{V}_{1-}^{T-1}]) = [\mathbf{V}_1]_{n \times 1} \\ ([\mathbf{V}_{2+}^0] - [\mathbf{V}_{2-}^0]) + ([\mathbf{V}_{2+}^1] - [\mathbf{V}_{2-}^1]) + \dots + ([\mathbf{V}_{2+}^{T-1}] - [\mathbf{V}_{2-}^{T-1}]) = [\mathbf{V}_2]_{n \times 1} \\ \vdots \\ ([\mathbf{V}_{K+}^0] - [\mathbf{V}_{K-}^0]) + ([\mathbf{V}_{K+}^1] - [\mathbf{V}_{K-}^1]) + \dots + ([\mathbf{V}_{K+}^{T-1}] - [\mathbf{V}_{K-}^{T-1}]) = [\mathbf{V}_K]_{n \times 1} \end{cases} \\
& \begin{cases} [\mathbf{B}_1^0]_{n \times m} [\mathbf{X}_1^0] - [\mathbf{I}]_n [\mathbf{V}_{1+}^0] + [\mathbf{I}]_n [\mathbf{V}_{1-}^0] = [\mathbf{0}]_{n \times 1} \\ [\mathbf{B}_2^0]_{n \times m} [\mathbf{X}_2^0] - [\mathbf{I}]_n [\mathbf{V}_{2+}^0] + [\mathbf{I}]_n [\mathbf{V}_{2-}^0] = [\mathbf{0}]_{n \times 1} \\ \vdots \\ [\mathbf{B}_K^0]_{n \times m} [\mathbf{X}_K^0] - [\mathbf{I}]_n [\mathbf{V}_{K+}^0] + [\mathbf{I}]_n [\mathbf{V}_{K-}^0] = [\mathbf{0}]_{n \times 1} \end{cases}
\end{aligned}$$

$$\begin{cases}
[\mathbf{B}_1^1]_{n \times m} [\mathbf{X}_1^1] - [\mathbf{I}]_n [\mathbf{V}_{1+}^1] + [\mathbf{I}]_n [\mathbf{V}_{1-}^1] = [\mathbf{0}]_{n \times 1} \\
[\mathbf{B}_2^1]_{n \times m} [\mathbf{X}_2^1] - [\mathbf{I}]_n [\mathbf{V}_{2+}^1] + [\mathbf{I}]_n [\mathbf{V}_{2-}^1] = [\mathbf{0}]_{n \times 1} \\
\vdots \\
[\mathbf{B}_K^1]_{n \times m} [\mathbf{X}_K^1] - [\mathbf{I}]_n [\mathbf{V}_{K+}^1] + [\mathbf{I}]_n [\mathbf{V}_{K-}^1] = [\mathbf{0}]_{n \times 1} \\
\vdots \\
[\mathbf{B}_1^{T-1}]_{n \times m} [\mathbf{X}_1^{T-1}] - [\mathbf{I}]_n [\mathbf{V}_{1+}^{T-1}] + [\mathbf{I}]_n [\mathbf{V}_{1-}^{T-1}] = [\mathbf{0}]_{n \times 1} \\
[\mathbf{B}_2^{T-1}]_{n \times m} [\mathbf{X}_2^{T-1}] - [\mathbf{I}]_n [\mathbf{V}_{2+}^{T-1}] + [\mathbf{I}]_n [\mathbf{V}_{2-}^{T-1}] = [\mathbf{0}]_{n \times 1} \\
\vdots \\
[\mathbf{B}_K^{T-1}]_{n \times m} [\mathbf{X}_K^{T-1}] - [\mathbf{I}]_n [\mathbf{V}_{K+}^{T-1}] + [\mathbf{I}]_n [\mathbf{V}_{K-}^{T-1}] = [\mathbf{0}]_{n \times 1} \\
[\mathbf{X}_1^0] + \dots + [\mathbf{X}_K^0] \leq [\mathbf{U}^0]_{m \times 1} \\
[\mathbf{X}_1^1] + \dots + [\mathbf{X}_K^1] \leq [\mathbf{U}^1]_{m \times 1} \\
\vdots \\
[\mathbf{X}_1^{T-1}] + \dots + [\mathbf{X}_K^{T-1}] \leq [\mathbf{U}^{T-1}]_{m \times 1} \\
[\mathbf{0}]_{m \times 1} \leq [\mathbf{X}_q^t]_{m \times 1} \leq [\mathbf{U}_q^t]_{m \times 1} \quad \forall t \in \mathbf{N}, \forall q \in \mathbf{K}, \\
[\mathbf{V}_{q+}^t]_{n \times 1} \geq [\mathbf{0}]_{n \times 1}, \quad [\mathbf{V}_{q-}^t]_{n \times 1} \geq [\mathbf{0}]_{n \times 1} \quad \forall t \in \mathbf{N}, \forall q \in \mathbf{K},
\end{cases}$$

where  $[\mathbf{V}_q^t]$  and  $[\mathbf{V}_q]$  are defined as the vector of free variables at time  $t$  with respect to commodity  $q$  and the vector of supply/demand numbers with respect to commodity  $q$ .  $[\mathbf{X}_q^t]$  and  $[\mathbf{U}_q^t]$  are the  $m$ -vectors of flow and capacities in  $t$  for commodity  $q$ .  $[\mathbf{U}]$  is the  $m$ -vector of horizon capacities and  $[\mathbf{U}^t]$  is the  $m$ -vector of period capacities with respect to period  $t$ . Let  $[\mathbf{C}_q^t]$  represent the vector of arc costs at  $t \in \mathbf{N}$  for commodity  $q$ .

Matrix decomposition allows us to express the following *matrix form* as a linear program whose special structure has a great advantage to exploit efficient algorithms for its solution. If we set  $[\mathbf{V}_{q+}^T] = [\mathbf{V}_{q-}^T] = [\mathbf{0}]$  and let without loss of generality  $[\mathbf{U}^T] = [\mathbf{0}]$  while  $[\mathbf{X}_1^T], [\mathbf{X}_2^T], \dots, [\mathbf{X}_K^T]$  and  $[\mathbf{S}^0], [\mathbf{S}^1], \dots, [\mathbf{S}^T]$  denote the slack variables, then the above LP can be rewritten as:

$$\text{Min}_{\mathbf{X}_q^t, \mathbf{V}_{q+}^t, \mathbf{V}_{q-}^t, t \in \mathbf{N}} \sum (\mathbf{C}_1^t, \mathbf{C}_2^t, \dots, \mathbf{C}_K^t, \mathbf{0}, \mathbf{0}, \mathbf{0}, \dots, \mathbf{0}, \mathbf{0}, \mathbf{0}) (\mathbf{X}_1^t, \mathbf{X}_2^t, \dots, \mathbf{X}_K^t, \mathbf{V}_{1+}^t, \mathbf{V}_{1-}^t, \mathbf{V}_{2+}^t, \mathbf{V}_{2-}^t, \dots, \mathbf{V}_{K+}^t, \mathbf{V}_{K-}^t, \mathbf{S}^t)$$

$$\begin{aligned}
& \begin{bmatrix} \mathbf{I}_m & \mathbf{I}_m & \dots & \mathbf{I}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ & & & & \ddots & & & & \ddots & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{X}_1^0 \\ \mathbf{X}_2^0 \\ \vdots \\ \mathbf{X}_K^0 \\ \mathbf{V}_{1+}^0 \\ \mathbf{V}_{1-}^0 \\ \mathbf{V}_{2+}^0 \\ \mathbf{V}_{2-}^0 \\ \vdots \\ \mathbf{V}_{K+}^0 \\ \mathbf{V}_{K-}^0 \\ \mathbf{S}^0 \end{pmatrix} \\
& + \begin{bmatrix} \mathbf{I}_m & \mathbf{I}_m & \dots & \mathbf{I}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ & & & & \ddots & & & & \ddots & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{X}_1^1 \\ \mathbf{X}_2^1 \\ \vdots \\ \mathbf{X}_K^1 \\ \mathbf{V}_{1+}^1 \\ \mathbf{V}_{1-}^1 \\ \mathbf{V}_{2+}^1 \\ \mathbf{V}_{2-}^1 \\ \vdots \\ \mathbf{V}_{K+}^1 \\ \mathbf{V}_{K-}^1 \\ \mathbf{S}^1 \end{pmatrix} + \dots +
\end{aligned}$$

$$\begin{aligned}
& \begin{bmatrix} \mathbf{I}_m & \mathbf{I}_m & \dots & \mathbf{I}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ & & & & \ddots & & & & \ddots & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{X}_1^{T-1} \\ \mathbf{X}_2^{T-1} \\ \vdots \\ \mathbf{X}_K^{T-1} \\ \mathbf{V}_{1+}^{T-1} \\ \mathbf{V}_{1-}^{T-1} \\ \mathbf{V}_{2+}^{T-1} \\ \mathbf{V}_{2-}^{T-1} \\ \vdots \\ \mathbf{V}_{K+}^{T-1} \\ \mathbf{V}_{K-}^{T-1} \\ \mathbf{S}^{T-1} \end{pmatrix} + \\
& \begin{bmatrix} \mathbf{I}_m & \mathbf{I}_m & \dots & \mathbf{I}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ & & & & \ddots & & & & \ddots & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{X}_1^T \\ \mathbf{X}_2^T \\ \vdots \\ \mathbf{X}_K^T \\ \mathbf{V}_{1+}^T \\ \mathbf{V}_{1-}^T \\ \mathbf{V}_{2+}^T \\ \mathbf{V}_{2-}^T \\ \vdots \\ \mathbf{V}_{K+}^T \\ \mathbf{V}_{K-}^T \\ \mathbf{S}^T \end{pmatrix} = \begin{pmatrix} \mathbf{U} \\ \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \\ \vdots \\ \mathbf{V}_K \end{pmatrix} \\
& \begin{bmatrix} \mathbf{B}_1^0 & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_2^0 & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ & & & \ddots & & & & & \ddots & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{B}_K^0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} \\ - & - & - & - & - & - & - & - & - & - & - & - \\ \mathbf{I}_m & \mathbf{I}_m & \dots & \mathbf{I}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{I}_m \end{bmatrix} \begin{pmatrix} \mathbf{X}_1^0 \\ \mathbf{X}_2^0 \\ \vdots \\ \mathbf{X}_K^0 \\ \mathbf{V}_{1+}^0 \\ \mathbf{V}_{1-}^0 \\ \mathbf{V}_{2+}^0 \\ \mathbf{V}_{2-}^0 \\ \vdots \\ \mathbf{V}_{K+}^0 \\ \mathbf{V}_{K-}^0 \\ \mathbf{S}^0 \end{pmatrix} = \begin{pmatrix} \mathbf{0}_n \\ \mathbf{0}_n \\ \mathbf{0}_n \\ \vdots \\ \mathbf{0}_n \\ \dots \\ \mathbf{U}^0 \end{pmatrix}_{(nk+m) \times 1}
\end{aligned}$$

$$\begin{bmatrix}
\mathbf{B}_1^1 & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{B}_2^1 & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\
& & \ddots & & & & & & \ddots & & & \\
\mathbf{0} & \mathbf{0} & \dots & \mathbf{B}_K^1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} \\
- & - & - & - & - & - & - & - & - & - & - & - \\
\mathbf{I}_m & \mathbf{I}_m & \dots & \mathbf{I}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{I}_m
\end{bmatrix}
\begin{pmatrix}
\mathbf{X}_1^1 \\
\mathbf{X}_2^1 \\
\vdots \\
\mathbf{X}_K^1 \\
\mathbf{V}_{1+}^1 \\
\mathbf{V}_{1-}^1 \\
\mathbf{V}_{2+}^1 \\
\mathbf{V}_{2-}^1 \\
\vdots \\
\mathbf{V}_{K+}^1 \\
\mathbf{V}_{K-}^1 \\
\mathbf{S}^1
\end{pmatrix}
=
\begin{pmatrix}
\mathbf{0}_n \\
\mathbf{0}_n \\
\mathbf{0}_n \\
\vdots \\
\mathbf{0}_n \\
- \\
- \\
\mathbf{U}^1
\end{pmatrix}_{(nk+m) \times 1}$$

...

$$\begin{bmatrix}
\mathbf{B}_1^T & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{B}_2^T & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\
& & \ddots & & & & & & \ddots & & & \\
\mathbf{0} & \mathbf{0} & \dots & \mathbf{B}_K^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} \\
- & - & - & - & - & - & - & - & - & - & - & - \\
\mathbf{I}_m & \mathbf{I}_m & \dots & \mathbf{I}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{I}_m
\end{bmatrix}
\begin{pmatrix}
\mathbf{X}_1^T \\
\mathbf{X}_2^T \\
\vdots \\
\mathbf{X}_K^T \\
\mathbf{V}_{1+}^T \\
\mathbf{V}_{1-}^T \\
\mathbf{V}_{2+}^T \\
\mathbf{V}_{2-}^T \\
\vdots \\
\mathbf{V}_{K+}^T \\
\mathbf{V}_{K-}^T \\
\mathbf{S}^T
\end{pmatrix}
=
\begin{pmatrix}
\mathbf{0}_n \\
\mathbf{0}_n \\
\mathbf{0}_n \\
\vdots \\
\mathbf{0}_n \\
- \\
- \\
\mathbf{U}^T
\end{pmatrix}_{(nk+m) \times 1},$$

$$\begin{bmatrix}
\mathbf{X}_1^t \\
\mathbf{X}_2^t \\
\vdots \\
\mathbf{X}_K^t \\
\mathbf{V}_{1+}^t \\
\mathbf{V}_{1-}^t \\
\mathbf{V}_{2+}^t \\
\mathbf{V}_{2-}^t \\
\vdots \\
\mathbf{V}_{K+}^t \\
\mathbf{V}_{K-}^t \\
\mathbf{S}^t
\end{bmatrix}_{(mk+2nk+m) \times 1}
\leq
\begin{bmatrix}
\mathbf{U}_1^t \\
\mathbf{U}_2^t \\
\vdots \\
\mathbf{U}_K^t \\
\mathbf{V}_1 \\
\mathbf{V}_1 \\
\mathbf{V}_2 \\
\mathbf{V}_2 \\
\vdots \\
\mathbf{V}_K \\
\mathbf{V}_K \\
\mathbf{U}^t
\end{bmatrix}_{(mk+2nk+m) \times 1}
\quad \forall t \in \mathbf{N} \cup \{T\},$$

which becomes as the following condensed form:

$$\begin{aligned}
& \underset{\mathbf{X}_q^t, \mathbf{V}_{q+}^t, \mathbf{V}_{q-}^t}{\text{Min}} \sum_{t \in \mathbf{N}} [\mathbf{C}_1^t \quad \mathbf{C}_2^t \quad \dots \quad \mathbf{C}_K^t] [\mathbf{X}_1^t \quad \mathbf{X}_2^t \quad \dots \quad \mathbf{X}_K^t]^{trans}, \\
& \sum_{t \in \mathbf{N}} ([\mathbf{X}_1^t] + \dots + [\mathbf{X}_K^t]) \leq [\mathbf{U}]_{m \times 1}, \\
& \sum_{t \in \mathbf{N}} ([\mathbf{V}_{q+}^t] - [\mathbf{V}_{q-}^t]) = [\mathbf{V}_q]_{n \times 1} \quad \forall q \in \mathbf{K}, \\
& [\mathbf{B}_q^t]_{n \times m} [\mathbf{X}_q^t] - [\mathbf{I}]_n [\mathbf{V}_{q+}^t] + [\mathbf{I}]_n [\mathbf{V}_{q-}^t] = [\mathbf{0}]_{n \times 1} \quad \forall t \in \mathbf{N}, \forall q \in \mathbf{K}, \\
& [\mathbf{X}_1^t] + \dots + [\mathbf{X}_K^t] \leq [\mathbf{U}^t]_{m \times 1} \quad \forall t \in \mathbf{N}, \\
& [\mathbf{0}]_{m \times 1} \leq [\mathbf{X}_q^t]_{m \times 1} \leq [\mathbf{U}_q^t]_{m \times 1} \quad \forall t \in \mathbf{N}, \forall q \in \mathbf{K}, \\
& [\mathbf{V}_{q+}^t]_{n \times 1} \geq [\mathbf{0}]_{n \times 1}, \quad [\mathbf{V}_{q-}^t]_{n \times 1} \geq [\mathbf{0}]_{n \times 1} \quad \forall t \in \mathbf{N}, \forall q \in \mathbf{K}.
\end{aligned}$$

Our modeling procedure reveals that we can formulate any MCDF problem on a SMMN as a problem which possesses the *block angular* structure. To show this, define

$$[\mathbf{C}^t] := (\mathbf{C}_1^t \quad \mathbf{C}_2^t \quad \dots \quad \mathbf{C}_K^t \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{0}) \quad t \in \{0, 1, \dots, T\},$$

$$[\mathbf{Y}^t] := (\mathbf{X}_1^t \quad \mathbf{X}_2^t \quad \dots \quad \mathbf{X}_K^t \quad \mathbf{V}_{1+}^t \quad \mathbf{V}_{1-}^t \quad \mathbf{V}_{2+}^t \quad \mathbf{V}_{2-}^t \quad \dots \quad \mathbf{V}_{K+}^t \quad \mathbf{V}_{K-}^t \quad \mathbf{S}^t)^{trans} \quad t \in \{0, 1, \dots, T\},$$

$$[\mathbf{W}^t] := (\mathbf{U}_1^t \quad \mathbf{U}_2^t \quad \dots \quad \mathbf{U}_K^t \quad \mathbf{V}_1 \quad \mathbf{V}_1 \quad \mathbf{V}_2 \quad \mathbf{V}_2 \quad \dots \quad \mathbf{V}_K \quad \mathbf{V}_K \quad \mathbf{U}^t)^{trans} \quad t \in \{0, 1, \dots, T\},$$

$$[\overline{\mathbf{W}}^t] := (\mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad \vdots \quad \mathbf{U}^t)^{trans} \quad t \in \{0, 1, \dots, T\},$$

$$[\mathbf{A}^t] := \begin{bmatrix} \mathbf{B}_1^t & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_2^t & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ & & \ddots & & & & & & \ddots & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{B}_K^t & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{I} & \mathbf{I} & \mathbf{0} \\ - & - & - & - & - & - & - & - & - & - & - & - \\ \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \quad t \in \{0, 1, \dots, T\},$$

$$[\mathbf{M}] := \begin{bmatrix} \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & -\mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ & & & & \ddots & & & & \ddots & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & -\mathbf{I} & \mathbf{0} \end{bmatrix}. \quad (\text{master matrix})$$

According to the newly introduced representation and applying our notations, the LP formulation of the problem becomes

$$\begin{aligned}
\text{Min} \quad & \sum_{t \in N} [C^t][Y^t] \\
& [M][Y^0] + [M][Y^1] + \dots + [M][Y^{T-1}] + [M][Y^T] = (U \quad V_1 \quad \dots \quad V_K)^{trans}, \\
& [A^0][Y^0] = [\bar{W}^0], \\
& [A^1][Y^1] = [\bar{W}^1], \\
& [A^2][Y^2] = [\bar{W}^2], \\
& \vdots \\
& [A^T][Y^T] = [\bar{W}^T], \\
& [0] \leq [Y^t] \leq [W^t] \quad t \in \{0, 1, \dots, T\}.
\end{aligned}$$

Block diagonal structure is desirable to speed up the solution process for a sparse linear programming problem. We may also conveniently exploit the decomposition procedures, slice modelling systems, and generalized upper bounding (GUB) techniques to solve such problems efficiently. However, Dantzig-Wolfe decomposition (*delayed column generation method*) is often the best method of choice when dealing with large scale sparse problems arising from repeated components, especially in terms of storage requirements and good-quality suboptimal solutions. DW decomposition will not rival mainstream techniques as an optimization method for all LP problems, but it has some niche areas of application: certain large scale classes of primal block angular structured problems, and in particular where the context demands rapid results using parallel optimization, or near optimal solutions with a guaranteed quality [83].

To better illustrate, let's assume an MMN problem with  $T$  subproblems, each with  $n$  conservation constraints and with a master condition of  $n_0$  constraints (all in standard form). The storage requirement of the revised simplex method for the original problem will be  $O((n_0 + Tn)^2)$ , which is the size of the revised simplex tableau. In contrast, the storage requirements of the decomposition method for the same problem turns out to be  $O((n_0 + T)^2)$  for the tableau of the master problem, and  $T \times O(n^2)$  for the revised simplex tableau of subproblems (that are easy to solve). Moreover, applying decomposition on a MMN problem maintains only one tableau stored in the main memory at any time. For instance, let  $T = 1000$  and  $n = n_0 \gg T$ . In this case, the main memory requirement of the decomposition method will be 1,000,000 times smaller than those of the revised simplex



method. Therefore, while the memory is a key bottleneck in handling very large LPs, like MMN problems, the decomposition approach dramatically enlarges the range of problems that can be solved practically [9][15][83][86].

Interestingly enough, in our approach, the master constraint has the same matrix for any set of variables  $(\mathbf{X}_1^t \ \mathbf{X}_2^t \ \dots \ \mathbf{X}_K^t \ \mathbf{V}_{1+}^t \ \mathbf{V}_{1-}^t \ \mathbf{V}_{2+}^t \ \mathbf{V}_{2-}^t \ \dots \ \mathbf{V}_{K+}^t \ \mathbf{V}_{K-}^t \ \mathbf{S}^t)^{trans}$ . In general, it is not necessary for either set of constraints to have a special structure, but when available, it helps speed up the solution method. Meanwhile, although the modeling language used in this study may resemble the *slice modelling systems*, there are a couple of important discrepancies between slice modeling and our polyhedral-based approach in this section (See Section 2).

In order to use the DW decomposition on our problem, the constraint matrix should be exploited by splitting the original problem into smaller *subproblems* and a connecting constraint, *master constraint* (the one containing the master matrix  $\mathbf{M}$ ). The structure of the time-varying block-angular system admits a natural decomposition into a set of  $T + 1$  independent well-structured smaller parts instead of solving the original problem whose size and complexity are beyond what can be solved within a reasonable amount of time, and then adjust the solution to take into account the interconnections. Thus, we may apply the block diagonal decomposition techniques to solve the foregoing problem to achieve the desired effect. Let us consider an application of a decomposition algorithm to the problem: define  $T + 1$  polyhedral sets  $\chi^t$  for each  $t \in \{0, 1, \dots, T\}$  as:

$$\chi^t := \left\{ [\mathbf{Y}^t] : [\mathbf{A}^t][\mathbf{Y}^t] = [\overline{\mathbf{W}}^t], \quad [\mathbf{0}] \leq [\mathbf{Y}^t] \leq [\mathbf{W}^t] \right\}.$$

Considering Minkowski's Representation Theorem, any  $[\mathbf{Y}^t] \in \chi^t$  can be expressed as a convex combination of a finite number of extreme points of  $\chi^t$  as

$$\begin{aligned} (\mathbf{X}_1^t \ \mathbf{X}_2^t \ \dots \ \mathbf{X}_K^t \ \mathbf{V}_{1+}^t \ \mathbf{V}_{1-}^t \ \mathbf{V}_{2+}^t \ \mathbf{V}_{2-}^t \ \dots \ \mathbf{V}_{K+}^t \ \mathbf{V}_{K-}^t \ \mathbf{S}^t) \in \chi^t &\Leftrightarrow \\ [\mathbf{Y}^t] &= \sum_{i=1}^{k^t} \alpha_i^t [\mathbf{y}_i^t] + \sum_{j=1}^{l^t} \mu_j^t [\mathbf{d}_j^t], \\ \sum_{i=1}^{k^t} \alpha_i^t &= 1, \quad \alpha_i^t \geq 0 \text{ for } i = 1, \dots, k^t, \text{ and } \mu_j^t \geq 0 \text{ for } j = 1, \dots, l^t, \end{aligned}$$

where  $[y_1^t], [y_2^t], \dots, [y_{k^t}^t]$  and  $[d_1^t], [d_2^t], \dots, [d_{l^t}^t]$  are extreme points and extreme directions (if any) of polyhedral  $\chi^t$ . The original problem can be reformulated as the *master problem* under Minkowski's mapping as follows.

$$\begin{aligned} & \mathbf{Min}_{\alpha_i^t, \mu_j^t} \sum_{t \in N} \sum_{i=1}^{k^t} \alpha_i^t [C^t][y_i^t] + \sum_{t \in N} \sum_{j=1}^{l^t} \mu_j^t [C^t][d_j^t], \\ & [\mathbf{M}] \sum_{i=0}^{k^0} \alpha_i^0 [y_i^0] + [\mathbf{M}] \sum_{j=1}^{l^1} \mu_j^0 [d_j^0] + \dots + [\mathbf{M}] \sum_{i=1}^{k^T} \alpha_i^T [x_i^T] + [\mathbf{M}] \sum_{j=1}^{l^T} \mu_j^T [d_j^T] = (\mathbf{U} \quad \mathbf{V}_1 \quad \mathbf{V}_2 \quad \dots \quad \mathbf{V}_K)^{trans} \\ & \sum_{i=1}^{k^t} \alpha_i^t = 1 \quad \forall t \in N \cup \{T\}, \\ & \alpha_i^t \geq 0 \quad \forall t \in N \cup \{T\} \quad i = 1, \dots, k^t, \\ & \mu_j^t \geq 0 \quad \forall t \in N \cup \{T\} \quad j = 1, \dots, l^t. \end{aligned}$$

Due to having a huge number of extreme points for each polyhedron, enumerating all the *extreme points*, and solving this problem directly seems impossible. Rather, we should find a reasonable approach without enumerating all the extreme points [9]. This is where we suggest the use of decomposition techniques, especially due to the special structure of our problem that greatly intensifies the efficiency of the decomposition methods. Next, we develop the most general form of the min-cost flow problem formulation for a SMMN as

(*Condensed Master Problem*)

$$\begin{aligned} & \mathbf{Min}_{\alpha_i^t, \mu_j^t} \sum_{t \in N \cup \{T\}} [C^t] \left( \sum_{i=1}^{k^t} \alpha_i^t [y_i^t] + \sum_{j=1}^{l^t} \mu_j^t [d_j^t] \right), \\ & [\mathbf{M}] \sum_{t \in N \cup \{T\}} \left( \sum_{i=1}^{k^t} \alpha_i^t [y_i^t] + \sum_{j=1}^{l^t} \mu_j^t [d_j^t] \right) = (\mathbf{U} \quad \mathbf{V}_1 \quad \mathbf{V}_2 \quad \dots \quad \mathbf{V}_K)^{trans}, \\ & \sum_{i=1}^{k^t} \alpha_i^t = 1 \quad \forall t \in N \cup \{T\}, \\ & \alpha_i^t \geq 0 \quad \forall t \in N \cup \{T\} \quad i = 1, \dots, k^t, \\ & \mu_j^t \geq 0 \quad \forall t \in N \cup \{T\} \quad j = 1, \dots, l^t. \end{aligned}$$

The formulation of the SMMN problem shows a much simpler constraint structure than the usual matrix form. It possesses only  $m+Kn+T+1$  constraints rather than  $(T+2)(m+Kn)$  in the earlier formulation. Problems of this type are well amenable by many decomposition algorithms and column generation methods. As a result, the computational advantage of the algorithm depends on the efficiency of the decomposition methods. We propose DW decomposition (or Benders algorithm for the dual) and so, the same analysis is applied for this case.

When the problem has many thousands of rows and unsolvable in a reasonable amount of time, however, our approach suggests a method to convert the large-scale (high dimensional problem) into one or more appropriately coordinated smaller sparse problems of manageable sizes. Having the *restricted master problem* (the latter model with only a small number of variables), a general Dantzig-Wolfe decomposition algorithm for a MMNF problem can be summarized as that in the following page.

**Remark 3.1** Considering Minkowski mapping and feasibility of the problem the optimal solution (corresponding to the optimal basis) of SMMNF from the condensed master problem will determine a set of original variables of form  $(\mathbf{X}'_1 \ \mathbf{X}'_2 \ \dots \ \mathbf{X}'_K \ \mathbf{V}'_{1+} \ \mathbf{V}'_{1-} \ \mathbf{V}'_{2+} \ \mathbf{V}'_{2-} \ \dots \ \mathbf{V}'_{K+} \ \mathbf{V}'_{K-} \ \mathbf{S}')$  for each time step with respect to each commodity conveying a positive flow. Any obtained optimal basis will detect one arc set for every time step  $t$  and for each commodity  $q$  that transports a positive amount of flow. Moreover, the values of  $v'_{iq}$  for each  $i$ ,  $t$ , and  $q$  will be determined at any basis. Moreover, it immediately follows that the optimal arc sets for every time step and for any commodity are not necessarily the same.

*MMNF DANTZIG-WOLFE ALGORITHM*

*{Initialization}*

Choose initial subsets of proposals

set  $kk(k)$  'current proposal';

$kk('proposal1') = \text{yes};$

loop((q,t),

solve subproblem, check feasibility

$c(i,j) = \text{cost}(q,t,i,j);$

$u(i,j) = \text{capacity}(q,t,i,j);$

$u(i,j) = \text{capacity}(t,i,j);$

.

.

.

$\mu_1(i,j) = 0;$

$\mu_2^{(t,q)} = 0.$

*while (true) do*

solve restricted master;

solve subproblems;

until no more proposals.

*{Initialization}*

Choose initial subsets of variables

While true do

*{Master problem}*

Solve the restricted master problem

$\mu_1 :=$  duals of master constraints

$\mu_2^{(t)} :=$  duals of the  $t^{\text{th}}$  convexity constraints

*{Subproblems}*

for  $t = 0, \dots, T$  do

Plug  $\mu_1$  and  $\mu_2^{(t)}$  into subproblem  $t$

Solve sub-problem  $t$

if  $(\text{reduced cost})_t = \min (c^t - \mu_1 M)Y_t - \mu_2^{(t)} < 0$  then

Add proposal  $Y_t$  to the restricted master problem

end if

end for

if no proposals generated then

Stop: optimality

end if

end while

For the initialization step of DW algorithm, we first solve each subproblem. If any of the subproblems is infeasible, the original MMN problem is clearly infeasible. Otherwise, we use the optimal values  $Y_t$  (or the unbounded rays) to generate an initial set of proposals/columns. The initial proposals may violate the master constraints. We formulate a Phase I problem by introducing artificial variables and minimizing those (e.g. [9]). The reduced costs of a Phase I problem are slightly different from the Phase II problem.

The Minkowski mapping proves that decomposition solves SMMNF problem by generating coefficient data as needed. Since the master problem is an LP, the decomposition algorithm inherits finite convergence from the (revised) simplex method. Recall that the simplex method solves LP problems in a finite number of steps, provided that a cycling prevention rule is used. For decomposition, the subproblem calculation ensures that the vectors introduced into the basis have positive reduced cost. Consequently, from the linear programming theory, the master problem is solved in a finite number of steps; the procedure thus determines an optimal solution by solving the condensed master problem and subproblem alternately a finite number of times [9][15].

### 3.3 A Solution Approach for MM Networks with Storage

In certain practical problems, the intermediate storage policy is another important issue that has to be considered in the models. It may be necessary that the  $q$ -flow (flow of commodity  $q$ ) waits at some nodes until it can continue on an arc, as it appears in many applications such as batch process scheduling, traffic routing, evacuation planning, energy transmission, inventory, and telecommunications [3][35]. This affects both the problem's complexity and optimal solution. When there are no storage equipments, Non-intermediate Storage (NIS) policy is assumed. It is also common in the industry for a process to have different storage policies for different intermediates that is called Mixed Intermediate Storage (MIS) policy.

In our model, we let the  $q$ -flow be stored at some (or all) intermediate nodes (or demand nodes) for only one time period with finite (or infinite) storage capacity depending on a pre-defined capacity function for each  $t$ ,  $q$ , and  $i$ . Note that, the dependency of capacity functions on time (and commodity) allows us to have different storage policies

in different time periods with respect to different products, but the time lag is still one. In general, the staircase structure may have longer time lags which will be discussed in the upcoming sections. We can allow flow storage in MMN by introducing loops in those nodes in which storage is allowed.

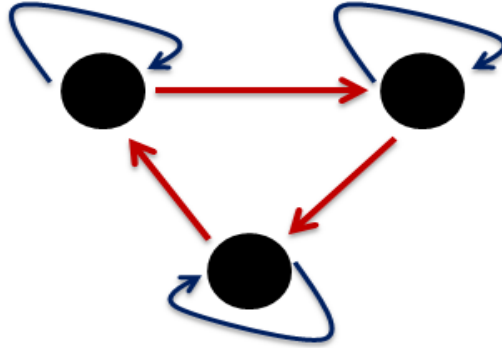


Figure 3.2 A typical MMN with flow storage at nodes

This leads to a slightly different notion of flow conservation. If we let the set of vertices  $V$  be divided into three subsets  $VS_Q, VI_Q, VD_Q$  comprising of source, intermediate, and sink nodes with respect to each  $q$ , respectively, we can state the flow conservation constraints as following. In this case,  $x(t): A \rightarrow \mathbb{R}^+$  is a *dynamic feasible flow* if it satisfies constraints (3.2)-(3.6) and (3.24) as well

$$\sum_j \int_0^\theta x_{ijq}(t) dt - \sum_j \int_0^\theta x_{jiq}(t) dt \leq 0 \quad \forall i \in V \setminus VS_Q, \forall q \in K, \forall \theta \in ]0, T[ \quad (3.24)$$

As flow travels through the distribution network, we may allow limited (or unlimited) flow storage at nodes, but prohibit any deficit by constraint (3.24). As before, all  $q$ -demands must be met, flow must not remain in the network after time  $T$ , and each source/sink must not exceed its forecasted supply/demand.

The discrete-time model is considered, in which all times are integral and bounded by an integer horizon. A *discrete dynamic flow* in  $G$ , which satisfies the following constraints, is said to be *feasible*. Such a flow is a non-negative function  $x: \{A \cup V\} \times N \times K \rightarrow \mathbb{R}^+$  satisfying (3.25)-(3.30).

$$\sum_j \sum_{t=0}^{T-1} x_{ijq}(t) - \sum_j \sum_{t=0}^{T-1} x_{jiq}(t) = v_{iq} \quad \forall i \in V, \forall q \in K, \quad (3.25)$$

$$\sum_j x_{ijq}(t) - \sum_j x_{jiq}(t) \leq 0 \quad \forall i \in V \setminus VS, \forall t \in N, \forall q \in K, \quad (3.26)$$

$$\sum_{q \in K} \sum_{t \in N} x_{ijq}(t) \leq u_{ij} \quad \forall (i, j) \in A, \quad (3.27)$$

$$\sum_{q \in K} x_{ijq}(t) \leq u_{ij}(t) \quad \forall (i, j) \in A, \forall t \in N, \quad (3.28)$$

$$0 \leq x_{ijq}(t) \leq u_{ijq}(t) \quad \forall (i, j) \in A, \forall t \in N, \forall q \in K, \quad (3.29)$$

$$x_{ijq}(t) = 0 \quad \forall (i, j) \in A, \forall t \notin N, \forall q \in K. \quad (3.30)$$

Let  $c_{iiq}(t)$  be the storage cost in node  $i$  at period  $t$  with respect to  $q$ . Thus, the total cost of a discrete dynamic flow  $x$  is defined by

$$\sum_{q \in K} \sum_{t \in N} \sum_{(i,j) \in A} c_{ijq}(t) x_{ijq}(t) + \sum_{q \in K} \sum_{t \in N} \sum_{i \in V} c_{iiq}(t) \left( \sum_j x_{jiq}(t) - \sum_j x_{ijq}(t) \right). \quad (3.31)$$

Having introduced the unrestricted variables  $v_{iq}(t)$ , we may reformulate the problem as

$$\mathbf{Min}_{x_{ijq}(t), x_{iiq}(t), v_{iq}(t)} \sum_{q \in K} \sum_{t \in N} \sum_{(i,j) \in A} c_{ijq}(t) x_{ijq}(t) + \sum_{q \in K} \sum_{t \in N} \sum_{i \in V} c_{iiq}(t) x_{iiq}(t) + \sum_{q \in K} \sum_{t \in N} \sum_{i \in V} c_{iq}(t) v_{iq}(t), \quad (3.32)$$

$$\sum_j x_{ijq}(t) - \sum_j x_{jiq}(t) + [x_{iiq}(t) - x_{iiq}(t-1)] - v_{iq}(t) = 0 \quad \forall i \in V, \forall t \in N, \forall q \in K, \quad (3.33)$$

$$\sum_{t \in N} v_{iq}(t) = v_{iq} \quad \forall i \in V, \forall q \in K, \quad (3.34)$$

$$\sum_{q \in K} \sum_{t \in N} x_{ijq}(t) \leq u_{ij} \quad \forall (i, j) \in A, \quad (3.35)$$

$$\sum_{q \in K} x_{ijq}(t) \leq u_{ij}(t) \quad \forall (i, j) \in A, \forall t \in N, \quad (3.36)$$

$$0 \leq x_{iiq}(t) \leq u_{iiq}(t) \quad \forall i \in V \setminus VSQ, \forall t \in N, \forall q \in K, \quad (3.37)$$

$$0 \leq x_{ijq}(t) \leq u_{ijq}(t) \quad \forall (i, j) \in A, \forall t \in N, \forall q \in K, \quad (3.38)$$

$$x_{ijq}(t) = 0 \quad \forall (i, j) \in A, \forall t \notin N, \forall q \in K, \quad (3.39)$$

$$x_{iiq}(t) = 0 \quad \forall i \in V, \forall q \in K, t = -1, T-1, \quad (3.40)$$

where  $v_{iq}(t)$  is a free variable which defines the difference between  $q$ -outflow and  $q$ -inflow at node  $i$  at time period  $t$ . Needless to say,  $v_{iq}(t)$  differs from storage decision variables, and they are determined optimally by the algorithm.  $x_{iiq}(t)$  and  $c_{iiq}(t)$  are the amount and cost of stored flow at node  $i$  in period  $t$  of product  $q$ . We set  $c_{iiq}(t)=0$  for each  $i, t$ , and  $q$ . Clearly, there is no need to have flow storage in period  $T-1$ , and this is ensured by (3.40). We prove that this model possesses a property which enables us to reduce the MCDF problem to a problem with special structure. By setting  $v_{iq}(t) = v_{iq+}^t - v_{iq-}^t$ , our problem is reduced to the following *matrix form* as an LP, whose special structure enables efficient solution of the model.

$$\begin{aligned}
& \underset{[\mathbf{X}_q^t], [\mathbf{X}_{sq}^t], [\mathbf{V}_{q+}^t], [\mathbf{V}_{q-}^t]}{\text{Min}} \quad \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} [\mathbf{C}_q^t][\mathbf{X}_q^t] + \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} [\mathbf{C}_{sq}^t][\mathbf{X}_{sq}^t], \\
& ([\mathbf{X}_1^0] + \dots + [\mathbf{X}_K^0]) + ([\mathbf{X}_1^1] + \dots + [\mathbf{X}_K^1]) + \dots + ([\mathbf{X}_1^{T-1}] + \dots + [\mathbf{X}_K^{T-1}]) \leq [\mathbf{U}]_{m \times 1} \\
& \left\{ \begin{aligned} & ([\mathbf{V}_{1+}^0] - [\mathbf{V}_{1-}^0]) + ([\mathbf{V}_{1+}^1] - [\mathbf{V}_{1-}^1]) + \dots + ([\mathbf{V}_{1+}^{T-1}] - [\mathbf{V}_{1-}^{T-1}]) = [\mathbf{V}_1]_{n \times 1} \\ & ([\mathbf{V}_{2+}^0] - [\mathbf{V}_{2-}^0]) + ([\mathbf{V}_{2+}^1] - [\mathbf{V}_{2-}^1]) + \dots + ([\mathbf{V}_{2+}^{T-1}] - [\mathbf{V}_{2-}^{T-1}]) = [\mathbf{V}_2]_{n \times 1} \\ & \quad \vdots \\ & ([\mathbf{V}_{K+}^0] - [\mathbf{V}_{K-}^0]) + ([\mathbf{V}_{K+}^1] - [\mathbf{V}_{K-}^1]) + \dots + ([\mathbf{V}_{K+}^{T-1}] - [\mathbf{V}_{K-}^{T-1}]) = [\mathbf{V}_K]_{n \times 1} \end{aligned} \right. \\
& \left\{ \begin{aligned} & [\mathbf{A}_1^0]_{n \times m} [\mathbf{X}_1^0] + [\mathbf{I}]_{n \times n} [\mathbf{X}_{s1}^0]_{n \times 1} + [-\mathbf{I}]_{n \times n} [\mathbf{X}_{s1}^{-1}]_{n \times 1} - ([\mathbf{V}_{1+}^0] - [\mathbf{V}_{1-}^0]) = [\mathbf{0}]_{n \times 1} \\ & [\mathbf{A}_2^0]_{n \times m} [\mathbf{X}_2^0] + [\mathbf{I}]_{n \times n} [\mathbf{X}_{s2}^0]_{n \times 1} + [-\mathbf{I}]_{n \times n} [\mathbf{X}_{s2}^{-1}]_{n \times 1} - ([\mathbf{V}_{2+}^0] - [\mathbf{V}_{2-}^0]) = [\mathbf{0}]_{n \times 1} \\ & \quad \vdots \\ & [\mathbf{A}_K^0]_{n \times m} [\mathbf{X}_K^0] + [\mathbf{I}]_{n \times n} [\mathbf{X}_{sK}^0]_{n \times 1} + [-\mathbf{I}]_{n \times n} [\mathbf{X}_{sK}^{-1}]_{n \times 1} - ([\mathbf{V}_{K+}^0] - [\mathbf{V}_{K-}^0]) = [\mathbf{0}]_{n \times 1} \end{aligned} \right. \\
& \left\{ \begin{aligned} & [\mathbf{A}_1^1]_{n \times m} [\mathbf{X}_1^1] + [\mathbf{I}]_{n \times n} [\mathbf{X}_{s1}^1]_{n \times 1} + [-\mathbf{I}]_{n \times n} [\mathbf{X}_{s1}^0]_{n \times 1} - ([\mathbf{V}_{1+}^1] - [\mathbf{V}_{1-}^1]) = [\mathbf{0}]_{n \times 1} \\ & [\mathbf{A}_2^1]_{n \times m} [\mathbf{X}_2^1] + [\mathbf{I}]_{n \times n} [\mathbf{X}_{s2}^1]_{n \times 1} + [-\mathbf{I}]_{n \times n} [\mathbf{X}_{s2}^0]_{n \times 1} - ([\mathbf{V}_{2+}^1] - [\mathbf{V}_{2-}^1]) = [\mathbf{0}]_{n \times 1} \\ & \quad \vdots \\ & [\mathbf{A}_K^1]_{n \times m} [\mathbf{X}_K^1] + [\mathbf{I}]_{n \times n} [\mathbf{X}_{sK}^1]_{n \times 1} + [-\mathbf{I}]_{n \times n} [\mathbf{X}_{sK}^0]_{n \times 1} - ([\mathbf{V}_{K+}^1] - [\mathbf{V}_{K-}^1]) = [\mathbf{0}]_{n \times 1} \end{aligned} \right. \\
& \quad \vdots \\
& \left\{ \begin{aligned} & [\mathbf{A}_1^{T-1}]_{n \times m} [\mathbf{X}_1^{T-1}] + [\mathbf{I}]_{n \times n} [\mathbf{X}_{s1}^{T-1}]_{n \times 1} + [-\mathbf{I}]_{n \times n} [\mathbf{X}_{s1}^{T-2}]_{n \times 1} - ([\mathbf{V}_{1+}^{T-1}] - [\mathbf{V}_{1-}^{T-1}]) = [\mathbf{0}]_{n \times 1} \\ & [\mathbf{A}_2^{T-1}]_{n \times m} [\mathbf{X}_2^{T-1}] + [\mathbf{I}]_{n \times n} [\mathbf{X}_{s2}^{T-1}]_{n \times 1} + [-\mathbf{I}]_{n \times n} [\mathbf{X}_{s2}^{T-2}]_{n \times 1} - ([\mathbf{V}_{2+}^{T-1}] - [\mathbf{V}_{2-}^{T-1}]) = [\mathbf{0}]_{n \times 1} \\ & \quad \vdots \\ & [\mathbf{A}_K^{T-1}]_{n \times m} [\mathbf{X}_K^{T-1}] + [\mathbf{I}]_{n \times n} [\mathbf{X}_{sK}^{T-1}]_{n \times 1} + [-\mathbf{I}]_{n \times n} [\mathbf{X}_{sK}^{T-2}]_{n \times 1} - ([\mathbf{V}_{K+}^{T-1}] - [\mathbf{V}_{K-}^{T-1}]) = [\mathbf{0}]_{n \times 1} \end{aligned} \right.
\end{aligned}$$



$$\begin{cases} [\mathbf{X}_1^0] + \dots + [\mathbf{X}_K^0] \leq [\mathbf{U}^0]_{m \times 1} \\ [\mathbf{X}_1^1] + \dots + [\mathbf{X}_K^1] \leq [\mathbf{U}^1]_{m \times 1} \\ \vdots \\ [\mathbf{X}_1^{T-1}] + \dots + [\mathbf{X}_K^{T-1}] \leq [\mathbf{U}^{T-1}]_{m \times 1} \end{cases}$$

$$[\mathbf{0}]_{m \times 1} \leq [\mathbf{X}_q^t]_{m \times 1} \leq [\mathbf{U}_q^t]_{m \times 1} \quad \forall t \in \mathbf{N}, \forall q \in \mathbf{K},$$

$$[\mathbf{0}]_{n \times 1} \leq [\mathbf{X}_{sq}^t]_{n \times 1} \leq [\mathbf{U}_{sq}^t]_{n \times 1} \quad \forall t \in \mathbf{N}, \forall q \in \mathbf{K},$$

$$[\mathbf{V}_{q+}^t]_{n \times 1} \geq [\mathbf{0}]_{n \times 1}, \quad [\mathbf{V}_{q-}^t]_{n \times 1} \geq [\mathbf{0}]_{n \times 1} \quad \forall t \in \mathbf{N}, \forall q \in \mathbf{K}.$$

Where  $[\mathbf{X}_q^t] = \{x_{ijq}(t)\} = \{x_{ijq}^t\}$  and  $[\mathbf{X}_{sq}^t] = \{x_{iiq}(t)\} = \{x_{iiq}^t\}$  are the vectors of flow and storage at time period  $t$ ,  $[\mathbf{U}_q^t] = \{u_{ijq}(t)\} = \{u_{ijq}^t\}$  and  $[\mathbf{U}_{sq}^t] = \{u_{iiq}(t)\} = \{u_{iiq}^t\}$  are the vectors of flow capacities and storage at  $t \in \mathbf{N}$ , respectively,  $[\mathbf{U}]$ ,  $[\mathbf{U}^t]$ ,  $[\mathbf{C}_q^t]$ , and  $[\mathbf{V}_q^t]$  are defined as before.  $[\mathbf{C}_{sq}^t] = \{c_{iiq}(t)\} = \{c_{iiq}^t\}$  is the vector of pre-defined storage costs.  $[\mathbf{A}]_{n \times m}$  is the node-arc incidence matrix of the underlying network.

In order to obtain a standard LP problem we decompose  $[\mathbf{V}_q^t]$  to non-negative vectors  $[\mathbf{V}_{q+}^t]$  and  $[\mathbf{V}_{q-}^t]$ .  $[\mathbf{V}_q] = \{v_{iq}\}$  is the vector of pre-defined supply/demand numbers of nodes with respect to commodity  $q$ . Let  $[\mathbf{X}_{sq}^{-1}] = \mathbf{0}$  be the vector of initial storage and  $[\mathbf{S}]$  be the vector of slack variables. Without any loss of generality, let  $[\mathbf{S}] := [\mathbf{X}_1^T] \geq [\mathbf{0}]$ ,  $[\mathbf{X}_2^T] = [\mathbf{0}], \dots, [\mathbf{X}_K^T] = [\mathbf{0}]$ ,  $[\mathbf{V}_{q+}^T] = [\mathbf{V}_{q-}^T] = [\mathbf{0}]$ , and  $[\mathbf{U}^T] = [\mathbf{0}]$  (since we do not need any flow in the last period). Then, we can convert the model into the following form by manipulating and introducing some matrices.

$$\text{Min} \sum_{t \in \mathbb{N}} (\mathbf{C}_1^t, \dots, \mathbf{C}_K^t, \mathbf{C}_{s1}^t, \dots, \mathbf{C}_{sK}^t, \mathbf{0}, \mathbf{0}, \mathbf{0}, \dots, \mathbf{0}, \mathbf{0}, \mathbf{0}) (\mathbf{X}_1^t, \dots, \mathbf{X}_K^t, \mathbf{X}_{s1}^t, \dots, \mathbf{X}_{sK}^t, \mathbf{V}_{1+}^t, \mathbf{V}_{1-}^t, \mathbf{V}_{2+}^t, \mathbf{V}_{2-}^t, \dots, \mathbf{V}_{K+}^t, \mathbf{V}_{K-}^t, \mathbf{S}^t)$$

$$\begin{bmatrix}
 \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
 \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\
 \vdots & & \ddots & & & & & & & & & & \ddots & & & \vdots \\
 \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & -\mathbf{I} & \mathbf{0} \\
 - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
 \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0}
 \end{bmatrix}
 \begin{pmatrix}
 \mathbf{X}_1^0 \\
 \mathbf{X}_2^0 \\
 \vdots \\
 \mathbf{X}_K^0 \\
 \mathbf{X}_{s1}^0 \\
 \mathbf{X}_{s2}^0 \\
 \vdots \\
 \mathbf{X}_{sK}^0 \\
 \mathbf{V}_{1+}^0 \\
 \mathbf{V}_{1-}^0 \\
 \mathbf{V}_{2+}^0 \\
 \mathbf{V}_{2-}^0 \\
 \vdots \\
 \mathbf{V}_{K+}^0 \\
 \mathbf{V}_{K-}^0 \\
 \mathbf{S}^0
 \end{pmatrix}
 +
 \begin{bmatrix}
 \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
 \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\
 \vdots & & \ddots & & & & & & & & & & \ddots & & & \vdots \\
 \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & -\mathbf{I} & \mathbf{0} \\
 - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
 \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0}
 \end{bmatrix}
 \begin{pmatrix}
 \mathbf{X}_1^1 \\
 \mathbf{X}_2^1 \\
 \vdots \\
 \mathbf{X}_K^1 \\
 \mathbf{X}_{s1}^1 \\
 \mathbf{X}_{s2}^1 \\
 \vdots \\
 \mathbf{X}_{sK}^1 \\
 \mathbf{V}_{1+}^1 \\
 \mathbf{V}_{1-}^1 \\
 \mathbf{V}_{2+}^1 \\
 \mathbf{V}_{2-}^1 \\
 \vdots \\
 \mathbf{V}_{K+}^1 \\
 \mathbf{V}_{K-}^1 \\
 \mathbf{S}^1
 \end{pmatrix}
 + \dots$$

$$\begin{aligned}
& \dots + \begin{bmatrix} \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & & \ddots & & & & & & & & & & \ddots & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & -\mathbf{I} & \mathbf{0} \\ - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{pmatrix} \mathbf{X}_1^{T-1} \\ \mathbf{X}_2^{T-1} \\ \vdots \\ \mathbf{X}_K^{T-1} \\ \mathbf{X}_{s1}^{T-1} \\ \mathbf{X}_{s2}^{T-1} \\ \vdots \\ \mathbf{X}_{sK}^{T-1} \\ \mathbf{V}_{1+}^{T-1} \\ \mathbf{V}_{1-}^{T-1} \\ \mathbf{V}_{2+}^{T-1} \\ \mathbf{V}_{2-}^{T-1} \\ \vdots \\ \mathbf{V}_{K+}^{T-1} \\ \mathbf{V}_{K-}^{T-1} \\ \mathbf{S}^{T-1} \end{pmatrix} + \\
& \begin{bmatrix} \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & & \ddots & & & & & & & & & & \ddots & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & -\mathbf{I} & \mathbf{0} \\ - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} = \begin{pmatrix} \mathbf{X}_1^T \\ \mathbf{X}_2^T \\ \vdots \\ \mathbf{X}_K^T \\ \mathbf{X}_{s1}^T \\ \mathbf{X}_{s2}^T \\ \vdots \\ \mathbf{X}_{sK}^T \\ \mathbf{V}_{1+}^T \\ \mathbf{V}_{1-}^T \\ \mathbf{V}_{2+}^T \\ \mathbf{V}_{2-}^T \\ \vdots \\ \mathbf{V}_{K+}^T \\ \mathbf{V}_{K-}^T \\ \mathbf{S}^T \end{pmatrix} = \begin{pmatrix} \mathbf{U} \\ \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \\ \vdots \\ \mathbf{V}_K \\ \mathbf{0} \end{pmatrix}
\end{aligned}$$

$$\begin{bmatrix}
\mathbf{A}_1^0 & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I}_n & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{A}_2^0 & \dots & \mathbf{0} & \mathbf{0} & \mathbf{I}_n & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\
\vdots & & \ddots & & & & & & & & & & \ddots & & & \vdots \\
\mathbf{0} & \mathbf{0} & \dots & \mathbf{A}_K^0 & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I}_n & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} \\
- & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
\mathbf{I}_m & \mathbf{I}_m & \dots & \mathbf{I}_m & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{I}_m
\end{bmatrix}
\begin{pmatrix}
\mathbf{X}_1^0 \\
\mathbf{X}_2^0 \\
\vdots \\
\mathbf{X}_K^0 \\
\mathbf{X}_{s1}^0 \\
\mathbf{X}_{s2}^0 \\
\vdots \\
\mathbf{X}_{sK}^0 \\
\mathbf{V}_{1+}^0 \\
\mathbf{V}_{1-}^0 \\
\mathbf{V}_{2+}^0 \\
\mathbf{V}_{2-}^0 \\
\vdots \\
\mathbf{V}_{K+}^0 \\
\mathbf{V}_{K-}^0 \\
\mathbf{S}^0
\end{pmatrix}
=
\begin{pmatrix}
\mathbf{0}_n \\
\mathbf{0}_n \\
\vdots \\
\mathbf{0}_n \\
\mathbf{0}_n \\
\mathbf{U}^0
\end{pmatrix}_{(nk+m) \times 1}$$

$$\begin{bmatrix}
\mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{I}_n & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & -\mathbf{I}_n & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\
\vdots & & \ddots & & & & & & & & & & \ddots & & & \vdots \\
\mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{I}_n & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
- & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
\mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0}
\end{bmatrix}
\begin{pmatrix}
\mathbf{X}_1^0 \\
\mathbf{X}_2^0 \\
\vdots \\
\mathbf{X}_K^0 \\
\mathbf{X}_{s1}^0 \\
\mathbf{X}_{s2}^0 \\
\vdots \\
\mathbf{X}_{sK}^0 \\
\mathbf{V}_{1+}^0 \\
\mathbf{V}_{1-}^0 \\
\mathbf{V}_{2+}^0 \\
\mathbf{V}_{2-}^0 \\
\vdots \\
\mathbf{V}_{K+}^0 \\
\mathbf{V}_{K-}^0 \\
\mathbf{S}^0
\end{pmatrix}
+$$



$$\begin{bmatrix}
\mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{I}_n & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & -\mathbf{I}_n & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
& & \ddots & & & & & & & & & & & \ddots & & \\
\mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{I}_n & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
- & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
\mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0}
\end{bmatrix}
\begin{pmatrix}
\mathbf{X}_1^{T-1} \\
\mathbf{X}_2^{T-1} \\
\vdots \\
\mathbf{X}_K^{T-1} \\
\mathbf{X}_{s1}^{T-1} \\
\mathbf{X}_{s2}^{T-1} \\
\vdots \\
\mathbf{X}_{sK}^{T-1} \\
\mathbf{V}_{1+}^{T-1} \\
\mathbf{V}_{1-}^{T-1} \\
\mathbf{V}_{2+}^{T-1} \\
\mathbf{V}_{2-}^{T-1} \\
\vdots \\
\mathbf{V}_{K+}^{T-1} \\
\mathbf{V}_{K-}^{T-1} \\
\mathbf{S}^{T-1}
\end{pmatrix}
+
\begin{bmatrix}
\mathbf{A}_1^T & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I}_n & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{A}_2^T & \dots & \mathbf{0} & \mathbf{0} & \mathbf{I}_n & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\
& & \ddots & & & & & & & & & & & \ddots & & \\
\mathbf{0} & \mathbf{0} & \dots & \mathbf{A}_K^T & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I}_n & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{I}_n & \mathbf{I}_n & \mathbf{0} \\
- & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
\mathbf{I}_m & \mathbf{I}_m & \dots & \mathbf{I}_m & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{I}_m
\end{bmatrix}
\begin{pmatrix}
\mathbf{X}_1^T \\
\mathbf{X}_2^T \\
\vdots \\
\mathbf{X}_K^T \\
\mathbf{X}_{s1}^T \\
\mathbf{X}_{s2}^T \\
\vdots \\
\mathbf{X}_{sK}^T \\
\mathbf{V}_{1+}^T \\
\mathbf{V}_{1-}^T \\
\mathbf{V}_{2+}^T \\
\mathbf{V}_{2-}^T \\
\vdots \\
\mathbf{V}_{K+}^T \\
\mathbf{V}_{K-}^T \\
\mathbf{S}^T
\end{pmatrix}
=
\begin{pmatrix}
\mathbf{0}_n \\
\mathbf{0}_n \\
\vdots \\
\mathbf{0}_n \\
- \\
\mathbf{U}^T
\end{pmatrix}$$

$$[\mathbf{0}]_{(mk+nk+2nk+m) \times 1} \leq
\begin{pmatrix}
\mathbf{X}'_1 \\
\mathbf{X}'_2 \\
\vdots \\
\mathbf{X}'_K \\
\mathbf{X}'_{s1} \\
\mathbf{X}'_{s2} \\
\vdots \\
\mathbf{X}'_{sK} \\
\mathbf{V}'_{1+} \\
\mathbf{V}'_{1-} \\
\mathbf{V}'_{2+} \\
\mathbf{V}'_{2-} \\
\vdots \\
\mathbf{V}'_{K+} \\
\mathbf{V}'_{K-} \\
\mathbf{S}'
\end{pmatrix}
\leq
\begin{pmatrix}
\mathbf{U}'_1 \\
\mathbf{U}'_2 \\
\vdots \\
\mathbf{U}'_K \\
\mathbf{U}'_{s1} \\
\mathbf{U}'_{s2} \\
\vdots \\
\mathbf{U}'_{sK} \\
\mathbf{V}_1 \\
\mathbf{V}_1 \\
\mathbf{V}_2 \\
\mathbf{V}_2 \\
\vdots \\
\mathbf{V}_K \\
\mathbf{V}_K \\
\mathbf{U}^t
\end{pmatrix}
_{(mk+nk+2nk+m) \times 1}
\quad \forall t \in \mathbf{N} \cup \{T\}.$$

The system arisen above emphasizes that the production/usage/storage at one time period might be affected by production/storage of the previous time step for any product, and affects the amount of usage/storage of the following time period. The foregoing reduces to the following matrix form

$$\text{Min}_{[\mathbf{X}_q^t], [\mathbf{X}_{sq}^t], [\mathbf{V}_{q+}^t], [\mathbf{V}_{q-}^t]} \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} [\mathbf{C}_q^t] [\mathbf{X}_q^t] + \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} [\mathbf{C}_{sq}^t] [\mathbf{X}_{sq}^t],$$

$$\sum_{t \in \mathbf{N}} ([\mathbf{X}_1^t] + \dots + [\mathbf{X}_K^t]) \leq [\mathbf{U}]_{m \times 1},$$

$$\sum_{t \in \mathbf{N}} ([\mathbf{V}_{q+}^t] - [\mathbf{V}_{q-}^t]) = [\mathbf{V}_q]_{n \times 1} \quad \forall q \in \mathbf{K},$$

$$[\mathbf{A}_q^t]_{n \times m} [\mathbf{X}_q^t] + [\mathbf{I}]_{n \times n} [\mathbf{X}_{sq}^t]_{n \times 1} + [-\mathbf{I}]_{n \times n} [\mathbf{X}_{sq}^{t-1}]_{n \times 1} - ([\mathbf{V}_{q+}^t] - [\mathbf{V}_{q-}^t]) = [\mathbf{0}]_{n \times 1} \quad \forall t \in \mathbf{N}, \forall q \in \mathbf{K},$$

$$[\mathbf{X}_1^t] + \dots + [\mathbf{X}_K^t] \leq [\mathbf{U}^t]_{m \times 1} \quad \forall t \in \mathbf{N},$$

$$[\mathbf{0}]_{m \times 1} \leq [\mathbf{X}_q^t]_{m \times 1} \leq [\mathbf{U}_q^t]_{m \times 1} \quad \forall t \in \mathbf{N}, \forall q \in \mathbf{K},$$

$$[\mathbf{0}]_{n \times 1} \leq [\mathbf{X}_{sq}^t]_{n \times 1} \leq [\mathbf{U}_{sq}^t]_{n \times 1} \quad \forall t \in \mathbf{N}, \forall q \in \mathbf{K},$$

$$[\mathbf{V}_{q+}^t]_{n \times 1} \geq [\mathbf{0}]_{n \times 1}, \quad [\mathbf{V}_{q-}^t]_{n \times 1} \geq [\mathbf{0}]_{n \times 1} \quad \forall t \in \mathbf{N}, \forall q \in \mathbf{K}.$$

The matrix form of the problem shows that we can formulate any min-cost problem on a multiperiod dynamic network (with storage) as a problem which possesses the *staircase structured system*. To show this, let's define

$$[\mathbf{Y}^t] := (\mathbf{X}_1^t \quad \mathbf{X}_2^t \quad \dots \quad \mathbf{X}_K^t \quad \mathbf{X}_{s1}^t \quad \mathbf{X}_{s2}^t \quad \dots \quad \mathbf{X}_{sK}^t \quad \mathbf{V}_{1+}^t \quad \mathbf{V}_{1-}^t \quad \mathbf{V}_{2+}^t \quad \mathbf{V}_{2-}^t \quad \dots \quad \mathbf{V}_{K+}^t \quad \mathbf{V}_{K-}^t \quad \mathbf{S}^t) \quad t \in \{0, 1, \dots, T\},$$

$$[\mathbf{C}^t] := (\mathbf{C}_1^t \quad \mathbf{C}_2^t \quad \dots \quad \mathbf{C}_K^t \quad \mathbf{C}_{s1}^t \quad \mathbf{C}_{s2}^t \quad \dots \quad \mathbf{C}_{sK}^t \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{0}) \quad t \in \{0, 1, \dots, T\},$$

$$[\mathbf{W}^t] := (\mathbf{U}_1^t \quad \mathbf{U}_2^t \quad \dots \quad \mathbf{U}_K^t \quad \mathbf{U}_{s1}^t \quad \mathbf{U}_{s2}^t \quad \dots \quad \mathbf{U}_{sK}^t \quad \mathbf{V}_1 \quad \mathbf{V}_1 \quad \mathbf{V}_2 \quad \mathbf{V}_2 \quad \dots \quad \mathbf{V}_K \quad \mathbf{V}_K \quad \mathbf{U}^t)^{trans} \quad t \in \{0, 1, \dots, T\},$$

$$[\bar{\mathbf{W}}^t] := (\mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad \vdots \quad \mathbf{U})^{trans} \quad t \in \{0, 1, \dots, T\},$$

$$[\mathbf{M}] := \begin{bmatrix} \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & & \ddots & & & & & & & & & & \ddots & & \vdots & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & -\mathbf{I} & \mathbf{0} \\ - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (\text{master matrix})$$

$$[\mathbf{A}] = [\mathbf{A}^t] := \begin{bmatrix} \mathbf{A}_1^t & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2^t & \dots & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ & & \ddots & & & & & & & & & & \ddots & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{A}_K^t & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{I} & \mathbf{I} & \mathbf{0} \\ - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\ \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \quad t \in \{0, 1, \dots, T\},$$

$$[\bar{\mathbf{M}}] := \begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & & \ddots & & & & & & & & & & \ddots & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Now, by considering the constraints structure and using our notations, we obtain

$$\begin{aligned} \text{Min} \quad & \sum_{t \in \mathbf{N}} [\mathbf{C}^t][\mathbf{Y}^t] \\ & [\mathbf{M}][\mathbf{Y}^{-1}] + [\mathbf{M}][\mathbf{Y}^0] + [\mathbf{M}][\mathbf{Y}^1] + \dots + [\mathbf{M}][\mathbf{Y}^{T-1}] + [\mathbf{M}][\mathbf{Y}^T] = (\mathbf{U} \quad \mathbf{V}_1 \quad \dots \quad \mathbf{V}_K \quad \mathbf{0})^{trans}, \\ & [\bar{\mathbf{M}}][\mathbf{Y}^{-1}] + [\mathbf{A}][\mathbf{Y}^0] = [\bar{\mathbf{W}}^0], \\ & [\bar{\mathbf{M}}][\mathbf{Y}^0] + [\mathbf{A}][\mathbf{Y}^1] = [\bar{\mathbf{W}}^1], \\ & [\bar{\mathbf{M}}][\mathbf{Y}^1] + [\mathbf{A}][\mathbf{Y}^2] = [\bar{\mathbf{W}}^2], \\ & [\bar{\mathbf{M}}][\mathbf{Y}^2] + [\mathbf{A}][\mathbf{Y}^3] = [\bar{\mathbf{W}}^3], \\ & \vdots \\ & [\bar{\mathbf{M}}][\mathbf{Y}^{T-1}] + [\mathbf{A}][\mathbf{Y}^T] = [\bar{\mathbf{W}}^T], \\ & [\mathbf{0}] \leq [\mathbf{Y}^t] \leq [\mathbf{W}^t] \quad t \in \{0, 1, \dots, T\}, \end{aligned}$$

where  $[\mathbf{Y}^{-1}] = [\mathbf{0}]$ . Note that we assume  $[\mathbf{A}] = [\mathbf{A}^t]$  for every  $t$ , since the underlying network will remain unchanged over time with respect to any product. Otherwise, we can use incidence matrix  $[\mathbf{A}^t]$  for time period  $t$  in the staircase formulation. Such structures have enjoyed a wide variety of applications in the form of production



planning and/or scheduling, distribution, integrated production-distribution models, manpower smoothing models, discrete optimal control problems with efficient handling. However, the most general approach might be the technique of using decomposition approach. In other word, the original problem is reformulated into a series of structurally similar LP subproblems with more tractable combinatorial structures. The structural similarity of the subproblems allows us to use decomposition techniques to well improve the computational efficiency [83]. In addition to structural similarity, this modelling technique has converted the MCDF problem to one with many sparse matrices/subproblems.

### 3.4 MM Networks with Storage and Spoilage (SSMM Network Flows)

This Section considers the most general case of multiperiod networks, i.e., MMN with storage at nodes and spoilage in arcs. We call these networks as *SSMM network flows*. The SSMM problem develops the MM network flow problem by allowing the  $q$ -flow to leak and be stored, at the same time, as it is sent through the network. Naturally, the *min-cost SSMMN flow problem* in the continuous time setting will be of the form stated in (3.41)-(3.47).

$$\mathbf{Min} \quad \sum_{q \in \mathbf{K}} \sum_{(i,j) \in \mathbf{A}} \int_0^T c_{ijq}(t) x_{ijq}(t) dt + \sum_{q \in \mathbf{K}} \sum_i \int_0^T c_{iq}(t) \sum_j [\lambda_{jiq}(t) x_{jiq}(t) - x_{ijq}(t)] dt, \quad (3.41)$$

$$\sum_j \int_0^T x_{ijq}(t) dt - \sum_j \int_0^T \lambda_{jiq}(t) x_{jiq}(t) dt = v_{iq} \quad \forall i \in \mathbf{V}, \forall q \in \mathbf{K}, \quad (3.42)$$

$$\sum_j \int_0^\theta x_{ijq}(t) dt - \sum_j \int_0^\theta \lambda_{jiq}(t) x_{jiq}(t) dt \leq 0 \quad \forall \theta \in [0, T[, \forall i \in \mathbf{V} \setminus \mathbf{VSQ}, \forall q \in \mathbf{K}, \quad (3.43)$$

$$\sum_{q \in \mathbf{K}} \int_0^T x_{ijq}(t) dt \leq u_{ij} \quad \forall (i, j) \in \mathbf{A}, \quad (3.44)$$

$$\sum_{q \in \mathbf{K}} x_{ijq}(t) \leq u_{ij}(t) \quad \forall (i, j) \in \mathbf{A}, \forall t \in [0, T], \quad (3.45)$$

$$0 \leq x_{ijq}(t) \leq u_{ijq}(t) \quad \forall (i, j) \in \mathbf{A}, \forall t \in [0, T], \forall q \in \mathbf{K}, \quad (3.46)$$

$$x_{ijq}(t) = 0 \quad \forall (i, j) \in \mathbf{A}, \forall t > T, \forall q \in \mathbf{K}. \quad (3.47)$$

If we replace (3.15) and (3.16) by (3.48) and (3.49), we get the min-cost SSMM network problem in the discrete time setting subject to introducing free variables  $v_{iq}^t := v_{iq+}^t - v_{iq-}^t$  ( $v_{iq+}^t \geq 0, v_{iq-}^t \geq 0$ ).

$$\mathbf{Min}_{x_{ij}^t, x_{ii}^t, v_i^t} \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} \sum_{(i,j) \in \mathbf{A}} c_{ijq}^t x_{ijq}^t + \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} \sum_{i \in \mathbf{V}} c_{iiq}^t x_{iiq}^t, \quad (3.48)$$

$$x_{iiq}^t - x_{iiq}^{t-1} + \sum_j x_{ijq}^t - \sum_j [\lambda_{jiq}^t x_{jiq}^t] - v_{iq}^t = 0 \quad \forall t \in \mathbf{N}, \forall i \in \mathbf{V}, \forall q \in \mathbf{K}. \quad (3.49)$$

To develop a polyhedral-based approach, we again use the  $t$ - $q$ -node-arc incidence matrix introduced in (3.23). It yields the node-arc incidence matrices of SS network with respect to each time step and commodity. As before, let  $[\mathbf{B}_q^t]_{n \times m}$  be the  $q$ - $t$ -node-arc incidence matrix at step  $t$  with respect to  $q$ . Therefore, having done all the necessary changes and transformations, the min-cost SSMM network problem can be modeled as that in previous Section. The only difference will be the polyhedrals' matrices. It suffices to make the following change with respect to each time period.

$$\begin{bmatrix} \mathbf{A}_1^t & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2^t & \dots & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ & & \ddots & & & & & & & & & & \ddots & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{A}_K^t & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{I} & \mathbf{I} & \mathbf{0} \\ - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\ \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}$$

↓

$$\begin{bmatrix} \mathbf{B}_1^t & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_2^t & \dots & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ & & \ddots & & & & & & & & & & \ddots & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{B}_K^t & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{I} & \mathbf{I} & \mathbf{0} \\ - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\ \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

The process of identifying block structures is a key step for our modeling approach. It might be supposed that it would be easy to identify a block structure from a constraint matrix of an arbitrary problem instance, but this is not the case. Even with a relatively small problem instance, it is a hard task to pick out a block structure visually from a plot

of the non-zero elements, when the rows and columns are not arranged to expose it. In other words, the way the model is developed is critical for constructing/identifying the block structures [13][53][85][86]. One may consult Voelker [85], Borndorfer and Ferreira [13], Kernighan and Lin [53], and Weil and Kettler [86] to get more information on this matter.

### 3.5 An Alternative Approach for MM Networks Having no Period Capacity

If we slightly change the definition of our master matrix and our decision vectors, we get the following alternatives for the min-cost flow problem on a MM network. In this setting, we define our parameters so as they look time-commodity varying. Consequently, we allow several different convex combinations and linear combinations (think of Minkowski theorem) for each subproblem (the set of constraints for each time step and product) since, as you will see, we have  $K(T+1)$  convexity constraints. This adds more flexibility to our approaches (in a lower dimensional space), but at the same time, increases the number of polyhedrals (and also, subproblems) from  $(T+1)$  to  $K(T+1)$

#### 3.5.1 Case 1: MM Networks with Spoilage

In this setting, we let our decision vectors  $[\mathbf{Y}_q^t]$  be time-commodity varying. They will consist of one set of nonnegative variables  $\mathbf{X}_q^t$  in the  $q$ -th place for each  $t$ , and  $\mathbf{V}_{q+}^t$  and  $\mathbf{V}_{q-}^t$  in the  $(K+2q-1)$ -th and  $(K+2q)$ -th places, respectively, and the other entries are set to zero. Note that, any place in our vector is a group of arc variables.

$$[\mathbf{Y}_q^t] := \left[ \mathbf{0} \ \mathbf{0} \ \dots \ \underbrace{\mathbf{X}_q^t}_q \ \mathbf{0} \ \dots \ \underbrace{\mathbf{V}_{q+}^t}_{(K+2q-1)} \ \underbrace{\mathbf{V}_{q-}^t}_{(K+2q)} \ \mathbf{0} \ \mathbf{0} \ \mathbf{0} \right]^{trans} \quad \forall q \in \mathbf{K}, \forall t \in \mathbf{N}.$$

Analogously, we should change our nod-arc incidence matrices with respect to each  $t$  and  $q$ .

$$[\mathbf{A}_q^t] := \left[ \mathbf{0} \ \mathbf{0} \ \dots \ \mathbf{B}_q^t \ \mathbf{0} \ \dots \ -\mathbf{I} \ \mathbf{I} \ \mathbf{0} \ \mathbf{0} \ \mathbf{0} \right] \quad \forall q \in \mathbf{K}, \forall t \in \mathbf{N}.$$

We also need to redefine our master matrix as follows.

$$[\mathbf{M}] := \begin{bmatrix} \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & -\mathbf{I} & \dots & \mathbf{0} & \mathbf{0} \\ & & & \ddots & & & & \ddots & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & -\mathbf{I} \end{bmatrix}.$$

In this case, also, the modeling procedure reveals that we can extract the block-angular structure from the problem, but with a bit more flexibility due to having *time-commodity varying subproblems*. Hence, the flow conservation conditions turn out to have the following form.

$$\begin{aligned} & [\mathbf{M}] \begin{pmatrix} \mathbf{X}_1^0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{V}_{1+}^0 \\ \mathbf{V}_{1-}^0 \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} + [\mathbf{M}] \begin{pmatrix} \mathbf{0} \\ \mathbf{X}_2^0 \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{V}_{2+}^0 \\ \mathbf{V}_{2-}^0 \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} + \dots + [\mathbf{M}] \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{X}_K^0 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{V}_{K+}^0 \\ \mathbf{V}_{K-}^0 \end{pmatrix} + [\mathbf{M}] \begin{pmatrix} \mathbf{X}_1^1 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{V}_{1+}^1 \\ \mathbf{V}_{1-}^1 \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} + [\mathbf{M}] \begin{pmatrix} \mathbf{0} \\ \mathbf{X}_2^1 \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{V}_{2+}^1 \\ \mathbf{V}_{2-}^1 \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} + \dots + [\mathbf{M}] \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{X}_K^1 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{V}_{K+}^1 \\ \mathbf{V}_{K-}^1 \end{pmatrix} \\ & + \dots + [\mathbf{M}] \begin{pmatrix} \mathbf{X}_1^T \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{V}_{1+}^T \\ \mathbf{V}_{1-}^T \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} + \dots + [\mathbf{M}] \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{X}_K^T \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{V}_{K+}^T \\ \mathbf{V}_{K-}^T \end{pmatrix} = \begin{pmatrix} \mathbf{U} \\ \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \\ \vdots \\ \mathbf{V}_K \end{pmatrix}, \end{aligned}$$

$$[\mathbf{A}_1^0] [\mathbf{X}_1^0 \ \mathbf{0} \ \dots \ \mathbf{0} \ \mathbf{V}_{1+}^0 \ \mathbf{V}_{1-}^0 \ \mathbf{0} \ \mathbf{0} \ \dots \ \mathbf{0} \ \mathbf{0}]^{trans} = [\mathbf{0}]_{n \times 1},$$

$$[\mathbf{A}_2^0] [\mathbf{0} \ \mathbf{X}_2^0 \ \dots \ \mathbf{0} \ \mathbf{0} \ \mathbf{0} \ \mathbf{0} \ \mathbf{V}_{2+}^0 \ \mathbf{V}_{2-}^0 \ \mathbf{0} \ \mathbf{0}]^{trans} = [\mathbf{0}]_{n \times 1},$$

⋮

$$[\mathbf{A}_K^0] [\mathbf{0} \ \mathbf{0} \ \dots \ \mathbf{X}_K^0 \ \mathbf{0} \ \mathbf{0} \ \mathbf{0} \ \mathbf{0} \ \dots \ \mathbf{V}_{K+}^0 \ \mathbf{V}_{K-}^0]^{trans} = [\mathbf{0}]_{n \times 1},$$

⋮

$$\begin{aligned}
[\mathbf{A}_1^T] [\mathbf{X}_1^T \ \mathbf{0} \ \dots \ \mathbf{0} \ \mathbf{V}_{1+}^T \ \mathbf{V}_{1-}^T \ \mathbf{0} \ \mathbf{0} \ \dots \ \mathbf{0} \ \mathbf{0}]^{trans} &= [\mathbf{0}]_{n \times 1}, \\
&\vdots \\
[\mathbf{A}_K^T] [\mathbf{0} \ \mathbf{0} \ \dots \ \mathbf{X}_K^T \ \mathbf{0} \ \mathbf{0} \ \mathbf{0} \ \mathbf{0} \ \dots \ \mathbf{V}_{K+}^T \ \mathbf{V}_{K-}^T]^{trans} &= [\mathbf{0}]_{n \times 1}, \\
[\mathbf{0}] \leq [\mathbf{0} \ \mathbf{0} \ \dots \ \mathbf{X}_q^t \ \mathbf{0} \ \dots \ \mathbf{V}_{q+}^t \ \mathbf{V}_{q-}^t \ \mathbf{0} \ \mathbf{0} \ \mathbf{0}] &\leq [\mathbf{0} \ \mathbf{0} \ \dots \ \mathbf{U}_q^t \ \mathbf{0} \ \dots \ \mathbf{V}_q \ \mathbf{V}_q \ \mathbf{0} \ \mathbf{0} \ \mathbf{0}] \\
&\quad \forall q \in \mathbf{K}, \forall t \in \mathbf{N}.
\end{aligned}$$

Applying our notations yields the desired structure provided that we define the cost and capacity vectors as follows.

$$\begin{aligned}
[\mathbf{C}^t] &:= [\mathbf{0} \ \mathbf{0} \ \dots \ \mathbf{C}_q^t \ \mathbf{0} \ \dots \ \mathbf{0} \ \mathbf{0} \ \mathbf{0} \ \mathbf{0} \ \mathbf{0}] \quad \forall q \in \mathbf{K}, \forall t \in \mathbf{N}, \\
[\mathbf{W}_q^t] &:= [\mathbf{0} \ \mathbf{0} \ \dots \ \mathbf{U}_q^t \ \mathbf{0} \ \dots \ \mathbf{V}_q \ \mathbf{V}_q \ \mathbf{0} \ \mathbf{0} \ \mathbf{0}] \quad \forall q \in \mathbf{K}, \forall t \in \mathbf{N}.
\end{aligned}$$

Then we conclude the following form for the min-cost MM network problem with spoilage.

$$\begin{aligned}
\text{Min} \quad & \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} [\mathbf{C}_q^t][\mathbf{Y}_q^t] \\
[\mathbf{M}][\mathbf{Y}_1^0] + \dots + [\mathbf{M}][\mathbf{Y}_K^0] + [\mathbf{M}][\mathbf{Y}_1^1] + \dots + [\mathbf{M}][\mathbf{Y}_K^1] + \dots + [\mathbf{M}][\mathbf{Y}_1^T] + \dots + [\mathbf{M}][\mathbf{Y}_K^T] &= [\mathbf{U} \ \mathbf{V}_1 \ \dots \ \mathbf{V}_K]^{trans}, \\
[\mathbf{A}_1^0][\mathbf{Y}_1^0] &= [\mathbf{0}], \\
&\vdots \\
[\mathbf{A}_K^0][\mathbf{Y}_K^0] &= [\mathbf{0}], \\
&\vdots \\
[\mathbf{A}_1^1][\mathbf{Y}_1^1] &= [\mathbf{0}], \\
&\vdots \\
[\mathbf{A}_K^1][\mathbf{Y}_K^1] &= [\mathbf{0}], \\
&\vdots \\
&\vdots \\
[\mathbf{A}_1^T][\mathbf{Y}_1^T] &= [\mathbf{0}], \\
&\vdots \\
[\mathbf{A}_K^T][\mathbf{Y}_K^T] &= [\mathbf{0}], \\
[\mathbf{0}] \leq [\mathbf{Y}_q^t] \leq [\mathbf{W}_q^t] &\quad \forall q \in \mathbf{K}, \forall t \in \mathbf{N}.
\end{aligned}$$

To proceed, we need to define  $K(T+1)$  time-commodity varying polyhedrals  $\chi_q^t$  and Minkowski's mapping as follows.

$$\begin{aligned}\chi_q^t &:= \{[\mathbf{Y}_q^t] : [\mathbf{A}_q^t][\mathbf{Y}_q^t] = [\mathbf{0}], \quad [\mathbf{0}] \leq [\mathbf{Y}_q^t] \leq [\mathbf{W}_q^t]\} \quad \forall q \in \mathbf{K}, \forall t \in \mathbf{N}, \\ [\mathbf{Y}_q^t] \in \chi_q^t &\Leftrightarrow [\mathbf{Y}_q^t] = \sum_{i=1}^{k_q^t} \alpha_{iq}^t [\mathbf{y}_{iq}^t] + \sum_{j=1}^{l_q^t} \mu_{jq}^t [\mathbf{d}_{jq}^t] \quad \forall q \in \mathbf{K}, \forall t \in \mathbf{N}, \\ \sum_{i=1}^{k_q^t} \alpha_{iq}^t &= 1, \quad \alpha_{iq}^t \geq 0, \quad \mu_{jq}^t \geq 0 \quad \forall q \in \mathbf{K}, \forall t \in \mathbf{N},\end{aligned}$$

where  $[\mathbf{y}_{1q}^t], [\mathbf{y}_{2q}^t], \dots, [\mathbf{y}_{k_q^t q}^t]$  and  $[\mathbf{d}_{1q}^t], [\mathbf{d}_{2q}^t], \dots, [\mathbf{d}_{l_q^t q}^t]$  are extreme points and extreme directions (if any) of polyhedrals  $\chi_q^t$ . The same analysis as that in Section 3 then can be applied.

### 3.5.2 Case 2: MM Networks with Storage

In this setting, our time-commodity varying decision vectors  $[\mathbf{Y}_q^t]$  will consist of one set of nonnegative variables  $\mathbf{x}_q^t$  and  $\mathbf{x}_{sq}^t$  in the  $q$ -th and  $(q+K)$ -th places for each  $t$ , and  $\mathbf{V}_{q+}^t$  and  $\mathbf{V}_{q-}^t$  in the  $(2K+2q-1)$ -th and  $(2K+2q)$ -th places, respectively, and the other entries are set to zero.

$$[\mathbf{Y}_q^t] := \left[ \begin{array}{cccccccccccc} \mathbf{0} & \mathbf{0} & \dots & \underbrace{\mathbf{x}_q^t}_q & \mathbf{0} & \dots & \mathbf{0} & \underbrace{\mathbf{x}_{sq}^t}_{(K+q)} & \mathbf{0} & \dots & \underbrace{\mathbf{V}_{q-}^t}_{(2K+2q-1)} & \underbrace{\mathbf{V}_{q+}^t}_{(2K+2q)} & \mathbf{0} & \dots & \mathbf{0} \end{array} \right]^{trans} \quad \forall q \in \mathbf{K}, \forall t \in \mathbf{N}.$$

Similarly, we change our nod-arc incidence matrices with respect to each  $t$  and  $q$ .

$$[\mathbf{C}] := [\mathbf{A}_q^t] := [\mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{A} \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad \mathbf{I} \quad \mathbf{0} \quad \dots \quad -\mathbf{I} \quad \mathbf{I} \quad \mathbf{0} \quad \dots \quad \mathbf{0}] \quad \forall q \in \mathbf{K}, \forall t \in \mathbf{N},$$

$$[\bar{\mathbf{C}}] := [\mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad -\mathbf{I} \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{0}] \quad \forall q \in \mathbf{K}, \forall t \in \mathbf{N}.$$

We should redefine our master matrix as following.

$$[\mathbf{M}] := \begin{bmatrix} \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ & & \ddots & & & & & & & & & & & & \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & -\mathbf{I} \end{bmatrix}.$$

Hence, the flow conservation constrains will be of the following from.

$$\begin{aligned}
& [\mathbf{M}] \begin{pmatrix} \mathbf{X}_1^0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{X}_{s1}^0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{V}_{1+}^0 \\ \mathbf{V}_{1-}^0 \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} + [\mathbf{M}] \begin{pmatrix} \mathbf{0} \\ \mathbf{X}_2^0 \\ \vdots \\ \mathbf{0} \\ \mathbf{X}_{s2}^0 \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{V}_{2+}^0 \\ \mathbf{V}_{2-}^0 \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} + \dots + [\mathbf{M}] \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{X}_K^0 \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{X}_{sK}^0 \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{V}_{K+}^0 \\ \mathbf{V}_{K+}^0 \end{pmatrix} + [\mathbf{M}] \begin{pmatrix} \mathbf{X}_1^1 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{X}_{s1}^1 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{V}_{1+}^1 \\ \mathbf{V}_{1-}^1 \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} + [\mathbf{M}] \begin{pmatrix} \mathbf{0} \\ \mathbf{X}_2^1 \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{X}_{s2}^1 \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{V}_{2+}^1 \\ \mathbf{V}_{2-}^1 \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} + \dots + [\mathbf{M}] \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{X}_K^1 \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{X}_{sK}^1 \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{V}_{K+}^1 \\ \mathbf{V}_{K+}^1 \end{pmatrix} + \dots + [\mathbf{M}] \begin{pmatrix} \mathbf{X}_1^T \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{X}_{s1}^T \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{V}_{1+}^T \\ \mathbf{V}_{1-}^T \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} + \\
& \dots + [\mathbf{M}] \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{X}_K^T \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{X}_{sK}^T \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{V}_{K+}^T \\ \mathbf{V}_{K+}^T \end{pmatrix} = \begin{pmatrix} \mathbf{U} \\ \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \\ \vdots \\ \mathbf{V}_K \end{pmatrix},
\end{aligned}$$

$$[\mathbf{C}] \begin{pmatrix} \mathbf{X}_1^t \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{X}_{s1}^t \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{V}_{1+}^t \\ \mathbf{V}_{1-}^t \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} = [\mathbf{0}] \quad , \quad [\mathbf{C}] \begin{pmatrix} \mathbf{0} \\ \mathbf{X}_2^t \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{X}_{s2}^t \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{V}_{2+}^t \\ \mathbf{V}_{2-}^t \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} = [\mathbf{0}] \quad , \dots , \quad [\mathbf{C}] \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{X}_K^t \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{X}_{sK}^t \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{V}_{K+}^t \\ \mathbf{V}_{K-}^t \end{pmatrix} = [\mathbf{0}] \quad \forall t \in \mathbf{N} ,$$

$$[\bar{\mathbf{C}}] \begin{pmatrix} \mathbf{X}_1^{t-1} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{X}_{s1}^{t-1} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{V}_{1+}^{t-1} \\ \mathbf{V}_{1-}^{t-1} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} + [\mathbf{C}] \begin{pmatrix} \mathbf{X}_1^t \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{X}_{s1}^t \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{V}_{1+}^t \\ \mathbf{V}_{1-}^t \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} = [\mathbf{0}] \quad , \quad \forall t \in \mathbf{N}$$

$$[\bar{\mathbf{C}}] \begin{pmatrix} \mathbf{0} \\ \mathbf{X}_2^{t-1} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{X}_{s2}^{t-1} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{V}_{2+}^{t-1} \\ \mathbf{V}_{2-}^{t-1} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} + [\mathbf{C}] \begin{pmatrix} \mathbf{0} \\ \mathbf{X}_2^t \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{X}_{s2}^t \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{V}_{2+}^t \\ \mathbf{V}_{2-}^t \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} = [\mathbf{0}] \quad , \dots , \quad \forall t \in \mathbf{N}$$



$$\begin{aligned}
& \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{X}_K^{t-1} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{X}_{sK}^{t-1} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{V}_{K+}^{t-1} \\ \mathbf{V}_{K+}^{t-1} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{X}_K^t \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{X}_{sK}^t \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{V}_{K+}^t \\ \mathbf{V}_{K+}^t \end{pmatrix} = [\mathbf{0}] \quad \forall t \in \mathbb{N}
\end{aligned}$$

$$\begin{aligned}
[\mathbf{0}] \leq & \left[ \mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{X}_q^t \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad \mathbf{X}_{sq}^t \quad \mathbf{0} \quad \dots \quad \mathbf{V}_{q-}^t \quad \mathbf{V}_{q-}^t \quad \mathbf{0} \quad \dots \quad \mathbf{0} \right] \leq \\
& \left[ \mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{U}_q^t \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad \mathbf{U}_{sq}^t \quad \mathbf{0} \quad \dots \quad \mathbf{V}_q \quad \mathbf{V}_q \quad \mathbf{0} \quad \dots \quad \mathbf{0} \right] \\
& \forall q \in \mathbb{K}, \forall t \in \mathbb{N}.
\end{aligned}$$

Having defined some appropriate vectors, we can formulate any min-cost problem on a multiperiod dynamic network (with storage) as a problem which possesses the *staircase structured system*. To show this, let's define

$$[\mathbf{C}_q^t] := \left[ \mathbf{0} \quad \mathbf{0} \quad \dots \quad \underbrace{\mathbf{C}_q^t}_q \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad \underbrace{\mathbf{C}_{sq}^t}_{(K+q)} \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{0} \right] \quad \forall q \in \mathbb{K}, \forall t \in \mathbb{N},$$

$$[\mathbf{W}_q^t] := \left[ \mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{U}_q^t \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad \mathbf{U}_{sq}^t \quad \mathbf{0} \quad \dots \quad \mathbf{V}_q \quad \mathbf{V}_q \quad \mathbf{0} \quad \dots \quad \mathbf{0} \right] \quad \forall q \in \mathbb{K}, \forall t \in \mathbb{N}.$$

Now, reconsider the constraints structure and use our notations.

$$\mathbf{Min} \quad \sum_{q \in \mathbb{K}} \sum_{t \in \mathbb{N}} [\mathbf{C}_q^t][\mathbf{Y}_q^t]$$

$$\begin{aligned}
& [\mathbf{M}][\mathbf{Y}_1^0] + [\mathbf{M}][\mathbf{Y}_2^0] + \dots + [\mathbf{M}][\mathbf{Y}_K^0] + [\mathbf{M}][\mathbf{Y}_1^1] + [\mathbf{M}][\mathbf{Y}_2^1] + \dots + [\mathbf{M}][\mathbf{Y}_K^1] + \dots + [\mathbf{M}][\mathbf{Y}_1^T] + \dots + [\mathbf{M}][\mathbf{Y}_K^T] \\
& = [\mathbf{U} \quad \mathbf{V}_1 \quad \dots \quad \mathbf{V}_K]^{trans},
\end{aligned}$$

$$[\mathbf{C}][\mathbf{Y}_q^0] = [\mathbf{0}] \quad \forall q \in \mathbf{K},$$

$$[\bar{\mathbf{C}}][\mathbf{Y}_q^0] + [\mathbf{C}][\mathbf{Y}_q^1] = [\mathbf{0}] \quad \forall q \in \mathbf{K},$$

$$[\bar{\mathbf{C}}][\mathbf{Y}_q^1] + [\mathbf{C}][\mathbf{Y}_q^2] = [\mathbf{0}] \quad \forall q \in \mathbf{K},$$

\(\vdots\)

$$[\bar{\mathbf{C}}][\mathbf{Y}_q^{T-1}] + [\mathbf{C}][\mathbf{Y}_q^T] = [\mathbf{0}] \quad \forall q \in \mathbf{K},$$

$$[\mathbf{0}] \leq [\mathbf{Y}_q^t] \leq [\mathbf{W}_q^t] \quad \forall q \in \mathbf{K}, \forall t \in \mathbf{N}.$$

Note that we assume  $[\mathbf{C}] = [\mathbf{A}_q^t]$  for every  $t$  and  $q$ , since our distribution network is unchanged over time with respect to any product. Otherwise, we can use incidence matrix  $[\mathbf{A}_q^t]$  for time period  $t$  (for any  $q$ ) in the staircase formulation. Still our problem has a time lag of length one. Namely, the amount of  $q$ -flow that should be distributed depends on the previous step's distributed flow and stored flow.

### 3.5.3 Case 3: MM Networks with Storage and Spoilage (SS Networks)

To handle this case, let  $[\mathbf{B}_q^t]$  be the  $q$ - $t$ -node-arc incidence matrix at step  $t$  with respect to  $q$ . Hence, having done all the necessary changes and definitions the same as those in Case 2, it is enough to do the following replacement with respect to each  $t$  and  $q$ .

$$[\mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{A} \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad \mathbf{I} \quad \mathbf{0} \quad \dots \quad -\mathbf{I} \quad \mathbf{I} \quad \mathbf{0} \quad \dots \quad \mathbf{0}]$$

\(\downarrow\)

$$[\mathbf{0} \quad \mathbf{0} \quad \dots \quad \mathbf{B}_q^t \quad \mathbf{0} \quad \dots \quad \mathbf{0} \quad \mathbf{I} \quad \mathbf{0} \quad \dots \quad -\mathbf{I} \quad \mathbf{I} \quad \mathbf{0} \quad \dots \quad \mathbf{0}].$$

## 3.6 On Applications and Computational Tuning-Testing

Electricity and data transmissions, road or air traffic control, evacuation planning, production and distribution, and transportation problems can be viewed as single-commodity or multi-commodity problems. When the above models involve a parameter of time, they may often be modeled as dynamic network flows (or multiperiod dynamic

networks). This section applies DW to instances of a model of an electricity-distribution (transportation) network and we present the results of our experiments that demonstrate the effectiveness of our approach. The topology of this class of network models has been already described in detail in previous chapters and also in Chapters 1 and 2, so we give only a brief description here.

We consider a distribution network that is a local, low-voltage part of the electricity system that connects the customers to the long-distance, high-voltage transmission system which, in turn, connects to generating plants. The distribution network may be viewed as connecting to the transmission system, via a substation, at a single point or source (in reality, it may connect to several points). We consider a number of customers (cities, factories etc.) with demand for a certain product (that may vary over time) over a specified time period, e.g., demand for electricity in our case study. We assume that demand is satisfied by shipping electricity in a fixed number of wires from a number of supply/production sites, where the cost of production is assumed to be time varying (or fixed for each time period). We restrict our attention to the case in which each wire must unload all of its goods (electricity) at the demand site upon arrival. The objective is to determine the production circuit and shipping electricity within the time period so as to minimize the daily cost (the planning horizon time is a day). To illustrate the performance of our approach, we conducted a series of experiments using a set of real data from our case study on real and random complete-bipartite MMNs. However, our results should be applicable to other types of problems as well.

Several parameters must be specified in order to generate the MMN topology, arc capacities and costs, losses and gains, and node storage capacities (if desired). These parameters are random seed, number of periods  $T$ , number of products  $K$ , number of supply/demand nodes, indegree and outdegree of each node, minimum and maximum values of arc capacities for each  $q$  and  $t$ , losses, gains, and costs associated with the arcs, which must be all nonnegative. The cost on each arc (for each time period for each product) is randomly chosen from a uniform distribution between user defined parameter  $c_{\min}$  and  $c_{\max}$ , and gain/loss factors for each commodity and period are also chosen from a  $[0, \lambda_{\max}]$ -uniform distribution where  $\lambda_{\max}$  should be given for each  $q$ . The user also sets the number of time periods, supply nodes, and demand nodes. The

demand for each time step with respect to each product  $q$  is to be randomly chosen from a uniform distribution between  $v_{q\min}$  and  $v_{q\max}$ , and likewise for supply, storage capacity, spoilage, arc capacities.

The experiments are conducted on random multiperiod transportation networks with 26, 40, 46, 62, 74, 82, 100 nodes and time horizon 13, 20, 23, 31, 37, 41, 50 for the first two cases and random networks with 6, 10, 14, 22, 26, 30, 34 nodes and time horizon 3, 5, 7, 11, 13, 15, 17 for the last case. For each value of  $n$  ( $n=n_1+n_2$  is the total number of generating plants and demand sites), we create distribution networks with different indegree and outdegree in a range from 1 to  $\max\{n_1, n_2\}$ . We denote by  $\delta$  the density of the network ( $\delta = m/n$ ). The minimum and maximum loss/gains are set to 0.999 and 1.100, respectively, the minimum and maximum capacities are set to 50 and 70, respectively, the minimum and maximum costs are set to 10 and 100. For each specific setting of  $n$  and  $m$ , we test a random transportation MMN. In addition, we have performed several computational tests and analyzed decomposition approach's sensitivity on a variety of min-cost MMN problem instances. We have investigated different implementation ideas and sensitivity of the method to various data parameters, such as the number of arcs, number of time increments and the congestion in the multiperiod network. We generated many MMN of various sizes, different number of time periods, and different levels of congestion.

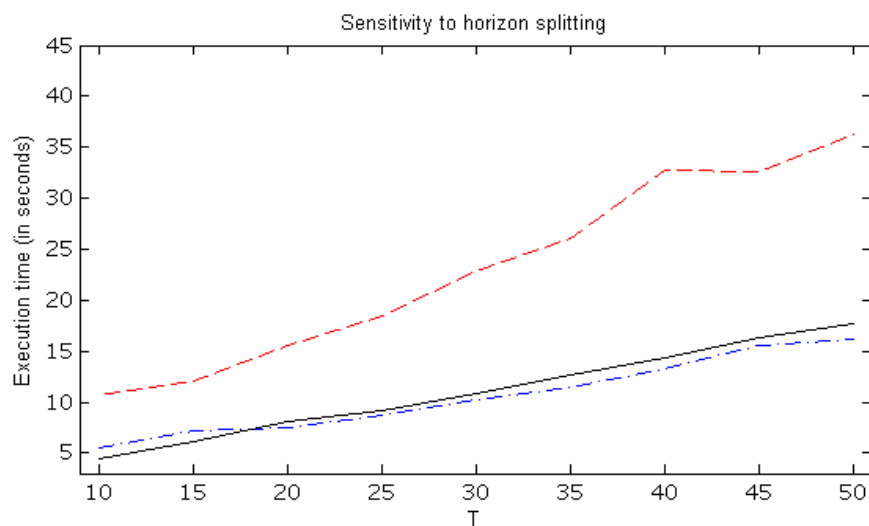


Figure 3.3 Sensitivity to time increments

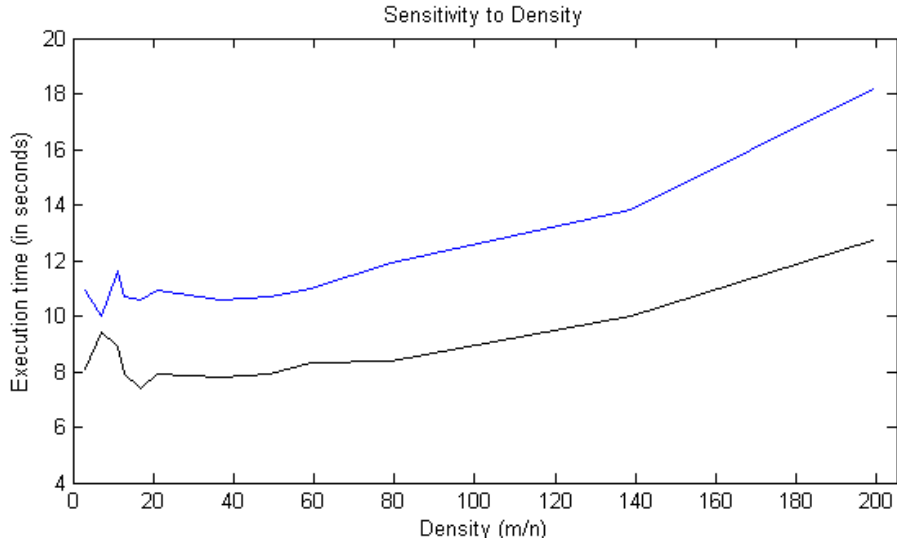


Figure 3.4 Sensitivity to density

A plot for each different network parameter,  $\delta$  and  $T$ , helps us visualize the effects of time splitting and density on the growth of the problem or the average CPU time. Figure 3.4 shows that execution time increases exponentially in denser networks. In any problem instances we tested, our solution approach showed an almost linear sensitivity with respect to time increments as shown in Figure 3.3. The same sensitivity is also observed with respect to number of products. This behavior is generally related to the increase in the numbers of subproblems; since any increase in the number of time periods or products will directly affect the numbers/complexity of subproblems, and consequently, the algorithm's running time increases. Our linear programming models, decomposition methods, were implemented in GAMS in a personal computer with a 2.13GHz processor and 4GB physical RAM. The results of a very small number of runs are summarized in Tables 3.1-3.4. Computational experiences shown in the first four rows of Tables 3.1-3.3 correspond to some electricity transmission MMNs from our case study.

**Table 3.1 Sizes and Computational Results for Some MMNs with  $|K|=1$  ( $T=13, 20, 23, 31, 37, 41, 50$ )**

	# of Polyhedrals	# of Variables	# of Constraints	# of Non-zeros	Work space Allocated (Mb)	Computational Time (s)
Data set 1	14	5070	4758	677	7.7	12.05
Data set 2	21	17600	16840	1601	8.2	17.10
Data set 3	24	26450	25438	2117	8.2	19.80
Data set 4	32	63426	61566	3845	9.3	26.90
Data set 5	38	106782	104118	5477	10.8	32.50
Data set 6	42	144566	141286	6725	11.9	37.10
Data set 7	51	260000	255100	10001	15.0	48.60

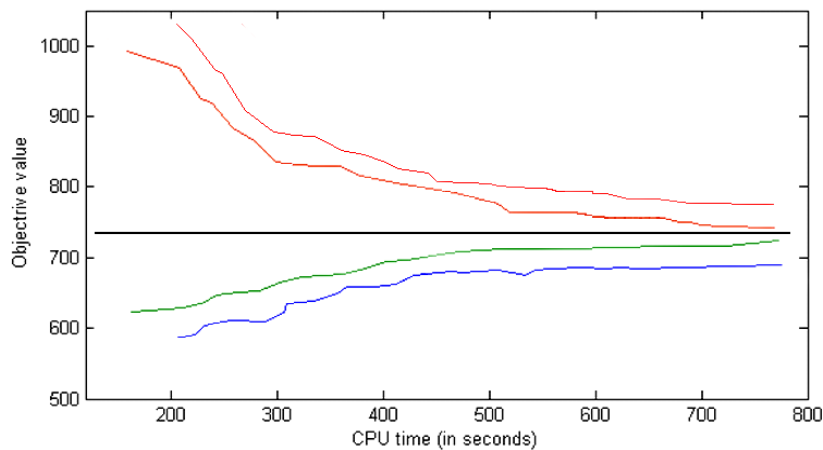
**Table 3.2 Sizes and Computational Results for Some MMNs with  $|K|=7$  ( $T=13, 20, 23, 31, 37, 41, 50$ )**

	<i># of Polyhedrals</i>	<i># of Variables</i>	<i># of Constraints</i>	<i># of Non-zeros</i>	<i>Work space Allocated (Mb)</i>	<i>Computational Time (s)</i>
<i>Data set 1</i>	98	35490	19110	677	8.7	70.9
<i>Data set 2</i>	147	123200	67480	1601	10.8	119.2
<i>Data set 3</i>	168	185150	101890	2117	12.9	142.9
<i>Data set 4</i>	224	443982	246450	3845	19.7	225.1
<i>Data set 5</i>	266	747474	416694	5477	27.1	319.9
<i>Data set 6</i>	294	1011962	565390	3362	33.9	764.7
<i>Data set 7</i>	357	1820000	1020700	10001	54.3	2510.2

**Table 3.3 Sizes and Computational Results for Some MMNs with  $|N|=2|K|=2T$  ( $T=3, 5, 7, 11, 13, 15, 17$ )**

	<i># of Polyhedrals</i>	<i># of Variables</i>	<i># of Constraints</i>	<i># of Non-zeros</i>	<i>Work space Allocated (Mb)</i>	<i>Computational Time (s)</i>
<i>Data set 1</i>	12	270	162	37	7.7	14.7
<i>Data set 2</i>	30	1750	950	101	7.7	37.8
<i>Data set 3</i>	56	6174	3234	197	7.7	68.6
<i>Data set 4</i>	132	34606	17666	485	8.7	134.2
<i>Data set 5</i>	182	65910	33462	677	9.8	114.6
<i>Data set 6</i>	240	114750	58050	901	10.8	145.3
<i>Data set 7</i>	306	186694	94214	1157	12.9	557.3

DW also provides a bound on the value of the objective function at each iteration, which allows the quality of the current solution to be assessed, so that the trade of between time and quality can be quantified. As a result, the procedure can be terminated, prior to finding an exact optimal solution, with a good estimate of how far the current value of the objective function can be from its optimal value [9][15][83].



**Figure 3.5 Objective and dual bound progress of two instances**

Practical experience suggests when decomposition is applied on an MCDF problem on an MMN, the algorithm makes substantial progress in the beginning, but the cost improvement becomes very slow later on. However, in spite of possible ill-conditioning of a decomposed problem, it usually turns out that its optimal solution is close to the true optimal solution, and so we may terminate it before the end having a very good suboptimal solution.

**Table 3.4 Computational Resources for Some Large Feasible MMNs**

An application of our approach and CPLEX for some large random feasible instances with: Uniform flow requirement $U(100, 900)$ , Uniform capacity $U(100,000*/K/, 1,000,000*/K/)$ , and Uniform cost $U(1,100)$ .						
For all problems, the planning horizon time is 24 hours, but with different discretization.						
<i>Instances</i> <i>Data</i>	<i>P.1</i>	<i>P.2</i>	<i>P.3</i>	<i>P.4</i>	<i>P.5</i>	<i>P.6</i>
<b>Problem's Status</b>	*** <i>Feasible</i> ***					
<b># Node-Commodities</b>	65,110	200,000	79,202	26,026	60,000	52,800
<b># Arc-Commodities</b>	6,155,700	1.00000E+7	7,880,599	1.30260E+7	9,000,000	2.11200E+7
<b># Commodities</b>	170	100	199	13	100	33
<b># Constraints</b>	6,293,230	1.03100E+7	8,039,003	1.50560E+7	9,240,000	2.24528E+7
<b># Variables</b>	6,155,700	1.00000E+7	7,880,599	1.30260E+7	9,000,000	2.11200E+7
<b>Network Density</b>	94.543	50.000	99.500	500.500	150.00	400.000
<b># Periods</b>	1/24	10/24	1/24	13/24	1/24	33/24
<b>Total Cost</b>	6.65645E+7	1.40419E+8	3.33284E+7	6,589,735.0	1.99913E+7	1.36029E+7
<b>Work Space Allocated for DW (reported by GAMS)</b>	321.7 Mb	517.3 Mb	407.2 Mb	916.8 Mb	471.2 Mb	1206.7 Mb
<b>Work Space Allocated for CPLEX (reported by GAMS)</b>	> 1,800 Mb	> 2,400 Mb	> 2,000 Mb	> 3,500 Mb	> 2,100 Mb	4,000 Mb
<b>Elapsed Time for DW (by GAMS)</b>	1,209 SECONDS	9,799 SECONDS	1,756 SECONDS	2,026 SECONDS	3,436 SECONDS	3,928 SECONDS
<b>Elapse Time for CPLEX (by GAMS)</b>	Failed to solve after 410.14 SEC	Failed to solve after 500.28 SEC	Failed to Solve after 400.00 SEC	Failed to Solve after 158.52 SEC	Failed to Solve after 401.89 SEC	Failed to Solve after 500.00 SEC

Figure 3.5 plots the progress of a pair of randomly generated MMN problems when we apply the decomposition method. The plots show how the objective values and dual bounds converge as the number of iterations (time) increases. Since we are dealing with min-cost flow problems, the objective value descends from above and the dual bound ascends from below.

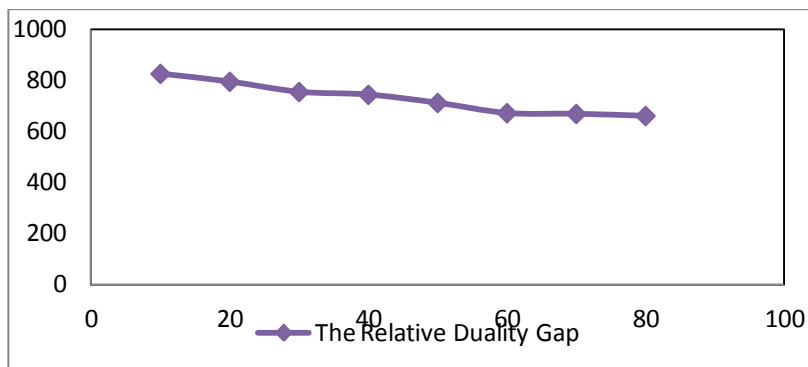


Figure 3.6 The relative duality gap observed solving the sample problem (density = 8.5, T=17, K=17)

Throughout Chapters 2 and 3, we discussed how to properly decompose MMN constraints into (sparse) blocks to be amenable to DW in order to save the computational expenses of solving such large-scale planning problems (see Table 3.4). On the hand, the decomposition method naturally lends itself to a coarse grained distributed memory parallelism based on assigning the pricing problems to different processors and solving them simultaneously. Hence, having decomposed to blocks, modern computers can take advantage of the algorithm's inherent parallelism to efficiently improve the elapsed time for MMN problems. In general, any properly decomposed problem can be provided to the DW algorithm for solving.

In a very inspiring work, Rios [76] reported computational results to motivate use of such parallel solvers; as such implementation outperforms state-of-the-art commercial solver, CPLEX 11.2, in terms of elapsed time. Applying his approach (in implementation) and ours (in decomposing) simultaneously will sufficiently demonstrate the utility of our approach in decomposing problems and his approach in parallelism. Here, we give a brief description on how this parallel implementation works and at the end we report the work done by Rios for such parallelism.

At the simplest level the master problem and pricing problems reside on separate processors; when there are more problems than processors, each processor may hold several problems which may be solved locally in sequence. Mirroring the algorithm, control is maintained by the process handling the master problem: it sends out prices to start up the slave processes; the slave processes solve the pricing problems and send the generated proposals back to the master processes[44][45][50][76][83].



The measure which is widely used to assess the performance of a parallel application in comparison to a serial application performing the same task is *speedup*. For this case in which Cplex is used, it is stated as

$$Speedup = (Time\ Cplex) / (Time\ DW\ Parallel),$$

where *Time* is either a measure of *elapsed time* or total *CPU time*. However, in most cases, the two speedups are similar. This implies that much of the speedup comes from the effectiveness of the DW algorithm at handling these LPs rather than the parallelization of the algorithm. The largest discrepancy between the elapsed time and CPU time speedups occurs with large subproblems. This is due to the fact that the parallelization in DW is exercised to solve the various subproblems in parallel versus being stuck in the serial bottleneck of solving-resolving the reduced master problem. With an understanding of DW, the results follow what might be expected [76][83].

The first parallel implementation of the decomposition method was the prototype ‘*decompar*’ by Ho et al. [44][45][50] based on the serial decomposition software ‘*decomp*’ used by Ho and Loute. *decomp*, and its underlying LP software ‘*lpml*’, were by now over fifteen years old, and much inferior to other LP software available. However, it could not be used for practical computational work. Because the maximum problem size was very restricted, with at most 10 blocks, each with 400 rows, 1000 columns and 10,000 non-zeros; and up to 99 global rows. When implemented in parallel, a new area of flexibility is opened up in the algorithm: the order in which the problems are solved on each cycle, and whether all problems are solved on every cycle.

A major implementation issue involves how the subproblems should be solved. This decision is based on several factors including *coding complexity*, *computing resources*, and *problem size*. Some problem instances of DW may have only one single subproblem, and thus do not have to be concerned with the decision of whether to solve subproblems in parallel or serially. Five strategies may be distinguished by the action taken in a typical cycle[76]:

**Basic strategy:** Solve all pricing problems, then solve the master problem.

**First pricing problem strategy:** Start solving all pricing problems, then as soon as one finishes, solve the master problem.

**First proposal strategy:** Start solving all pricing problems, then as soon as one proposal is generated, then solve the master problem.

**Instant feedback strategy:** All new information is acted on immediately all pricing problems and master constantly being solved, send out proposal from each pricing problem after every pivot, and send out prices from master after each pivot.

**Accelerated feedback strategy:** Whenever a pricing problem finishes, send out proposals generated, check for new prices, and start solving again as soon as new prices found. Whenever the master problem finishes, send out new prices, check for new proposals, and start solving again as soon as any found.

The original implementation by Ho could not be used for practical computational work and it was also tied to a no-longer available commercial LP solver from IBM. Recently, Rios described a general, parallel implementation of DW decomposition. By his approach it is hoped that future researchers in various domains will have access to a stable platform from which to begin experimentation without the need to implement the algorithm from scratch. Rios [76] detailed the implementation of the software in terms of parallelization and synchronization.

$$2.3 \leq Speedup_{[Rios's\ method]} \leq 8.5.$$

As he has shown, to minimize elapsed runtime, solving all subproblems in parallel is more efficient in the presence of multicore or cluster computers. All subproblems were launched simultaneously and each one was solved completely at each iteration of the DW algorithm as long as the subproblem had a new objective function for that iteration. All generated columns that might improve the master's objective function were added to the reduced master. When all subproblem threads had been serviced exactly once, the master thread performed additional computations while the subproblem threads block, awaiting the next iteration.

### 3.7 Summary and Concluding Remarks

This chapter addresses discrete-time dynamic min-cost network flow problem on multiperiod multiproduct distribution-production networks under the generalization of node storage or/and arc spoilage. We develop some decomposition-based approaches to solve the min-cost flow problem employing polyhedral sets hidden in the underlying distribution network. Having appropriately defined some matrices, the original problems are re-formulated into a series of structurally similar sparse LP subproblems (polyhedrals) which are utilized to develop decomposition-based techniques to decrease storage requirements. At the end, we discussed different issues and strategies on parallelization of the algorithm to improve the wall-clock time along side with computational recourses.

Note that our approach should be seen as two-phase method. In the first phase, we get the  $t$ - $q$ -node-arc incidence matrices, and the second phase is an application of DW method. Evidently, the performance of our algorithm highly depends on the first phase. A simple analysis reveals that the overall complexity of the first phase is  $O(TKnm^2)$  if a simple data structure is used to maintain the factors for each time period with respect to each product. As mentioned, although we consider time-commodity varying network parameters, all the  $t$ - $q$ -incidence matrices/transformations can be updated/run off-line and in parallel. Therefore, if the solutions of the first phase are calculated in parallel, we can expect to obtain the optimal solution for any min-cost MMN problem in a reasonable amount of time.

# Chapter 4

## 4 A Penalty-Based Scaling Algorithm for Multiperiod Multiproduct Distribution Planning Problem

This chapter addresses the minimum cost dynamic flow (MCDF) on multiperiod multiproduct distribution planning problem in the discrete-time settings with varying network parameters, and a network-based scaling algorithm is developed to obtain an optimal solution for a deterministic predefined finite planning horizon. The basis of our solution approach exploits ideas from penalty function methods in nonlinear programming and scaling algorithms in network flow theory, simultaneously.

We formulate a mathematical programming problem to find an optimal non-simultaneous shipment of commodities from production stations (sources) to consumption sites (sinks) minimizing a deterministic non-negative distribution cost function such that no capacity conditions are violated. We develop a cost-scaling-based approach to solve the MCDF problem on a multiperiod multiproduct network. Our algorithm solves the MCDF problem as a sequence of nonlinear penalty problems, each of which is constructed by relaxing some constraints and adding a term for their violation to the objective function. Each penalty problem's objective function is improved through some *augmentations* until reaching a user-defined accuracy using a network based scaling algorithm. This method keeps iteratively detecting and modifying time-commodity varying flows around cycles at each scaling phase to improve the objective function of the nonlinear penalty problem. The algorithm finds a  $\delta$ -optimal solution to the penalty problem.

Static network flow problems with multiple commodities can be posed as single commodity network problem with side constraints. Many applications of such problems lead to huge LPs that are too large to be handled by a direct application of an LP software. Hence, researchers have developed specialized adaptations of LP algorithms that exploit the special structure and the sparsity inherent in this class of problems. Exact algorithms for such problems are mostly based on DW decomposition and specializations of the simplex method. There are also a couple of heuristic algorithms to

handle side-constrained network problems. They use Lagrangean relaxation techniques to determine lower bounds, and decomposition to determine upper bounds for the value of the solution. Some approximation algorithms so far have also been developed for static network flow problems [3][11][15][20][24][48][51][54][75][78].

#### 4.1 Penalty Function Method - Transformation Approach

The mathematical programming problem (*MPP*) is to determine a vector  $x^* = (x_1^*, x_2^*, \dots, x_n^*)$  that solves the problem

$$\begin{aligned}
 (MPP) \quad & \mathbf{Min} \quad f(x_1, x_2, \dots, x_n) \\
 & g(x_1, x_2, \dots, x_n) \leq 0 \\
 & h(x_1, x_2, \dots, x_n) = 0
 \end{aligned}$$

When the problem functions  $f$ ,  $g$ , and  $h$  are all linear, the (*MPP*) is called a *linear programming problem*. If any of the functions is nonlinear the problem is called a *nonlinear programming problem*. There are other terms, such as convex, concave, separable, quadratic, and factorable, which may apply to special cases of (*MPP*). Usually the functions of (*MPP*) are required to be continuous. Much of the theory of nonlinear programming concerns the case when the functions are continuously differentiable, or twice continuously differentiable. In these instances, it is possible to prove theorems which characterize solutions to (*MPP*). These theorems in turn influence the development of algorithms for solving the mathematical programming problems. Particular results and algorithms have been obtained for *quadratic programming* where  $f(x)$  is a positive semi-definite quadratic form, and the constraints are linear [7][12][24][27][58][75].

Special methods have been developed when  $f(x)$  is a convex function and the constraints are concave/linear. The smoothness of the problem functions makes the problem well behaved, and the *convexity-concavity* assumptions assure that the feasible region is convex and, most importantly, that any local solution is also global [7][12][27].

Our penalty-scaling method is based on *Transformation Approach* in *Non-Linear Programming*. Transformation Approach is executed by defining an appropriate auxiliary function, in terms of the problem functions, to define a new objective function whose minima are unconstrained in some domain of interest. By gradually removing the effect of the constraints in the auxiliary function, a sequence (or family) of unconstrained problems with solutions converging to a solution of the original constrained problem is generated [27]. For the sake of simplicity, we proceed formally to sketch the basic idea. The problem is to find a solution  $x^*$  of

$$(MPP-1) \quad \begin{aligned} \text{Min} \quad & f(x_1, x_2, \dots, x_n) \\ & g_i(x_1, x_2, \dots, x_n) \geq 0, \quad i = 1, 2, \dots, m. \end{aligned}$$

A typical unconstrained auxiliary function may have the form

$$MPP(x, \lambda(t)) := f(x_1, x_2, \dots, x_n) + \sum_{i=1}^m \lambda_i(t) \times G(g_i(x_1, x_2, \dots, x_n)),$$

where  $t$  is a parameter,  $\lambda_i(t)$  are weighting factors,  $x = (x_1, x_2, \dots, x_n)$ , and  $G(y)$  is generally a monotonic function of  $y$  that behaves in some well-chosen manner at  $y = 0$ . Typical choices are either that  $G(y) > 0$  for  $y < 0$  and  $G(y) = 0$  for  $y \geq 0$ . The former choice usually is associated with procedures that are not concerned with constraint satisfaction except at the solution, and the latter, where constraint satisfaction is enforced throughout. When successful, the method generally proceeds computationally as follows. Select a sequence  $\{t_k; t_k \geq 0\}$ , and  $t_k \rightarrow \infty$  as  $k \rightarrow \infty$ . Compute a minimum  $n$ -vector  $x^k$  of  $MPP(x, \lambda(t))$  for  $k = 1, 2, \dots$ . Under appropriate conditions such an  $x^k$  always exists and is an unconstrained minimum of  $MPP(x, \lambda(t))$ . Usually the most desirable result is that  $x^k \rightarrow x^*$  as  $k \rightarrow \infty$ , a solution of (MPP-1). A weaker result is that  $f(x^k) \rightarrow f(x^*)$  a minimum value of the objective function. The result follows that

$$\sum_{i=1}^m \lambda_i(t) \times G(g_i(x_1, x_2, \dots, x_n)) \rightarrow 0 \quad \text{as } k \rightarrow \infty.$$

So that we will have

$$MPP(x^k, \lambda(t_k)) \rightarrow f(x^*) \quad \text{as } k \rightarrow \infty.$$

That is, the modified objective function  $MPP(x, \lambda(t))$  converges to the same minimal value, as the original objective function ( $MPP-1$ ). This means that the influence of the constraints on the modified objective or auxiliary function is gradually relinquished and finally removed in the limit.

Penalty function methods solve a constrained optimization problem by a sequence of optimization problems in which all or some of the constraints are relaxed. A penalty term for the violation of the relaxed constraints is added to the objective function. We refer to the resulting problem as the penalty problem. Thus, the penalty problems are either unconstrained or have a special structure. Detailed descriptions of penalty methods can be found in Fiacco and McCormick [27] and Luenberger [58]. In most penalty function algorithms, the optimal solutions to the sequence of penalty problems approach the optimal solution to the original problem. A penalty problem associated with ( $MPP$ ) above may be depicted as

$$(Penalty - MPP) \quad \mathbf{Min} \quad f(x) + \rho\varphi(g(x)) + \rho\Psi(h(x)).$$

Observe that  $x^* = (x_1^*, x_2^*, \dots, x_n^*)$  is the optimal solution of ( $Penalty - MPP$ ) if and only if it is the optimal solution of ( $MPP$ ) (Luenberger [58]). In practice, the optimization of ( $Penalty - MPP$ ) is difficult due to the discontinuity of the objective function on the boundary of the feasible region; this is why we consider some special functions for  $\varphi(x)$  and  $\Psi(x)$ . The parameter  $\rho$  is called the *penalty parameter*. In order to cope with the discontinuity curse, we will consider some continuous functions as  $\varphi(x)$  and  $\Psi(x)$  satisfying the following conditions:

$$\varphi(g(x)) = \begin{cases} 0 & \text{if } g(x) \leq 0 \\ M & \text{if } g(x) > 0 \end{cases} \quad \text{and} \quad \Psi(h(x)) = \begin{cases} 0 & \text{if } h(x) = 0 \\ M & \text{if } h(x) \neq 0 \end{cases},$$

where  $M$  is a constant positive number. However, in theory and implementations we often use  $\varphi(x)$  and  $\Psi(x)$  of the following forms:

$$\varphi(g(x)) = \frac{1}{2}[g(x) + |g(x)|]^a \quad \text{or} \quad \varphi(g(x)) = [\max\{0, g(x)\}]^a$$

and

$$\Psi(h(x)) = |h(x)|^b,$$

where  $a, b = 1$  or  $2$  usually.

In general, *penalty function approaches* consist of solving a sequence of penalty problems, such as (*Penalty – MPP*), with an increasing penalty parameter  $\rho$ .

**Theorem 4.1 KKT necessary conditions [7][27]**

Consider the following nonlinear program (NLP).

$$\begin{aligned} (NLP) \quad & \text{Min } f(x), \\ & g_i(x) \leq 0 \quad i = 1, 2, \dots, m, \\ & h_j(x) = 0 \quad j = 1, 2, \dots, l. \end{aligned}$$

Suppose that the objective function  $f(x)$  and the constraint functions  $g_i(x)$  and  $h_j(x)$  are continuously differentiable at a point  $x^* = (x_1^*, x_2^*, \dots, x_n^*)$ . If  $x^*$  is a local minimum that satisfies some *regularity conditions*, then there exist constants  $\mu_i$  and  $\lambda_j$ , called *KKT multipliers*, such that

$$\begin{aligned} \nabla f(x^*) + \sum_{i=1}^m \mu_i \nabla g_i(x^*) + \sum_{j=1}^l \lambda_j \nabla h_j(x^*) &= 0 \\ g_i(x^*) &\leq 0 \quad i = 1, 2, \dots, m \\ h_j(x^*) &= 0 \quad j = 1, 2, \dots, l \\ \mu_i &\geq 0 \quad i = 1, 2, \dots, m \\ \mu_i g_i(x^*) &= 0 \quad i = 1, 2, \dots, m. \end{aligned}$$

The KKT conditions (also known as KT conditions) are first order necessary conditions for a solution in continuous nonlinear programming to be optimal, provided that some *regularity conditions* are satisfied [27][58].



**Theorem 4.2 KKT conditions for an NLP with non-negativity constraints [58]**

Suppose that in the following NLP problem, the objective function  $f(x)$  and the constraint functions  $g_i(x)$  satisfying certain regularity conditions are differentiable at  $x^* = (x_1^*, x_2^*, \dots, x_n^*)$ .

(Nonnegative – NLP)

**Min**  $f(x)$

$$g_i(x) \leq 0 \quad i = 1, 2, \dots, m,$$

$$x_j \geq 0 \quad j = 1, 2, \dots, n.$$

Then,  $x^* = (x_1^*, x_2^*, \dots, x_n^*)$  can be an optimal solution for (Nonnegative – NLP) only if there exist numbers  $\mu_i$  and  $\lambda_j$  such that all the following *KKT* conditions are satisfied:

$$\frac{\partial}{\partial x_j} f(x^*) + \sum_{i=1}^m \mu_i \frac{\partial}{\partial x_j} g_i(x^*) - \lambda_j = 0 \quad j = 1, 2, \dots, n$$

$$x_j^* \left( \frac{\partial}{\partial x_j} f(x^*) + \sum_{i=1}^m \mu_i \frac{\partial}{\partial x_j} g_i(x^*) \right) = 0 \quad j = 1, 2, \dots, n$$

$$\mu_i g_i(x^*) = 0 \quad i = 1, 2, \dots, m$$

$$g_i(x^*) \leq 0 \quad i = 1, 2, \dots, m$$

$$\mu_i \geq 0 \quad i = 1, 2, \dots, m$$

$$x_j \geq 0, \lambda_j \geq 0 \quad j = 1, 2, \dots, n$$

In some cases, the necessary conditions are also sufficient for optimality. In general, the necessary conditions are not sufficient for optimality and additional information is necessary, such as the *Second Order Sufficient Conditions*.

**Theorem 4.3 Sufficient conditions [7][12][27]**

The KKT conditions are necessary to find an optimum, but not necessarily sufficient. However, these necessary conditions are sufficient for optimality if the objective function  $f(x)$  and the inequality constraints  $g_i(x)$  are continuously differentiable convex functions and the equality constraints  $h_j(x)$  are affine functions.

#### Theorem 4.4 Sufficient conditions for convex problem with linear constraints [27]

Suppose the case  $m=0$  i.e., when there are no inequality constraints in  $(NLP)$ . If  $f(x_1, x_2, \dots, x_n)$  is a convex function and each  $h_j(x_1, x_2, \dots, x_n)$  are linear function, then any point  $x^* = (x_1^*, x_2^*, \dots, x_n^*, \lambda_1^*, \lambda_2^*, \dots, \lambda_l^*)$  satisfying the following condition

$$\frac{\partial}{\partial x_1} L(x, \lambda) = \frac{\partial}{\partial x_2} L(x, \lambda) = \dots = \frac{\partial}{\partial x_n} L(x, \lambda) = \frac{\partial}{\partial \lambda_1} L(x, \lambda) = \frac{\partial}{\partial \lambda_2} L(x, \lambda) = \dots = \frac{\partial}{\partial \lambda_l} L(x, \lambda) = 0,$$

will yield an optimal solution  $x^* = (x_1^*, x_2^*, \dots, x_n^*)$  to  $(NLP)$ .

## 4.2 Network Flow - Scaling Approach

Scaling is a powerful idea that has produced algorithmic improvements to many problems in combinatorial optimization like network flow problems (see Dinic [23]). In our case, we start with the optimality conditions for the network flow problem we are examining, but instead of enforcing these conditions exactly, we generate an *approximate solution* that is permitted to violate one (or more) of the conditions by an amount  $\Delta$ . Initially, by choosing  $\Delta$  quite large we are able to find a starting solution that satisfies the relaxed optimality conditions. We then reset the parameter  $\Delta$  to  $\Delta/F$  and re-optimize so that the approximate solution now violates the optimality conditions by an amount of at most  $\Delta/F$ . We then repeat the procedure, re-optimizing again until the approximate solution violates the conditions by an amount of at most  $\Delta/F \times 1/F$ , and so on. Scaling algorithms rely on the integrality assumption. Integrality ensures applicability of the method. It requires the given numbers to be integers. This permits an easy transition from a current solution to optimum. If the given numbers are rational they must be scaled up to integers before the method is applicable. So scaling approach can find a solution with any desired accuracy [3][23].

### • Residual Network

Associated with any network flow  $G := G(V, A)$  with node set  $V$ , arc set  $A$ , and each flow  $x$ , there is another network flow, called *residual network*, giving the amount of available capacity. We define the residual network  $G(x)$  with respect to a given feasible flow  $x$  as  $G(x) = (V, A_1^r \cup A_2^r)$  where

$$A_1^r = \{(i, j) \in A : u_{ij} > x_{ij}\} \quad \text{and} \quad A_2^r = \{(i, j) \in A : 0 < x_{ij}\}.$$

The residual network  $G(x)$  is provided with *residual arc capacities* as:

$$u_{ij}^r = u_{ij} - x_{ij} \quad \text{for } (i, j) \in A_1^r \quad \text{and} \quad u_{ji}^r = x_{ij} \quad \text{for } (i, j) \in A_2^r.$$

- **$\Delta$ -Residual Network**

Given the residual network  $G(x)$  with respect to flow  $x$ , we define the  $\Delta$ -residual network  $G(x, \Delta)$  as a sub graph of  $G(x)$  containing arcs whose residual capacity is at least  $\Delta$ . Note that  $G(x, 1) = G(x)$  for a network flow with integral capacities.

- **Augmenting Path and Cycle**

A path constructed by repeatedly finding a path of positive capacity from a source  $s$  to a sink  $t$  and then adding it to the flow. Equivalently, an augmenting path is a path  $i_1, i_2, i_3, \dots, i_n$  in the residual network, where  $i_1 = s$ ,  $i_n = t$  and  $u_{ij}^r > 0$  for any  $i = i_k$  and  $j = i_{k+1}$ . A cycle  $W$  (not necessarily directed) in  $G$  is called an augmenting cycle with respect to the flow  $x$  if by augmenting a positive amount of flow  $f(W)$  around the cycle, the flow remains feasible. The augmentation increases the flow on forward arcs in the cycle  $W$  and decreases the flow on backward arcs in the cycle. In other words, an augmenting cycle is a directed cycle  $i_1, i_2, i_3, \dots, i_n$  in the residual network, where  $i_1 = s$ ,  $i_n = s$ , and  $u_{ij}^r > 0$  for any  $i = i_k$  and  $j = i_{k+1}$ .

- **The Cost and Mean Cost of a Cycle**

We next extend the notations of  $\delta_{ij}(W)$  and  $c(W)$  for cycles that are not necessarily directed. We define  $\delta_{ij}(W)$  as

$$\begin{aligned} \delta_{ij}(W) &= 1 && \text{if arc } (i, j) \text{ is a forward arc in the cycle } W, \\ \delta_{ij}(W) &= -1 && \text{if arc } (i, j) \text{ is a backward arc in the cycle } W, \\ \delta_{ij}(W) &= 0 && \text{if arc } (i, j) \text{ is not in the cycle } W. \end{aligned}$$

Notice that in terms of residual networks, each augmenting cycle  $W$  with respect to a flow  $x$  corresponds to a directed cycle  $W$  in  $G(x)$  and vice versa. We define the *cost of an augmenting cycle*  $W$  as

$$c(W) = \sum_{(i,j) \in W} c_{ij} \delta_{ij}(W).$$

The cost of an augmenting cycle represents the change in the cost of a feasible solution if we augment one unit of flow along the cycle. Therefore, the change in flow cost for augmenting  $f(W)$  units along the cycle  $W$  is  $c(W)f(W)$ . We also define the *mean cost* of a cycle as its cost divided by the number of arcs it contains. A *minimum mean cycle* is a cycle whose mean cost is as small as possible.

- **Reduced Cost of an Arc**

In many network flow algorithms, we measure the cost of an arc related to some cost associated with the nodes. These costs are typical intermediate data of the algorithm. Suppose that  $G(V, A)$  is a directed network flow with some real numbers  $\pi_i$  associated with each node  $i \in V$  as the *potential* of node  $i$ . For a given set of node potentials  $\pi_i$ , we define the *reduced cost* as

$$c_{ij}^{\pi} := c_{ij} - \pi_i + \pi_j \quad \forall (i, j) \in A.$$

These reduced costs are applicable to the residual network as well as the original network. That is

$$c_{ij}^{\pi} := c_{ij} - \pi_i + \pi_j \quad \forall (i, j) \in (A_1^r \cup A_2^r).$$

**Remark 4.1** In many minimum cost flow algorithms, the reduced cost replaces the cost, especially in the residual network. Thus, it is crucial to understand the relationship between  $c^{\pi}$  and  $c$  with respect to the minimum flow problem. We define the reduced costs in the residual network just as we did the costs, but now using  $c_{ij}^{\pi}$  in place of  $c_{ij}$ .

**Theorem 4.5 Shortest path optimality conditions [3]**

Suppose that  $G := G(V, A)$  is a directed network flow. Let us define the *reduced arc length*  $\pi_{ij}^d$  of an arc  $(i, j) \in A$  with respect to the *distance label function*  $d(\cdot)$  as  $\pi_{ij}^d := c_{ij} + d_i - d_j$ . And suppose that  $d(\cdot)$  is such that for every node  $i$ ,  $d(i)$  denotes the length of some directed path from the source node to node  $i$ . Then the numbers  $d(i)$  represent shortest path distances if and only if they satisfy the following shortest path optimality conditions:

$$\pi_{ij}^d := c_{ij} + d_i - d_j \geq 0 \quad \forall (i, j) \in A.$$

Various uses of the shortest path optimality conditions suggest that similar sets of conditions might be valuable for designing and analyzing algorithms for the minimum cost flow problem, and also they yield three different (but equivalent) optimality conditions for minimum cost flow problem. All these optimality conditions have an intuitive network interpretation and are rather direct extensions of their shortest path counterparts. We will consider three optimality conditions: (1) *negative cycle optimality conditions*, (2) *reduced cost optimality conditions*, and (3) *complementary slackness optimality conditions*.

**Theorem 4.6 Negative cycle optimality conditions [3]**

Suppose that  $G := G(V, A)$  is a directed network flow. A feasible solution  $x$  is an optimal solution of the minimum cost flow problem if and only if the residual network  $G(x)$  contains no negative cost directed cycle.

**Theorem 4.7 Reduced cost optimality conditions [3]**

Suppose that  $G := G(V, A)$  is a directed network flow. A feasible solution  $x$  is an optimal solution of the minimum cost flow problem if and only if some set of node potentials  $\pi_i$  satisfy the following reduced cost optimality conditions:

$$c_{ij}^\pi := c_{ij} - \pi_i + \pi_j \geq 0 \quad \forall (i, j) \in (A_1^r \cup A_2^r).$$

We know that a flow is optimal if there is no negative cost cycle in the residual network [3]. If there is no negative cost cycle in the residual network, it means that the shortest path distance is well defined with respect to the cost function. On the contrary, if there is a negative cost cycle, the shortest path distance is not well-defined. We will express the ‘*well-definedness*’ of the shortest path in terms of reduced costs, thanks to the shortest path optimality conditions. This gives rise to the following theorem.

**Theorem 4.8 Complementary slackness optimality conditions [3]**

Suppose that  $G := G(V, A)$  is a directed network flow. A feasible solution  $x$  is an optimal solution of the minimum cost flow problem if and only if for some set of node potentials  $\pi_i$ , the reduced costs and flow values satisfy the following complementary slackness optimality conditions:

$$\text{If } c_{ij}^\pi > 0, \text{ then } x_{ij} = 0 \quad \forall (i, j) \in A,$$

$$\text{If } c_{ij}^\pi < 0, \text{ then } x_{ij} = u_{ij} \quad \forall (i, j) \in A,$$

$$\text{If } 0 < x_{ij} < u_{ij}, \text{ then } c_{ij}^\pi = 0 \quad \forall (i, j) \in A.$$

**4.3 Min-Cost Multiperiod Multiproduct Distribution Problem**

As mentioned, in the min-cost multiperiod multiproduct distribution networks we aim to find a routing plan to non-simultaneously ship the products from source nodes to sink nodes through a distribution network without exceeding the time-varying, time-commodity varying, and horizon capacities at the minimal cost during the finite planning horizon. The products may either be differentiated by their physical characteristics or simply by their origin-destination pairs. However, arc capacities bind different products together. In fact, the essential issue addressed by the multiperiod multiproduct network problem is the allocation of the capacities of each arc to the individual products in a way that minimizes overall flow costs.

Given distribution network  $G = (V, A, T, K)$  where  $V$  is the set of sources and sinks,  $A$  is the set of all possible connections between sites (arcs),  $K = \{1, 2, \dots, K\}$  is the set of products, and  $T$  is the time horizon, the MCDF on MMN can be expressed as:

$$(CTMCDF) \quad \mathbf{Min}_{x_{ijq}(t)} \sum_{q \in \mathbf{K}} \sum_{(i,j) \in \mathbf{A}} \int_0^T c_{ijq}(t) x_{ijq}(t) dt, \quad (4.1)$$

$$\sum_j \int_\alpha^\beta x_{ijq}(t) dt - \sum_j \int_\alpha^\beta x_{jii}(t) dt = \int_\alpha^\beta v_{iq}(t) dt \quad \forall i \in \mathbf{V}, \forall q \in \mathbf{K}, \forall \alpha, \beta \in [0, T], \quad (4.2)$$

$$\sum_{q \in \mathbf{K}} \int_0^T x_{ijq}(t) dt \leq u_{ij} \quad \forall (i, j) \in \mathbf{A}, \quad (4.3)$$

$$\sum_{q \in \mathbf{K}} x_{ijq}(t) \leq u_{ij}(t) \quad \forall (i, j) \in \mathbf{A}, \forall t \in [0, T], \quad (4.4)$$

$$0 \leq l_{ijq}(t) \leq x_{ijq}(t) \leq u_{ijq}(t) \quad \forall (i, j) \in \mathbf{A}, \forall q \in \mathbf{K}, \forall t \in [0, T], \quad (4.5)$$

$$x_{ijq}(t) = 0 \quad \forall (i, j) \in \mathbf{A}, \forall q \in \mathbf{K}, \forall t > T. \quad (4.6)$$

In this setting,  $x_{ijq}(t)$  describes the dynamic flow decision variable as the vector of flow amount of commodity  $q$  entering arc  $(i, j)$  at time moment  $t$ , and  $c_{ijq} : [0, T] \rightarrow \mathbb{R}^+$  is the non-negative time-varying cost function with respect to product  $q$ , and  $v_{iq}(t)$  denotes the pre-defined supply/demand capacities at node  $i$  over time. Constraint (4.2) involves the flow conservation constraints for each commodity. We refer to (4.3) as *horizon capacity* constraints. Horizon capacity of an arc limits the amount of total flow (of all commodities) on the arc throughout the entire horizon. Constraint (4.4) represents the maximum possible amount of total flow that can enter  $(i, j)$  at time  $t$ : it is referred to as the *moment capacity* constraint. Constraint (4.5) is the time-commodity varying capacity constraint for each commodity at each moment. The domain of decision variables prescribed in (4.6) also emphasizes that commodities can flow on the network only until the end of pre-specified time horizon.

#### 4.3.1 Canceling the Time-Commodity Varying Lower Bounds

The dynamic flow  $x(t)$  with time horizon  $T$  is a feasible flow of the (CTMCDF) problem if it satisfies the conditions (4.2)-(4.6). By replacing  $x_{ijq}(t) = x'_{ijq}(t) + l_{ijq}(t)$  for every  $(i, j) \in \mathbf{A}$ , we obtain the following equivalent constraints:

$$\sum_j \int_{\alpha}^{\beta} x'_{ijq}(t) dt - \sum_j \int_{\alpha}^{\beta} x'_{jiq}(t) dt = \int_{\alpha}^{\beta} v_{iq}(t) dt + \int_{\alpha}^{\beta} v'_{iq}(t) dt \quad \forall i \in V, \forall q \in K, \forall \alpha, \beta \in [0, T], \quad (4.7)$$

$$\sum_{q \in K} x'_{ijq}(t) \leq u'_{ij}(t) \quad \forall (i, j) \in A, \forall t \in [0, T], \quad (4.8)$$

$$\sum_{q \in K_0} \int_0^T x'_{ijq}(t) dt \leq u'_{ij} \quad \forall (i, j) \in A, \quad (4.9)$$

$$0 \leq x'_{ijq}(t) \leq u'_{ijq}(t) \quad \forall (i, j) \in A, \forall q \in K, \forall t \in [0, T], \quad (4.10)$$

Where

$$v'_{iq}(t) = \sum_j l_{jiq}(t) - \sum_j l_{ijq}(t), \quad (4.11)$$

is considered as supplies/demands of node  $i$  at time  $t \in [0, T]$  for product  $q$ . Observe that

$$u'_{ijq}(t) = u_{ijq}(t) - l_{ijq}(t), \quad (4.12)$$

$$u'_{ij} = u_{ij} - \sum_{q \in K_0} \int_0^T l_{ijq}(t) dt, \quad (4.13)$$

$$u'_{ij}(t) = u_{ij}(t) - \sum_{q \in K} l_{ijq}(t). \quad (4.14)$$

Therefore, the network problem formulated in (4.1)-(4.6) is equivalent to the problem in the *transformed network* with parameters mentioned in (4.11)-(4.14). That is if  $x'_{ijq}(t)$  is a multiperiod multiproduct feasible flow in the transformed network, then  $x_{ijq}(t) = x'_{ijq}(t) + l_{ijq}(t)$  is a multiperiod multiproduct feasible solution of the original network.

Using the discussion above, we can formulate any (CTMCDF) problem on a MMN with time varying lower bounds as a (CTMCDF) problem without lower bounds. Henceforth, we only focus on the (CTMCDF) problems without lower bounds. Another approach can also be used to remove the time-commodity varying upper bounds (Ahuja et al. [3]).



### 4.3.2 Discrete Time Multiperiod Multiproduct Distribution Problem

The problem formulation in (4.1)-(4.6) represents MCDF in a continuous-time setting. By using the *natural discretization transformation* mentioned in Sections 2.2 and 3.1, we can alternatively represent MCDF by discrete time increments.

We also observe that a (*CTMCDF*) problem with multiple sources and multiple sinks for each product can be easily transformed to a single-source, single-sink (*CTMCDF*) problem. One way of transformation is by adding an artificial source node (super-source) and an artificial destination node (super-sink) for each product. The artificial super-source is connected with each origin of that product with a cost of zero (for each time period), and the artificial super-sink is connected with each source of that product with cost of zero (for each time period). Any flow in the single-source single-sink network corresponds to a flow in the multi-source multi-sink network, and vice versa. Hence, we can only consider single-source single-sink multiproduct multiperiod network flow problems. To better illustrate this transformation, we provide Figure 4.1.

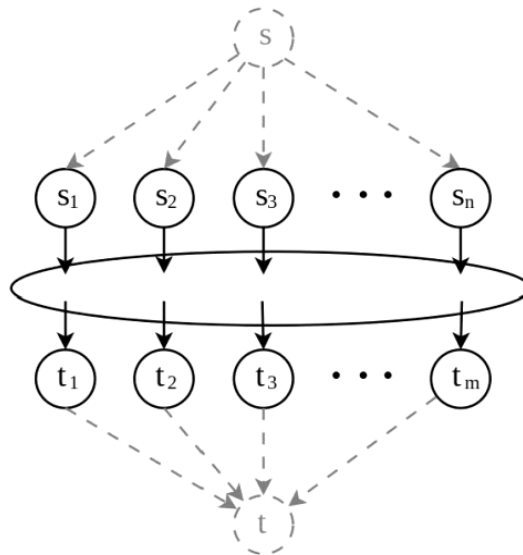


Figure 4.1 Transformation of a multi-source multi-sink single-product multi-period network problem into a single-source single-sink single-product multi-period network problem

Therefore, a discrete multiproduct multiperiod feasible flow is a non-negative function  $x = \{x_{ijq}^t\} : A \times N \times K \rightarrow \mathbb{R}^+$  satisfying (4.15)-(4.18), and the discrete-time minimum cost dynamic flow problem becomes as

(DTMCDF)

$$\mathbf{Min}_{x_{ijq}^t} \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} \sum_{(i,j) \in \mathbf{A}} c_{ijq}^t x_{ijq}^t ,$$

$$\sum_j x_{ijq}^t - \sum_j x_{jiq}^t = v_{iq}^t \quad \forall i \in \mathbf{V}, \forall q \in \mathbf{K}, \forall t \in \mathbf{N}, \quad (4.15)$$

$$\sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t \leq u_{ij} \quad \forall (i, j) \in \mathbf{A}, \quad (4.16)$$

$$\sum_{q \in \mathbf{K}} x_{ijq}^t \leq u_{ij}^t \quad \forall (i, j) \in \mathbf{A}, \forall t \in \mathbf{N}, \quad (4.17)$$

$$0 \leq x_{ijq}^t \quad \forall (i, j) \in \mathbf{A}, \forall t \in \mathbf{N}, \forall q \in \mathbf{K}. \quad (4.18)$$

(4.15)-(4.18) represent the flow feasibility conditions in a MMN in discrete time settings. For each commodity  $q$  there is a required time-varying flow of  $v_q^t$  units at time period  $t$  from its source node  $s_q$  to its sink node  $t_q$ . The cost of a unit  $q$ -flow at period  $t$  on arc  $(i, j)$  is  $c_{ijq}^t$  and the amount of flow is denoted as  $x_{ijq}^t$ . Let  $v_{iq}^t$  be the flow balance of commodity  $q$  at node  $i$  at period  $t$ . Thus, we conclude that

$$v_{iq}^t = \begin{cases} v_q^t & \text{if } i = s_q \\ -v_q^t & \text{if } i = t_q \\ 0 & \text{if } i \neq s_q, t_q \end{cases} .$$

#### 4.4 The Penalty-Based Scaling Approach for MCDF

We introduce a penalty-based scaling algorithm for the (DTMCDF) problem. The efficiency of the algorithm is induced by using the scaling approach and by exploiting the network structure of the problem. The scaling network-based algorithm finds a  $\delta$ -optimal solution to the penalty problem as discussed earlier while this solution is optimal to another problem in which part of the data is modified by at most  $\delta$ . We develop a penalty problem MCDF( $\rho$ ) of (DTMCDF) problem by moving the horizon and period capacity constraints to the objective function and choosing a quadratic penalty function for violation. As discussed, a sequence of solutions to the penalty problem MCDF( $\rho$ ) with increasing penalty parameter converges to the optimal solution of (DTMCDF). The quadratic penalty problem is formulated as

$$\begin{aligned}
& \text{MCDF}(\rho) \\
& \Downarrow \\
\mathbf{Min} \quad f_\rho(x) &= \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} \sum_{(i,j)} c_{ijq}^t x_{ijq}^t + (1/4) \sum_{(i,j) \in \mathbf{A}} \rho \left( \left( \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \right) + \left| \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \right| \right)^2 \\
& \quad + (1/4) \sum_{t \in \mathbf{N}} \sum_{(i,j) \in \mathbf{A}} \rho \left( \left( \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t \right) + \left| \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t \right| \right)^2, \\
& \quad \sum_j x_{ijq}^t - \sum_j x_{jiq}^t = v_{iq}^t \quad \forall i \in \mathbf{V}, q \in \mathbf{K}, t \in \mathbf{N}, \\
& \quad 0 \leq x_{ijq}^t \quad \forall (i,j) \in \mathbf{A}, q \in \mathbf{K}, t \in \mathbf{N}.
\end{aligned}$$

By using the definition of absolute value function  $|\cdot|$ , we can modify the objective function in  $\text{MCDF}(\rho)$  by defining the following *excess functions*.

$$\begin{aligned}
e_{ij}(x) &= \max \left\{ 0, \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \right\} \quad \forall (i,j) \in \mathbf{A}, \\
e_{ij}^t(x) &= \max \left\{ 0, \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t \right\} \quad \forall (i,j) \in \mathbf{A}, t \in \mathbf{N}. \tag{4.19}
\end{aligned}$$

In fact,  $e_{ij}(x)$  and  $e_{ij}^t(x)$  are the amount by which the total flow on arc  $(i,j)$  exceeds its horizon and period capacity (with respect to period  $t$ ). The resulting penalty problem  $\text{MCDF}(\rho)$  is

$$\begin{aligned}
& \text{MCDF}(\rho) \\
\mathbf{Min} \quad f_\rho(x) &= \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} \sum_{(i,j) \in \mathbf{A}} c_{ijq}^t x_{ijq}^t + \rho \left( \sum_{(i,j) \in \mathbf{A}} (e_{ij}(x))^2 + \sum_{t \in \mathbf{N}} \sum_{(i,j) \in \mathbf{A}} (e_{ij}^t(x))^2 \right), \tag{4.20}
\end{aligned}$$

$$\sum_j x_{ijq}^t - \sum_j x_{jiq}^t = v_{iq}^t \quad \forall i \in \mathbf{V}, q \in \mathbf{K}, t \in \mathbf{N}, \tag{4.21}$$

$$0 \leq x_{ijq}^t \quad \forall (i,j) \in \mathbf{A}, q \in \mathbf{K}, t \in \mathbf{N}. \tag{4.22}$$

Generally, any penalty function other than the quadratic may be used. However, we prefer the quadratic function because (1) it assigns large penalty to a large excess in a nonlinear fashion, and (2) the penalty objective function becomes convex, and its derivative is linear. The objective function, however, is non-separable and nonlinear.

Hence, we eliminate the complicating constraints and we decomposed  $\text{MCDF}(\rho)$  into some *single-product flow problems*, but introduce convex and non-separable terms into the objective function. These features simplify the analysis and the derivation of the theoretical properties.

In order to determine the cycles of interest (negative cost cycles), we build an auxiliary residual network. A  $\delta$ -shift for product  $q$  at time  $t$  is found by detecting a *negative cost cycle* on the residual network which corresponds to that product at that time. Here, at each scaling phase we use an associated  $\delta$ -residual network  $G_q^t(x, \delta)$  with respect to each  $t$  and  $q$ , and flow  $x$ . A negative cost cycle on a  $\delta$ -residual network means that shifting  $\delta$  units around this cycle results with a negative net change of the value of the penalty objective function  $f_\rho(x)$ . To shift  $\delta$  units around a cycle  $W_q^t$  is to increase the flow of commodity  $q$  at period  $t$  by  $\delta$  units on original arcs, which correspond to forward arcs in  $G_q^t(x, \delta)$  that in the cycle, and to decrease the flow of commodity  $q$  at period  $t$  by  $\delta$  units on original arcs, which correspond to backward arcs in  $G_q^t(x, \delta)$  that in the cycle.

- **The Cost of a  $\delta$ -Flow Around a Cycle**

The cost of a  $\delta$ -flow around cycle  $W_q^t$ , which is not necessarily directed, is the net change in the objective function of  $\text{MCDF}(\rho)$  problem by sending  $\delta$  units of  $q$ -flow around it at period  $t$ . We define the cost of cycle  $W_q^t$  with respect to product  $q$  at time  $t$  and with respect to parameter  $\rho$  as

$$c(W_q^t) = f_\rho(x + x(W_q^t, \delta)) - f_\rho(x), \quad (4.23)$$

where  $x(W_q^t, \delta)$  represents a  $\delta$  units of  $t$ - $q$ -flow around cycle  $W_q^t$ .

- **$\delta$ -Residual Network**

For a given flow  $x$  and scaling parameter  $\delta$ , we define  $\delta$ -residual network  $G_q^t(x, \delta) = (V, A_1^r \cup A_2^r)$  for each  $q$  and  $t$  as

$$A_1^r = \{(i, j) : (i, j) \in A, 0 \leq x_{ij}^t\} \quad \text{and} \quad A_2^r = \{(j, i) : (i, j) \in A, \delta \leq x_{ij}^t\}.$$

Using the  $\text{MCDF}(\rho)$  objective function, we define the cost of each arc in the  $t$ - $q$ - residual network as following:

$\mathbf{C}_{ijq}^t$ : Cost of each forward arc  $(i, j)$  in the residual network for time period  $t$  and product  $q$  will be equal to the net change in  $f_\rho(x)$  obtained by increasing the  $q$ -flow on arc  $(i, j)$  by  $\delta$  units at period  $t$ .

$\mathbf{C}_{jiq}^t$ : Cost of each backward arc  $(j, i)$  in the residual network for time period  $t$  and product  $q$  is defined as the net change in  $f_\rho(x)$  obtained by decreasing the  $q$ -flow on arc  $(i, j)$  by  $\delta$  units at period  $t$ .

**Observation 4.1** The original arcs of the network form the forward arcs of the residual network. Moreover, if  $(i, j)$  is a forward arc in the associated  $\delta$ -residual network  $G_q^t(x, \delta)$  then the cost of arc  $(i, j)$  in the residual network for time period  $t$  and product  $q$  will be obtained by

$$\mathbf{C}_{ijq}^t = \delta c_{ijq}^t + 2\rho\delta(\delta + e_{ij}(x) + e_{ij}^t(x)) \quad \forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \geq 0 \quad \text{and} \quad \sum_{q \in \mathbf{K}} x_{ijq}^t \geq u_{ij}^t$$

$$\mathbf{C}_{ijq}^t = \delta c_{ijq}^t + \rho\delta(\delta + 2e_{ij}(x)) \quad \forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \geq 0 \quad \text{and} \quad \sum_{q \in \mathbf{K}} x_{ijq}^t \leq u_{ij}^t - \delta$$

$$\mathbf{C}_{ijq}^t = \delta c_{ijq}^t + \rho\delta(\delta + 2e_{ij}(x)) + \rho \left( \delta + \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t \right)^2$$

$$\forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \geq 0 \quad \text{and} \quad -\delta < \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t < 0$$

$$\mathbf{C}_{ijq}^t = \delta c_{ijq}^t + \rho\delta(\delta + 2e_{ij}^t(x)) \quad \forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t \leq u_{ij} - \delta \quad \text{and} \quad \sum_{q \in \mathbf{K}} x_{ijq}^t \geq u_{ij}^t$$

$$\mathbf{C}_{ijq}^t = \delta c_{ijq}^t \quad \forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t \leq u_{ij} - \delta \quad \text{and} \quad \sum_{q \in \mathbf{K}} x_{ijq}^t \leq u_{ij}^t - \delta$$

$$\mathbf{C}_{ijq}^t = \delta c_{ijq}^t + \rho \left( \delta + \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t \right)^2 \quad \forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t \leq u_{ij} - \delta \quad \text{and} \quad -\delta < \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t < 0$$

$$\mathbf{C}_{ijq}^t = \delta c_{ijq}^t + \rho \delta (\delta + 2e_{ij}^t(x)) + \rho \left( \delta + \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \right)^2$$

$$\forall (i, j) \in \mathbf{A} : -\delta < \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} < 0 \text{ and } \sum_{q \in \mathbf{K}} x_{ijq}^t \geq u_{ij}^t$$

$$\mathbf{C}_{ijq}^t = \delta c_{ijq}^t + \rho \left( \delta + \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \right)^2$$

$$\forall (i, j) \in \mathbf{A} : -\delta < \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} < 0 \text{ and } \sum_{q \in \mathbf{K}} x_{ijq}^t \leq u_{ij}^t - \delta$$

$$\mathbf{C}_{ijq}^t = \delta c_{ijq}^t + \rho \left( \left( \delta + \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \right)^2 + \left( \delta + \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t \right)^2 \right)$$

$$\forall (i, j) \in \mathbf{A} : -\delta < \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} < 0 \text{ and } -\delta < \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t < 0$$

**Observation 4.2** A backward arc will be included in the  $\delta$ -residual network of time  $t$  for product  $q$  if and only if we can decrease the flow of product  $q$  at period  $t$ . Therefore, a backward arc may be used in a negative cost cycle only if the flow of product  $q$  on its corresponding forward arc is at least  $\delta$ . Moreover, if  $(i, j)$  is a backward arc in the associated  $\delta$ -residual network  $G_q^t(x, \delta)$ , then the cost of arc  $(i, j)$  in the residual network for period  $t$  and product  $q$  will be obtained by

$$\mathbf{C}_{jiq}^t = -\delta c_{ijq}^t + 2\rho \delta (\delta - e_{ij}(x) - e_{ij}^t(x)) \quad \forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \geq \delta \text{ and } \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t \geq \delta$$

$$\mathbf{C}_{jiq}^t = -\delta c_{ijq}^t + \rho \delta (\delta - 2e_{ij}(x)) \quad \forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \geq \delta \text{ and } \sum_{q \in \mathbf{K}} x_{ijq}^t \leq u_{ij}^t$$

$$\mathbf{C}_{jiq}^t = -\delta c_{ijq}^t + \rho (\delta (\delta - 2e_{ij}(x)) - e_{ij}^t(x)^2)$$

$$\forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \geq \delta \text{ and } 0 < \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t < \delta$$

$$\mathbf{C}_{jiq}^t = -\delta c_{ijq}^t + \rho \delta (\delta - 2e_{ij}^t(x)) \quad \forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \leq 0 \text{ and } \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t \geq \delta$$

$$\mathbf{C}_{jiq}^t = -\delta c_{ijq}^t \quad \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} \leq 0 \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t \leq u_{ij}^t$$

$$\mathbf{C}_{jiq}^t = -\delta c_{ijq}^t - \rho e_{ij}^t(x)^2 \quad \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} \leq 0 \quad \text{and} \quad 0 < \sum_{q \in K} x_{ijq}^t - u_{ij}^t < \delta$$

$$\mathbf{C}_{jiq}^t = -\delta c_{ijq}^t + \rho (\delta (\delta - 2e_{ij}^t(x)) - e_{ij}^t(x)^2)$$

$$\forall (i, j) \in A : 0 < \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} < \delta \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t - u_{ij}^t \geq \delta$$

$$\mathbf{C}_{jiq}^t = -\delta c_{ijq}^t - \rho e_{ij}^t(x)^2 \quad \forall (i, j) \in A : 0 < \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} < \delta \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t \leq u_{ij}^t$$

$$\mathbf{C}_{jiq}^t = -\delta c_{ijq}^t - \rho (e_{ij}^t(x)^2 + e_{ij}^t(x)^2) \quad \forall (i, j) \in A : 0 < \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} < \delta \quad \text{and} \quad 0 < \sum_{q \in K} x_{ijq}^t - u_{ij}^t < \delta.$$

The algorithm starts with an initial value for the *scaling parameter*  $\delta$ . At each *scaling phase*,  $\delta$  has a fixed value. Focusing on one product at a time period, we try to improve the objective function of  $\text{MCDF}(\rho)$  by modifying some of the flows of this product by exactly  $\delta$  units. A flow of that product on an arc may either increase by  $\delta$ , decrease by  $\delta$ , or remain unchanged. We call such a modification a  *$\delta$ -shift*. When there is no  $\delta$ -shift which leads to an improvement for any product at any time, the value of  $\delta$  is decreased. We have chosen to decrease  $\delta$  by a factor of 2 at the end of each scaling phase (Ahuja et al. [3]). So, our algorithm, MMNF algorithm, can be outlined as follows.

### Algorithm MMNF

**begin**

Determine an initial feasible dynamic solution  $x$  (satisfying only flow requirements);

Set values for the parameters  $R, \varepsilon, \rho_u$ ;

Choose initial values for  $\delta$  and  $\rho$ ;

**while** the algorithm's termination criteria are not met **do**

**begin**

IMPROVE-APPROXIMATION ( $x, \delta, \rho$ );

$\delta := \delta/2$ ;  $\rho := \min\{R\rho, \rho_u\}$ ;

**end**

**end**

Let  $\text{SEARCH}(q, t)$  be a procedure that searches for a negative cost cycle on  $G_q^t(x, \delta)$ , and let  $\text{AUGMENT}(w_q^t, \delta)$  be a procedure which shifts  $\delta$  units of  $q$ -flow around a cycle  $w_q^t$ . Then, the procedure  $\text{IMPROVE-APPROXIMATION}$  may be described as follows.

**Procedure IMPROVE-APPROXIMATION**

```

begin
  LIST: = K;
  while LIST  $\neq$  null do
    begin
      select a commodity  $q$  from LIST;
       $w_q^t := \text{SEARCH}(q, t)$  for any  $t$ ;
      if  $w_q^t = \text{null}$  then delete  $q$  from LIST
      else
        begin
          AUGMENT( $w_q^t, \delta$ );
          LIST: = K;
        end
      end
    end
  end

```

Our network-based approach for  $\text{MCDF}(\rho)$  can be viewed as a scaling algorithm in which the step size at each phase is fixed at a value of  $\delta$ . The algorithm will be trying to find a direction of interest such that by modifying the flow by  $\delta$  units in this direction the penalty objective function  $f_\rho(x)$  decreases. When it is unable to find such an improving direction with step size  $\delta$ , it decreases the step size to  $\delta/2$ . The accuracy of the algorithm highly depends on the value of  $\delta\rho$  and how we want to increase the penalty cost [75]. Hence, we consider an upper bound for  $\delta\rho$  (say  $\varepsilon$ ) and we set a fixed value for increasing the penalty cost (say  $R$ ) for use in the termination rule.

For setting the initial flow satisfying (only) the supply/demand constraints one may, for example, satisfy the requirement of each product by sending each demand on the shortest path from its origin to its destination with respect to the flow costs. Other initial solutions which satisfy the supply/demand constraints may be used as well. As a very common choice in scaling algorithms, we may let the initial value of  $\delta$  be the size of the largest demand, i.e.,  $\delta_0 := v_{\max}$  where  $v_{\max} = \max\{v_q^t : t, q\}$  or  $\delta_0 := 2^{\lfloor \log(v_{\max}) \rfloor}$ . The best initial penalty parameter value and its modification rate,  $R$ , may be empirically



determined [75][78]. However, we believe that  $\rho_0 = 0.5-1.9$  and  $R=1.3-1.9$  are some good choices.

Observe that  $\delta$  and  $\rho$  are modified simultaneously. For theoretical purposes, which are already discussed, we need  $\delta\rho$  to decrease in each phase. Eventually we would like  $\delta\rho \rightarrow 0$ . Since  $\delta$  decreases by a factor of 2 at each phase, we require that  $R < 2$ . Moreover,  $\varepsilon$  is chosen small enough to assure that sufficient number of phases are performed. Thus, it assures that  $\delta$  is sufficiently small and  $\rho$  is sufficiently large at the end of the algorithm [20][75]. In practice, to help maintain numerical stability, we might want to preclude large values of the penalty parameter. In addition, the user may want to control the final value of  $\rho$  in order to help maintain numerical stability or to ensure faster termination of the algorithm. For these reasons, we bound  $\rho$  from above by a user-defined value  $\rho_u$ . The algorithm terminates when  $\delta\rho$  is sufficiently small (or the duality gap is sufficiently small) for a penalty problem with a sufficiently large penalty parameter. The value of  $\delta\rho$  indicates how close the solution is to optimality. The user defined lower bounds on  $\delta\rho$  provides the user the option to compromise a bit on the quality of the solution in order to have the algorithm run faster [75][78].

At each step, we need to pick a product  $q$  for some  $t$  and search for an improving cycle. To this aim, we calculate the costs of the arcs in the  $\delta$ -residual network  $G_q^t(x, \delta)$  of product  $q$  for any  $t$ . As mentioned in Observations 4.1 and 4.2, the cost of each arc is the costs of modifying the flow of product  $q$  on that arc by  $\delta$ . Then, determine whether there is a negative cost cycle in  $G_q^t(x, \delta)$ . If a negative cost cycle for commodity  $q$  is detected, we shift  $\delta$  units of flow around it. This means: increasing the flow of commodity  $q$  by  $\delta$  on original arcs which correspond to forward arcs in  $G_q^t(x, \delta)$  and belong to the cycle, and decreasing the flow of commodity  $q$  by  $\delta$  on original arcs which correspond to backward arcs in  $G_q^t(x, \delta)$  and belong to the cycle. If all the commodities have been scanned and no negative cost cycle has been found, we terminate the current scaling phase and we modify the values of  $\delta$  and  $\rho$  as  $\delta := \delta/2$  and  $\rho := R\rho$  (if  $R\rho < \rho_u$ ).

#### 4.4.1 Some Notes on the Theoretical Properties

We present some theoretical properties of our algorithm regarding optimality and convergence. These properties lead to some bounds on the number of basic operations which the algorithm performs. Typically, we introduce and prove general properties first, and then we deduce the properties applied to algorithm. Some background on computational time bounds and complexity analysis may be found in Ahuja et al. [3][20]. The proofs in this section are based on the ideas of  $(\delta, \varepsilon)$ -optimality and  $(0, \varepsilon)$ -optimality.

- **Approximate Optimality or  $\varepsilon$ -optimality**

A flow (or pseudo flow)  $\mathbf{x}$  is said to be  $\varepsilon$ -optimal for some  $\varepsilon > 0$  if for some node potentials  $\boldsymbol{\pi}$ , the pair  $(\mathbf{x}, \boldsymbol{\pi})$  satisfies the following  $\varepsilon$ -optimality conditions:

$$c_{ij}^{\pi} > \varepsilon \quad \Rightarrow \quad x_{ij} = 0, \quad (4.24)$$

$$-\varepsilon < c_{ij}^{\pi} < \varepsilon \quad \Rightarrow \quad 0 \leq x_{ij} \leq u_{ij}, \quad (4.25)$$

$$c_{ij}^{\pi} < -\varepsilon \quad \Rightarrow \quad x_{ij} = u_{ij}. \quad (4.26)$$

These conditions are relaxations of the (exact) complementary slackness optimality conditions we already discussed; these conditions reduce to complementary slackness optimality conditions when  $\varepsilon = 0$ . The *exact* optimality conditions imply that any combination of  $(x_{ij}, c_{ij}^{\pi})$  lying on the thick lines shown in Figure 4.2 is optimal. The  $\varepsilon$ -optimality conditions (4.24)-(4.26) imply that any combination of  $(x_{ij}, c_{ij}^{\pi})$  lying on the thick lines or in the hatched region in Figure 4.3 is  $\varepsilon$ -optimal [3].

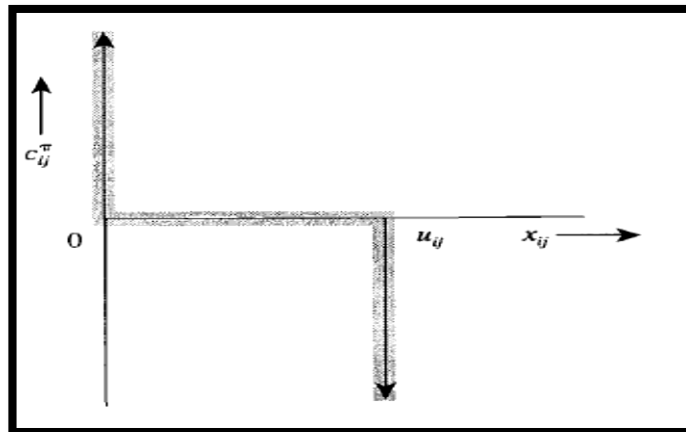


Figure 4.2 Exact optimality for  $(i, j)$

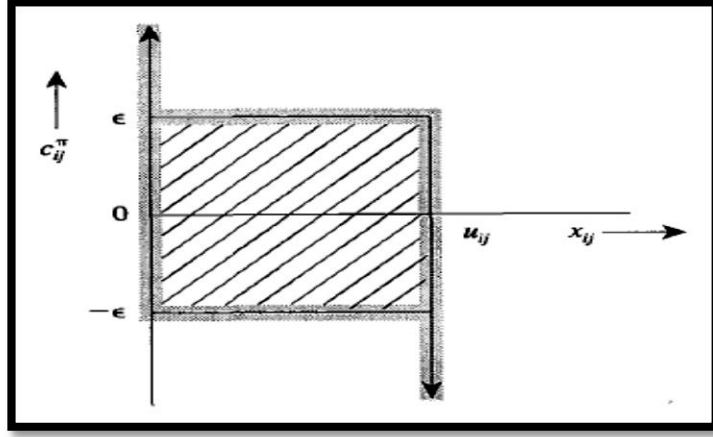


Figure 4.3 Approximate optimality for  $(i, j)$

The  $\epsilon$ -optimality conditions assume the following simpler form when stated in terms of the residual network: A flow  $\mathbf{x}$  or a (pseudo flow)  $\mathbf{x}$  is said to be  $\epsilon$ -optimal for some  $\epsilon > 0$  if  $\mathbf{x}$ , together with some node potential vector  $\boldsymbol{\pi}$ , satisfies the following  $\epsilon$ -optimality:

$$c_{ij}^{\pi} \geq -\epsilon \quad \text{for every arc } (i, j) \text{ in the residual network.} \quad (4.27)$$

- **$(\delta, \epsilon)$ -optimal solution**

If we define  $r_{ijq}^t = c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t - \mu_{jq}^t$  as the *reduced cost* for each product  $q$  and period  $t$  for some node potentials  $\boldsymbol{\mu}$ , then a flow (or pseudo flow)  $\mathbf{x}$  is said to be  $(\delta, \epsilon)$ -optimal solution to the penalty problem for some  $\epsilon > 0$  if the pair  $(\mathbf{x}, \boldsymbol{\mu})$  satisfies the following  $(\delta, \epsilon)$ -optimality conditions:

$$x_{ijq}^t \geq \delta \quad \Rightarrow \quad -\epsilon \leq r_{ijq}^t \leq \epsilon, \quad (4.28)$$

$$x_{ijq}^t < \delta \quad \Rightarrow \quad -\epsilon \leq r_{ijq}^t. \quad (4.29)$$

The  $(\delta, \epsilon)$ -optimality conditions show that  $x$  is a  $(\delta, \epsilon)$ -optimal solution if by modifying the flow by at most  $\delta$  units we get a solution which is optimal to another problem in which the demands are modified by at most  $\delta$  and the flow costs are modified by at most  $\delta$ . In other words, let  $\mathbf{x}''$  be a flow in which each flow component is different by at most  $\delta$  from its corresponding component in  $\mathbf{x}$ . We refer to  $\mathbf{x}''$  as a  $\delta$ -modified flow. In addition, let  $P''$  be a penalty problem in which the demands are different by at most  $\delta$  from the demands in  $P$ , and the flow costs are different by at most  $\delta$  from the flow costs in  $P$ . We also refer to these demands and costs as the  $\delta$ -modified demands and  $\delta$ -

*modified costs*, respectively. Thus,  $\mathbf{x}$  is a  $(\delta, \varepsilon)$ -optimal solution to P if  $\mathbf{x}''$  is an optimal solution to P''. A  $(\delta, \varepsilon)$ -optimal solution is therefore a solution which is optimal to another problem in which only the flow costs are modified by at most  $\delta$ .

To improve the theoretical convergence of the algorithm, we can limit our search only to sufficiently negative cycles. To this aim, we let the mean of a cycle be the average cost per arc. More formally,

$$\text{Mean}(W_q^t) = c(W_q^t) / \sum_{(i,j) \in A} \delta_{ij}(W_q^t).$$

Thus, we can limit our search to cycles whose mean is at most  $-\bar{\delta}$ . Therefore, in the algorithm, one can determine whether there is a negative cost cycle on  $G_q^t(x, \delta)$  whose mean  $\leq -\bar{\delta}$  rather than determining whether a negative cost cycle. As shown in Lemma 4.1, one way to limit the search to cycles with mean less than  $-\bar{\delta}$  is to add  $\bar{\delta}$  units to the cost of each arc in the  $\delta$ -residual network. It is proved that, in this case, detecting negative cost cycles on the residual network when  $\bar{\delta}$  units are added to the cost of each residual arc is equivalent to detecting negative cost cycles whose mean  $\leq -\bar{\delta}$ . For example, in our algorithm we can add  $\delta^2 \rho$  units to the cost of each arc in the  $\delta$ -residual network. Thus, we can begin searching for very negative cycles and when such cycles cannot be found, we search for less negative cycles.

**Lemma 4.1** When  $\bar{\delta}$  units are added to the cost of each arc in the residual network, the mean of a negative cost cycle is at most  $-\bar{\delta}$ . Moreover, shifting  $\delta$  units of flow around a cycle in the  $\delta$ -residual network whose mean is less than  $-\bar{\delta}$  improves the penalty objective function value by at least  $\bar{\delta}$ .

**Proof** Let  $W_q^t$  be a negative cost cycle, and let  $C_{ijq}^t$  be the costs on the  $\delta$ -residual network  $G_q^t(x, \delta)$ . By adding  $\bar{\delta}$  to the cost of each residual arc  $(i, j)$  the cost on that arc becomes  $C_{ijq}^t + \bar{\delta}$ . Thus, for each detected negative cycle:

$$\sum_{(i,j) \in W_q^t} (C_{ijq}^t + \bar{\delta}) \leq 0 \Rightarrow \sum_{(i,j) \in W_q^t} C_{ijq}^t + \bar{\delta} \sum_{(i,j) \in A} \delta_{ij}(W_q^t) \leq 0 \Rightarrow \sum_{(i,j) \in W_q^t} C_{ijq}^t \leq -\bar{\delta} \sum_{(i,j) \in A} \delta_{ij}(W_q^t),$$

and therefore,

$$\text{Mean}(W_q^t) = c(W_q^t) / \sum_{(i,j) \in A} \delta_{ij}(W_q^t) = \sum_{(i,j) \in W_q^t} \mathbf{C}_{ijq}^t / \sum_{(i,j) \in A} \delta_{ij}(W_q^t) \leq -\bar{\delta}.$$

To prove the second part, note that the costs in the  $\delta$ -residual network are referred to the change in the objective function value resulting from sending exactly  $\delta$  units of  $t$ - $q$ -flow. Thus, if we send  $\delta$  units around a cycle  $W_q^t$ , the improvement in the objective function is  $|c(W_q^t)|$ . When the mean is at most  $-\bar{\delta}$ , we get

$$c(W_q^t) = \sum_{(i,j) \in W_q^t} \mathbf{C}_{ijq}^t \leq -\bar{\delta} \sum_{(i,j) \in A} \delta_{ij}(W_q^t) \leq -\bar{\delta}.$$

Thus, the improvement resulting from any cycle of mean at most  $-\bar{\delta}$  is at least  $-\bar{\delta}$ .  $\square$

We already discussed the optimality conditions for various types of nonlinear programming problems along with KT conditions. When applied to a MCDF( $\rho$ ) problem, the optimality conditions for each arc  $(i, j)$  and product  $q$ , and period  $t$  turn out to be:

$$\begin{aligned} x_{ijq}^t (c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t - \mu_{jq}^t) &= 0, \\ c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t - \mu_{jq}^t &\geq 0, \\ \mu_{iq}^t, \mu_{jq}^t &\text{ free variables.} \end{aligned}$$

The optimality conditions may now be written as:

$$x_{ijq}^t > 0 \quad \Rightarrow \quad c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t - \mu_{jq}^t = 0, \quad (4.30)$$

$$x_{ijq}^t = 0 \quad \Rightarrow \quad c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t - \mu_{jq}^t \geq 0. \quad (4.31)$$

**Theorem 4.9** The solution at the end of a  $\delta$ - $\rho$ -phase of the MMNF algorithm is a  $(\delta, \delta\rho)$ -optimal solution to the penalty problem MCDF( $\rho$ ). Furthermore, a flow  $\mathbf{x}$  is optimal for problem MCDF( $\rho$ ) if and only if  $\mathbf{x}$  is  $(\delta, \varepsilon)$ -optimal for all sufficiently small (non-negative) values of  $\delta$ .

**Proof** Considering that there is no negative cycle, the shortest path problem from the source node of each product to any other node in the residual network is well defined [3]. It is also easy to see that with respect to the following arc costs there will be no negative cost cycle either.

$$\text{For each forward arc } (i, j) \text{ in } G_q^t(x, \delta): \quad \tilde{C}_{ijq}^t = c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \delta\rho ,$$

$$\text{For each backward arc } (j, i) \text{ in } G_q^t(x, \delta): \quad \tilde{C}_{jiq}^t = -c_{ijq}^t - 2\rho e_{ij}(x) - 2\rho e_{ij}^t(x) + \delta\rho .$$

Let  $\mu_{iq}^t$  denote the length of the shortest path from the source node of product  $q$  to node  $i$  in  $G_q^t(x, \delta)$  with respect to cost vector  $\tilde{C}$ . By the optimality conditions of the shortest path problem the following conditions be satisfied.

For any forward arc  $(i, j)$  in the residual network  $G_q^t(x, \delta)$ :

$$\mu_{jq}^t \leq \tilde{C}_{ijq}^t + \mu_{iq}^t \quad \Rightarrow \quad \mu_{jq}^t \leq c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t + \delta\rho .$$

The last inequality can be written as following by rearranging the terms.

$$-\delta\rho \leq c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t - \mu_{jq}^t . \quad (4.38)$$

For any backward arc  $(j, i)$  in the residual network  $G_q^t(x, \delta)$ ,

$$\mu_{iq}^t \leq \tilde{C}_{jiq}^t + \mu_{jq}^t \quad \Rightarrow \quad c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t - \mu_{jq}^t \leq \delta\rho . \quad (4.39)$$

Note that we include a backward arc in the residual network only for a forward arc whose  $q$ -flow at period  $t$  is  $> \delta$ . Therefore, for each arc  $(i, j)$  with flow  $x_{ijq}^t \geq \delta$  inequalities (4.38) and (4.39) apply, and therefore,

$$-\delta\rho \leq c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t + \mu_{jq}^t \leq \delta\rho \quad \forall (i, j) : x_{ijq}^t \geq \delta .$$

For each arc  $(i, j)$  with flow  $x_{ijq}^t < \delta$  only inequality (4.38) applies. Then, we get

$$-\delta\rho \leq c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t - \mu_{jq}^t \quad \forall (i, j) : x_{ijq}^t < \delta .$$

Thus, at the end of a  $\delta$ - $\rho$ -phase the following conditions are satisfied:

$$\begin{aligned} x_{ijq}^t \geq \delta &\Rightarrow -\delta\rho \leq r_{ijq}^t \leq \delta\rho, \\ x_{ijq}^t < \delta &\Rightarrow -\delta\rho \leq r_{ijq}^t. \end{aligned}$$

Consequently, the  $(\delta, \varepsilon)$ -optimality conditions reveal that the solution at the end of a  $\delta$ - $\rho$ -phase of our algorithm is  $(\delta, \delta\rho)$ -optimal solution to the penalty problem  $\text{MCDF}(\rho)$ . Now, by letting  $\delta \rightarrow 0$  and using the definition of  $(\delta, \varepsilon)$ -optimal we will meet the exact optimality conditions.  $\square$

As mentioned before, one way to assure that the algorithm detects only negative cost cycles whose mean means is at most  $-\bar{\delta}$  is to add  $\bar{\delta}$  units to the cost of each arc in the residual network. The following lemma proves that in such a case, our algorithm finds a  $(\delta, \delta\rho + \bar{\delta}/\delta)$ -optimal solution at the end of a  $\delta$ - $\rho$ -scaling phase.

**Lemma 4.2** Assuming that there is no negative cost cycle in the  $\bar{\delta}$ -added residual network whose mean is  $\leq -\bar{\delta}$ . Then, the solution is a  $(\delta, \delta\rho + \bar{\delta}/\delta)$ -optimal solution to the penalty problem  $\text{MCDF}(\rho)$ .

**Proof** The proof is similar to the proof of Theorem 4.9, with an amount of  $\bar{\delta}$  added to the cost to assure detections of cycles whose mean  $\leq -\bar{\delta}$ . Note that a positive amount  $\bar{\delta}$  is added to both forward and backward arcs. Thus, the cost of each forward arc  $(i, j)$  becomes  $C_{ijq}^t + \bar{\delta}$  and the flow cost on each backward arc  $(j, i)$  becomes  $C_{jiq}^t + \bar{\delta}$ . Therefore, along the same lines as those in Theorem 4.9 we get:

$$\begin{aligned} x_{ijq}^t \geq \delta &\Rightarrow -\delta\rho - \bar{\delta}/\delta \leq r_{ijq}^t \leq \delta\rho + \bar{\delta}/\delta, \\ x_{ijq}^t < \delta &\Rightarrow -\delta\rho - \bar{\delta}/\delta \leq r_{ijq}^t. \end{aligned}$$

Thus, when the search is limited to cycles whose mean is at most  $-\bar{\delta}$ , the algorithm finds a  $(\delta, \delta\rho + \bar{\delta}/\delta)$ -optimal solution to the penalty problem at the end of a  $\delta$ - $\rho$ -phase.  $\square$

In our algorithm, we first determine the step size and then we look for an improving direction, so it can be referred to as the  $\delta$ -shift scaling algorithm. However, there are some other approaches known as optimal-shift scaling algorithm. The algorithm is motivated by Frank-Wolfe gradient-based algorithm for quadratic programming [24][75][78]. The optimal-shift scaling algorithm, however, is more similar to conventional nonlinear algorithms. Having determined the direction of improvement, the optimal step size along this direction is then calculated. There are two main differences between the optimal shift scaling algorithm and our algorithm. These differences are in the cost of the arcs of the residual network and the amount of flow shifted around negative cost cycles. A direction of improvement is found by detecting a negative cost cycle in the residual network of each product. The cost of an arc in the residual network  $G_q^t(x, \delta)$  is the derivative cost of the penalty objective function  $f_\rho(x)$  with respect to the flow  $t$ - $q$ -flow on that arc. The costs will be calculated as follows:

$$\text{For each forward arc } (i, j) \text{ in } G_q^t(x, \delta): \quad \mathbf{C}_{ijq}^t = c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x), \quad (4.32)$$

$$\text{For each backward arc } (j, i) \text{ in } G_q^t(x, \delta): \quad \mathbf{C}_{jiq}^t = -c_{ijq}^t - 2\rho e_{ij}(x) - 2\rho e_{ij}^t(x). \quad (4.33)$$

In the optimal-shift scaling algorithms, once a negative cycle is detected, the amount of flow to be shifted around is calculated. They shift the optimal amount, i.e., the amount which results in the maximum improvement of the penalty objective function. To determine the amount of shifted flow they need to solve a one-dimensional search problem. There are various methods in the literature for solving one-dimensional search problems for convex functions [7][12]. The following lemma shows that the optimal amount is the amount that drives the derivative cost along the cycle to 0.

**Lemma 4.3** The optimal amount to shift on a negative cost cycle with respect to derivative costs is the amount which drives the total cycle cost to 0.

**Proof** In the residual network  $G_q^t(x, \delta)$ , the costs on the arcs are the derivative cost of the penalty objective function with respect to the flow on the arc. Therefore, as long as the total derivative cost of the cycle is negative, shifting an additional infinitesimal amount improves/decrease the penalty objective function value. Let  $\bar{\delta}$  be the amount which derives the cycle cost to zero. Once  $\bar{\delta}$  is shifted, any additional infinitesimal



amount shifted around the cycle increases the penalty objective function. Thus,  $\bar{\delta}$  leads to the maximum improvement of the penalty objective function.

**Lemma 4.4** Consider the residual network where the arc costs represent the derivative costs as in (4.32)-(4.33). If there are no negative cost cycles on the residual network, the current solution is optimal.

**Proof** Given the fact that there is no negative cycle, the shortest path problem from the source node of each product to any other node in the network is well defined. The negative of a dual variable of each node in an optimal solution of a shortest path problem equals the cost of the shortest path from the origin to that node [3]. Let  $\mu_{iq}^t$  denote the length of the shortest path from the source node of product  $q$  to node  $i$  in  $G_q^t(x, \delta)$ . Note that  $\mu_{iq}^t$  for each  $t$  and  $q$  could be different because the arc costs are not necessarily the same for each product at each time period. By the optimality conditions of the shortest path problem the following conditions are satisfied.

For any forward arc  $(i, j)$  in the residual network  $G_q^t(x, \delta)$ :

$$\mu_{jq}^t \leq C_{ijq}^t + \mu_{iq}^t \Rightarrow \mu_{jq}^t \leq c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t \quad (4.34)$$

By using the optimality conditions, inequality (4.34) can be written as:

$$0 \leq c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t - \mu_{jq}^t. \quad (4.35)$$

For any backward arc  $(j, i)$  in the residual network  $G_q^t(x, \delta)$ :

$$\mu_{iq}^t \leq C_{jiq}^t + \mu_{jq}^t \Rightarrow \mu_{iq}^t \leq -c_{jiq}^t - 2\rho e_{ij}(x) - 2\rho e_{ij}^t(x) + \mu_{jq}^t \quad (4.36)$$

By rearranging the terms, inequality (3.36) can be written as:

$$0 \geq c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t - \mu_{jq}^t. \quad (4.37)$$

Therefore, for each arc  $(i, j)$  with flow  $x_{ijq}^t > 0$  inequalities (4.35) and (4.37) apply.

$$0 = c_{ijq}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t + \mu_{jq}^t \quad \forall (i, j) : x_{ijq}^t > 0.$$

For each arc  $(i, j)$  with flow  $x_{ij}^t = 0$  only inequality (4.35) applies.

$$c_{ij}^t + 2\rho e_{ij}(x) + 2\rho e_{ij}^t(x) + \mu_{iq}^t - \mu_{jq}^t \geq 0 \quad \forall (i, j) : x_{ij}^t = 0.$$

Thus, in the absence of negative cycles on  $G_q^t(x, \delta)$  the optimality conditions (4.30)-(4.31) are satisfied, and so, the solution is optimal.  $\square$

**Remark 4.2** If we set  $\bar{\delta} = \delta^2 \rho$ , then the solution at the end of a  $\delta$ - $\rho$ -phase of the algorithm, when the search is limited to cycles whose mean is  $-\bar{\delta}$ , is a  $(\delta, 2\delta\rho)$ -optimal solution to the penalty problem  $\text{MCDF}(\rho)$ .

**Remark 4.3** When the search is limited to cycles with mean  $\leq -\delta^2 \rho$ , the improvement in the objective function resulting from shifting  $\delta$  units of flow around a negative cost cycle is at least  $-\delta^2 \rho$ . Hence, we can set a lower bound on how much the penalty objective function improves when one shifts  $\delta$  units of flow around a negative cost cycle in algorithm.

**Remark 4.4** The best implementation of the *Label Correcting Algorithm*, which permits arcs with negative cost, performs  $O(nm)$  operations [3]. Hence, let  $O(nm)$  be the time bound to detect negative cost cycle. Since we search for a negative cycle on a residual network of one product at a time, we may search at most  $T \cdot K$  times before detecting a negative cost cycle. Thus, the number of operations performed before a negative cost cycle is detected is  $O(nmTK)$ .

**Remark 4.5** One may prefer to use another approach to detect the cycles. The bound on the number of negative cycles detected at each  $\delta$ - $\rho$ -phase is derived by dividing the maximum possible improvement at this phase by the lower bound on the improvement per negative cost cycle as given in Lemma 4.1. Klein et al [51] and Schneur [78] present a detailed analysis of the bounds on the number of algorithmic operations for some problems with side constraints. By taking their works and feasibility of the problem into account, the most possible improvement in the objective function will be  $O(n\delta\rho v_{\max})$ .

Since the improvement for sending flow around a negative  $\delta$ -cycle is at least  $\delta^2 \rho$ , the number of negative cost cycles detected in each phase is  $O(nv_{\max}/\delta)$ .

#### 4.5 Analysis of the Algorithm and Two Specific Implementations

Nowadays, dynamic and multiperiod network flow models are applied to a variety of situations. Dynamic multiperiod flows are widely used in modeling of control processes from different technical, electrical, economic and information systems. Electricity and data transmissions, road or air traffic control, production systems, evacuation planning, production and distribution, telecommunication, transportation, communication, and management problems can be formulated and solved as single-commodity or multi-commodity problems on (multiperiod) dynamic networks. Examples and applications can be found in the literature such as Aronson [6], Cal [16], Hoppe [34], Lozovanu [57], Skutella [80], Moret [63], Moin [64], Neiro [66], and Stefansson [82].

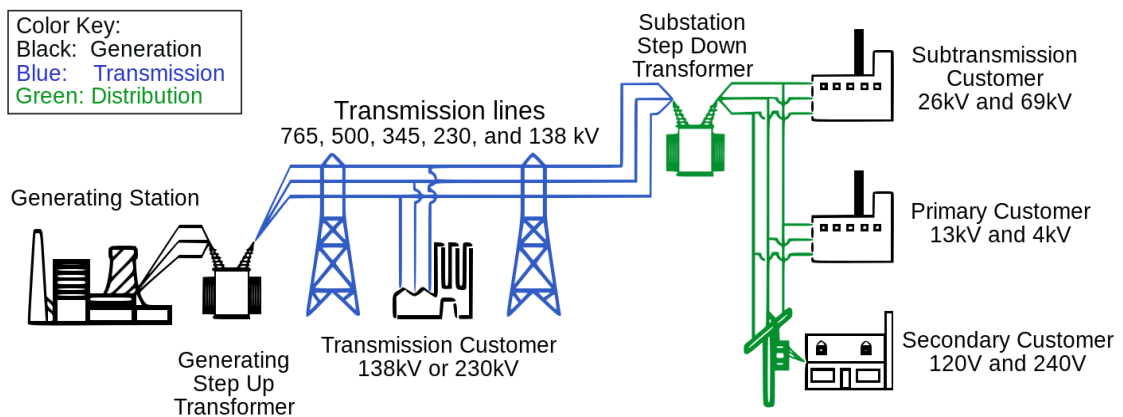


Figure 4.4 A typical diagram of an AC electricity distribution from generation stations to consumers.

This section focuses on the computational testing and analysis of our algorithm on electricity distribution-transmission network mentioned in Chapter 1. The problem is to determine the production circuit and shipping pattern within a finite planning horizon so as to minimize the daily cost (the planning horizon is a day). To illustrate the performance of our approach, we conduct a series of experiments using real data from our case study. However, our results should be applicable to any other types of problems as well. We mainly consider an interconnected distribution network that is a local low-voltage part of a large distribution-transmission electricity system that

connects the customers to the long-distance high-voltage transmission system which, in turn, connects to generating plants. It contains a number of customers (cities, factories, homes etc.) with demand for certain products (different voltage of electricity) over a specified time period.

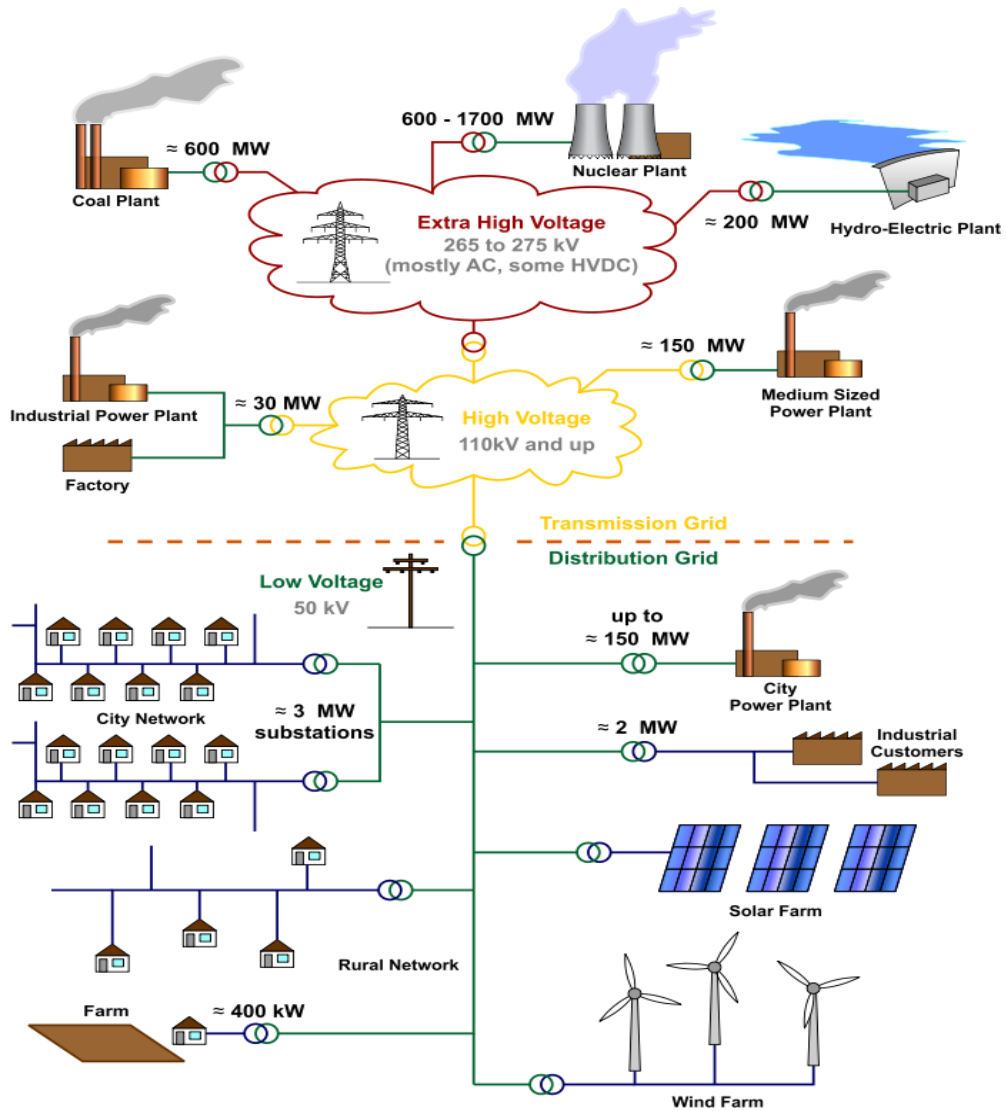


Figure 4.5 A general layout of an electricity network

We assume that demand is satisfied by shipping electricity in a fixed number of wires/lines from a number of supply/production sites, where the cost of production is assumed to be varying for different time increments and for different voltage of electricity (or fixed for each time period). We concentrate on the case in which each wire must unload all of its goods (electricity) at the demand site upon arrival.

Several parameters must be specified in order to illustrate topology of the grid, namely, various types of arc capacities (time-varying, time-commodity varying, and horizon), arc costs, node storage capacities (if desired), production sites, demand sites, the connections (wiring), number of periods  $|\mathbf{N}|$ , number of products  $|\mathbf{K}|$ , indegree and outdegree of each site, minimum and maximum values of arc capacities for each product and time period, and finally, the production/consumption capacity of each site which are, of course, varying over time and commodity.

Chapter 4 mostly discusses the scaling algorithm for minimum cost multiproduct multiperiod distribution flow problems exploiting an associated penalty problem with quadratic penalty functions. However, the general framework can be used when applying other forms of penalty functions. In this part, we investigate the MMN problems by introducing a *linear penalty function* into the auxiliary non-linear penalty problem, which leads to develop a special implementation of the MMNF algorithm. Moreover, a similar framework may also be used for solving other MMN problems, as well as MMN problems with side constraints. For example, the *multiproduct multiperiod feasibility problem* (MMFP) is similar to the minimum cost multiproduct multiperiod distribution flow problem. Here, we also discuss feasibility problem and we propose a specific implementation of MMNF algorithm as a solution approach for MMFP. Although the general steps of the algorithms are similar, they are quite different when it comes to constructing the residual networks and defining the residual costs.

#### 4.5.1 The Penalty-Based Scaling Approach with Linear Penalty Function

If we define the *excess functions* associated with each violated capacity constraints as (4.19) and consider a *linear penalty* term for violation, we get the following penalty problem  $\text{MCDF}(\rho) - \text{Linear}$ .

$$\begin{aligned}
 \text{MCDF}(\rho) - \text{Linear} \quad & \mathbf{Min} \quad f_{\rho}(x) = \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} \sum_{(i,j) \in \mathbf{A}} c_{ijq}^t x_{ijq}^t + \rho \sum_{(i,j) \in \mathbf{A}} \left( e_{ij}(x) + \sum_{t \in \mathbf{N}} e_{ij}^t(x) \right) \\
 & \sum_j x_{ijq}^t - \sum_j x_{jiq}^t = v_{iq}^t \quad \forall i \in \mathbf{V}, q \in \mathbf{K}, t \in \mathbf{N}, \\
 & 0 \leq x_{ijq}^t \quad \forall (i,j) \in \mathbf{A}, q \in \mathbf{K}, t \in \mathbf{N}.
 \end{aligned}$$

In fact, we eliminate the complicating constraints and we decompose the problem into  $|T||K|$  single-product flow problems, but introduce non-separable piece-wise linear terms into the objective function. Termination of the algorithm depends on the values of  $\delta$  and  $\rho$ . The algorithm finds a direction of interest such that by modifying the flow by  $\delta$  units in this direction the penalty objective function  $f_\rho(x)$  decreases. When such an improving direction cannot be found, the step size is decreased to  $\delta/2$ .

**Property 4.1** The original arcs of the network form the forward arcs of the residual network. Moreover, if  $(i, j)$  is a forward arc in the associated  $\delta$ -residual network  $G_q^t(x, \delta)$ , then the cost of  $(i, j)$  is the net change in the objective function of model MCDF( $\rho$ )–Linear obtained by increasing the flow on arc  $(i, j)$  by  $\delta$  units.

$$C_{ijq}^t = \delta c_{ijq}^t + 2\rho\delta \quad \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} \geq 0 \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t \geq u_{ij}^t$$

$$C_{ijq}^t = \delta c_{ijq}^t + \rho\delta \quad \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} \geq 0 \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t \leq u_{ij}^t - \delta$$

$$C_{ijq}^t = \rho \sum_{q \in K} x_{ijq}^t + \delta c_{ijq}^t + 2\rho\delta - \rho u_{ij}^t \quad \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} \geq 0 \quad \text{and} \quad -\delta < \sum_{q \in K} x_{ijq}^t - u_{ij}^t < 0$$

$$C_{ijq}^t = \delta c_{ijq}^t + \rho\delta \quad \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t \leq u_{ij} - \delta \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t \geq u_{ij}^t$$

$$C_{ijq}^t = \delta c_{ijq}^t \quad \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t \leq u_{ij} - \delta \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t \leq u_{ij}^t - \delta$$

$$C_{ijq}^t = \rho \sum_{q \in K} x_{ijq}^t + \delta c_{ijq}^t + \rho\delta - \rho u_{ij}^t \quad \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t \leq u_{ij} - \delta \quad \text{and} \quad -\delta < \sum_{q \in K} x_{ijq}^t - u_{ij}^t < 0$$

$$C_{ijq}^t = \rho \sum_{q \in K} \sum_{t \in N} x_{ijq}^t + \delta c_{ijq}^t + 2\rho\delta - \rho u_{ij} \quad \forall (i, j) \in A : -\delta < \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} < 0 \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t \geq u_{ij}^t$$

$$C_{ijq}^t = \rho \sum_{q \in K} \sum_{t \in N} x_{ijq}^t + \delta c_{ijq}^t + \rho\delta - \rho u_{ij} \quad \forall (i, j) \in A : -\delta < \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} < 0 \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t \leq u_{ij}^t - \delta$$

$$C_{ijq}^t = \rho \left( \sum_{q \in K} \sum_{t \in N} x_{ijq}^t + \sum_{q \in K} x_{ijq}^t \right) + \delta c_{ijq}^t + 2\rho\delta - \rho u_{ij} - \rho u_{ij}^t$$

$$\forall (i, j) \in A : -\delta < \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} < 0 \quad \text{and} \quad -\delta < \sum_{q \in K} x_{ijq}^t - u_{ij}^t < 0 .$$

**Property 4.2** A backward arc is included in the  $\delta$ -residual network of time  $t$  for product  $q$  if and only if we can decrease the flow of product  $q$  at period  $t$ . Moreover, if  $(i, j)$  is a backward arc in the associated  $\delta$ -residual network  $G_q^t(x, \delta)$ , then the cost of arc  $(i, j)$  in the residual network  $(i, j)$  is the net change in the objective function of model  $\text{MCDF}(\rho)$ -Linear obtained by decreasing the flow on arc  $(i, j)$  by  $\delta$  units.

$$C_{jiq}^t = -\delta c_{ijq}^t - 2\rho\delta \quad \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} \geq \delta \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t - u_{ij} \geq \delta$$

$$C_{jiq}^t = -\delta c_{ijq}^t - \rho\delta \quad \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} \geq \delta \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t \leq u_{ij}^t$$

$$C_{jiq}^t = -\delta c_{ijq}^t - \rho\delta - \rho e_{ij}^t(x) \quad \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} \geq \delta \quad \text{and} \quad 0 < \sum_{q \in K} x_{ijq}^t - u_{ij}^t < \delta$$

$$C_{jiq}^t = -\delta c_{ijq}^t - \rho\delta \quad \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} \leq 0 \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t - u_{ij}^t \geq \delta$$

$$C_{jiq}^t = -\delta c_{ijq}^t \quad \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} \leq 0 \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t \leq u_{ij}^t$$

$$C_{jiq}^t = -\delta c_{ijq}^t - \rho e_{ij}^t(x) \quad \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} \leq 0 \quad \text{and} \quad 0 < \sum_{q \in K} x_{ijq}^t - u_{ij}^t < \delta$$

$$C_{jiq}^t = -\delta c_{ijq}^t - \rho\delta - \rho e_{ij}^t(x) \quad \forall (i, j) \in A : 0 < \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} < \delta \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t - u_{ij}^t \geq \delta$$

$$C_{jiq}^t = -\delta c_{ijq}^t - \rho e_{ij}^t(x) \quad \forall (i, j) \in A : 0 < \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} < \delta \quad \text{and} \quad \sum_{q \in K} x_{ijq}^t \leq u_{ij}^t$$

$$C_{jiq}^t = -\delta c_{ijq}^t - \rho e_{ij}^t(x) - \rho e_{ij}^t(x) \quad \forall (i, j) \in A : 0 < \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} < \delta \quad \text{and} \quad 0 < \sum_{q \in K} x_{ijq}^t - u_{ij}^t < \delta.$$

#### 4.5.2 The Multiperiod Multiproduct Feasibility Problem (MMFP)

In a MMFP, a discrete multiperiod multiproduct feasible flow is a non-negative function  $x = \{x_{ijq}^t\} : A \times N \times K \rightarrow \mathbb{R}^+$  satisfying (4.15)-(4.18). Due to the structure of the MMFP, the penalty terms will be the only components of the objective function in the associated penalty problem. On the other hand, the penalty parameter has no influence on the solution of the penalty problem and can be arbitrarily fixed. Thus, only one penalty problem is solved (through scaling phases) with a decreasing scaling parameter.

Therefore, if we define the excess functions associated with each violated capacity constraints as (4.19), and set the value of the penalty parameter  $\rho$  to 1, we get the following penalty problem  $MMFP(\rho)$  for  $MMFP$  :

$$\begin{aligned}
MMFP(\rho) \quad & \mathbf{Min} \quad f_\rho(x) = \sum_{(i,j) \in A} [e_{ij}(x)^2 + \sum_{t \in N} e_{ij}^t(x)^2], \\
& \sum_j x_{ijq}^t - \sum_j x_{jiq}^t = v_{iq}^t \quad \forall i \in V, q \in K, t \in N, \\
& 0 \leq x_{ijq}^t \quad \forall (i,j) \in A, q \in K, t \in N.
\end{aligned}$$

An optimal solution to the penalty problem is a feasible solution to the MMFP if such a solution exists. Hence, the optimal value of the penalty objective function in this case is zero. The algorithm starts with an initial value for the scaling parameter  $\delta$ . At each scaling phase,  $\delta$  is fixed. Focusing on one product at one time period, we try to improve the objective function of  $MMFP(\rho)$  by modifying some of the flows of this product by exactly  $\delta$  units. The algorithm can be outlined as follows:

#### Algorithm MMFP;

**begin**

Determine an initial dynamic solution  $\mathbf{x}$  satisfying (only) flow requirements;

Set values for the parameter  $\varepsilon$  ;

Choose initial values for  $\delta$ ;

**while** the algorithm's termination criteria  $\{ \delta \leq \varepsilon \text{ or/and } gap \leq gap_u \}$  are not met **do**

**begin**

IMPROVE-APPROXIMATION ( $x, \delta$ );

$\delta := \delta/2$ ;

**end**

**end**

As already mentioned, the procedure IMPROVE-APPROXIMATION consists of two subroutines, namely, SEARCH( $q, t$ ) and AUGMENT( $w_q^t, \delta$ ). SEARCH( $q, t$ ) is a procedure that searches for a negative cost cycle on  $G_q^t(x, \delta)$ . AUGMENT( $w_q^t, \delta$ ) is a procedure which shifts  $\delta$  units of  $q$ -flow around a cycle  $w_q^t$ . The arc costs on the residual network should be defined according to the following properties.



**Property 4.3** If  $(i, j)$  is a forward arc in the  $\delta$ -residual network  $G_q^t(x, \delta)$ , then the cost of arc  $(i, j)$  in the residual network for time period  $t$  and product  $q$  is obtained by

$$\mathbf{C}_{ijq}^t = 2\delta^2 + 2\delta e_{ij}(x) + 2\delta e_{ij}^t(x) \quad \forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t \geq u_{ij} \quad \text{and} \quad \sum_{q \in \mathbf{K}} x_{ijq}^t \geq u_{ij}^t$$

$$\mathbf{C}_{ijq}^t = \delta^2 + 2\delta e_{ij}(x) \quad \forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t \geq u_{ij} \quad \text{and} \quad \sum_{q \in \mathbf{K}} x_{ijq}^t + \delta \leq u_{ij}^t$$

$$\mathbf{C}_{ijq}^t = \left( \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t \right)^2 + 2\delta \left( \sum_{q \in \mathbf{K}} x_{ijq}^t + e_{ij}(x) \right) - 2u_{ij}^t \delta + 2\delta^2$$

$$\forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t \geq u_{ij} \quad \text{and} \quad -\delta < \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t < 0$$

$$\mathbf{C}_{ijq}^t = \delta^2 + 2\delta e_{ij}^t(x) \quad \forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t + \delta \leq u_{ij} \quad \text{and} \quad \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t \geq 0$$

$$\mathbf{C}_{ijq}^t = 0 \quad \forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t + \delta \leq u_{ij} \quad \text{and} \quad \sum_{q \in \mathbf{K}} x_{ijq}^t + \delta \leq u_{ij}^t$$

$$\mathbf{C}_{ijq}^t = (1 + 2\delta) \sum_{q \in \mathbf{K}} x_{ijq}^t - 2u_{ij}^t \delta + \delta^2 - u_{ij}^t$$

$$\forall (i, j) \in \mathbf{A} : \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t + \delta \leq u_{ij} \quad \text{and} \quad -\delta < \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t < 0$$

$$\mathbf{C}_{ijq}^t = (1 + 2\delta) \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t + 2\delta e_{ij}^t(x) + 2\delta^2 - 2u_{ij} \delta - u_{ij}$$

$$\forall (i, j) \in \mathbf{A} : -\delta < \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} < 0 \quad \text{and} \quad \sum_{q \in \mathbf{K}} x_{ijq}^t \geq u_{ij}^t$$

$$\mathbf{C}_{ijq}^t = \left( \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \right)^2 + 2\delta \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t + \delta^2 - 2u_{ij} \delta$$

$$\forall (i, j) \in \mathbf{A} : -\delta < \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} < 0 \quad \text{and} \quad \sum_{q \in \mathbf{K}} x_{ijq}^t + \delta \leq u_{ij}^t$$

$$\mathbf{C}_{ijq}^t = \left( \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} \right)^2 + \left( \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t \right)^2 + 2\delta \left( \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t + \sum_{q \in \mathbf{K}} x_{ijq}^t \right) + 2\delta^2 - 2u_{ij}^t \delta - 2u_{ij} \delta$$

$$\forall (i, j) \in \mathbf{A} : -\delta < \sum_{q \in \mathbf{K}} \sum_{t \in \mathbf{N}} x_{ijq}^t - u_{ij} < 0 \quad \text{and} \quad -\delta < \sum_{q \in \mathbf{K}} x_{ijq}^t - u_{ij}^t < 0.$$

**Property 4.4** If  $(i, j)$  is a backward arc in the  $\delta$ -residual network  $G_q^t(x, \delta)$ , then the cost of arc  $(i, j)$  in the residual network for period  $t$  and product  $q$  is obtained by

$$\begin{aligned}
\mathbf{C}_{jq}^t &= 2\delta^2 - 2\delta e_{ij}(x) - 2\delta e_{ij}^t(x) & \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t &\geq \delta + u_{ij} \text{ and } \sum_{q \in K} x_{ijq}^t &\geq \delta + u_{ij}^t \\
\mathbf{C}_{jiq}^t &= \delta^2 - 2\delta e_{ij}(x) & \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t &\geq \delta + u_{ij} \text{ and } \sum_{q \in K} x_{ijq}^t &\leq u_{ij}^t \\
\mathbf{C}_{jq}^t &= \delta^2 - 2\delta e_{ij}(x) - e_{ij}^t(x)^2 & \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t &\geq \delta + u_{ij} \text{ and } 0 < \sum_{q \in K} x_{ijq}^t - u_{ij}^t < \delta \\
\mathbf{C}_{jiq}^t &= \delta^2 - 2\delta e_{ij}^t(x) & \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} &\leq 0 \text{ and } \sum_{q \in K} x_{ijq}^t &\geq \delta + u_{ij}^t \\
\mathbf{C}_{jq}^t &= 0 & \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} &\leq 0 \text{ and } \sum_{q \in K} x_{ijq}^t &\leq u_{ij}^t \\
\mathbf{C}_{jiq}^t &= -e_{ij}^t(x)^2 & \forall (i, j) \in A : \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} &\leq 0 \text{ and } 0 < \sum_{q \in K} x_{ijq}^t - u_{ij}^t < \delta \\
\mathbf{C}_{jq}^t &= \delta^2 - 2\delta e_{ij}^t(x) - e_{ij}(x)^2 & \forall (i, j) \in A : 0 < \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} < \delta & \text{ and } \sum_{q \in K} x_{ijq}^t &\geq \delta + u_{ij}^t \\
\mathbf{C}_{jiq}^t &= -e_{ij}(x)^2 & \forall (i, j) \in A : 0 < \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} < \delta & \text{ and } \sum_{q \in K} x_{ijq}^t &\leq u_{ij}^t \\
\mathbf{C}_{jq}^t &= -e_{ij}(x)^2 - e_{ij}^t(x)^2 & \forall (i, j) \in A : 0 < \sum_{q \in K} \sum_{t \in N} x_{ijq}^t - u_{ij} < \delta & \text{ and } 0 < \sum_{q \in K} x_{ijq}^t - u_{ij}^t < \delta .
\end{aligned}$$

We have analyzed MMNF algorithm and its special implementations from both theoretical and practical perspectives. The practical performances seem also support the theoretical properties we have already derived. The results of a very small number of runs over the real networks will be summarized later in this section. Computational experiences and tuning are demonstrated using many actual instances corresponding to some transmission-distribution networks from our case study having different parameters. Our networks usually have a density of almost 6/5, 4/5, or 1 and sometimes, the self-similar patterns of our networks let us exploit parallel computing. The ratio of transshipment sites to electrical substations ( $t/e$ ) is usually 1/5 or 1/2.

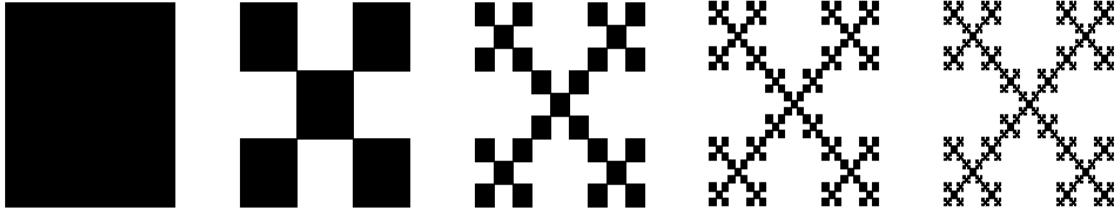


Figure 4.6 A typical self-similar pattern distribution network

Some experiments are also conducted on some random (but with real topology) multiperiod distribution networks with various parameters. The cost on each arc (for each time period for each product) is randomly chosen from a uniform distribution between user defined parameter  $c_{\min}$  and  $c_{\max}$ . The demand/requirement for each time step with respect to each product  $q$  is to be set such that the problem is feasible and likewise for arc capacities etc. For each value of  $n$  ( $n=ns+nd$  is the total number of production plants and demand sites), we create distribution networks with different indegree and outdegree in a range from 1 to  $\max\{ns, nd\}$ . The minimum and maximum capacities are set to  $u_{\min}$  and  $u_{\max}$  (usually,  $100|N|$  and  $10000|N|$ , respectively), the minimum and maximum costs are varying from problem to problem (usually, between 100 and 1000). For large values of  $u_{\max}$  no arc is saturated and therefore, the problem is easy to solve since it decomposes to  $K$  single-product problems. In order to have an interesting enough problem instances we selected a value of  $u_{\max}$  for which the problem is just feasible. To generate less congested problems, one may either use the same demand and increase the capacity, or decrease the demand and use the same capacity. In general, we expect that less congested problem instances have fewer saturated arcs at optimality. In order to generate less congested problem instance, one may solved the problem with  $u_{\max}$  replaced by  $5u_{\max}$ . This also helps us generate feasible problems. One may consider large period and/or horizon capacities to guarantee the feasibility.

For the sake of simplicity, for each arc the cost of sending a unit of flow on it is independent of the product and in addition, each product has one origin and one destination. In our tests the detected negative cost cycles have sufficiently large absolute costs relative to  $\delta\rho$ . Thus, for practical purposes, there is no need to add an amount of  $\delta\rho$  to each arc of the residual network. For each specific setting of  $n$  and  $m$ , we tested a random distribution network with a different topology and density. In addition, we have performed several computational tests and analyzed our approach's sensitivity on a

variety of problem instances. However, the behaviors are similar across our problem instances. We have investigated different implementation ideas and parametric sensitivity of the method to various data parameters, such as the number of arcs, number of time increments, density ( $m/n$ ), number of products, and the congestion in the multiperiod network, penalty parameter etc. Our algorithm and all modellings were implemented in GAMS (General Algebraic Modelling System) on a personal computer (Intel(R) Pentium(R) CPU P6200 @ 2.13GHz and 4 GB of RAM).

For feasibility problem (MMFP), we restrict our attention to the case in which each arc must unload all of its goods at the demand site upon arrival. In the following, we check the validity of MMFP algorithm on some small instances.

In MMFP, the objective is to determine whether it is possible to have a production circuit and shipping good within the predetermined time period. And if there is no feasible flow, the goal is to determine where and when this infeasibility occurs and the magnitude of the infeasibility. We may be able to handle this infeasibility curse by providing the necessary budget for creating more capacity. This feature is one of the most important attributes of our algorithm, which exactly detects where, when, and how much of infeasibility exists in the problem, rather than just reporting “*infeasible*”.

**Table 4.1 Tuning-Testing for Algorithm MMFP (checking)**

<i>Problem</i>	<i>Problem Status</i>	<i>Density, <math> K , m, n</math></i>	$\epsilon$	<i>Total Cost by (MMFP)</i>	<i>Max Violation (MMFP)</i>	<i>Execution Time (MMFP)</i>	
P. 1	<i>feasible</i>	6/5, 2, 12,10	1	6.571099E-7	0.543	0.004	
P. 2			0.9	0.098	0.312	0.002	
P. 3			0.1	0.410	0.641	0.002	
P. 4		6/5, 2, 12,10	0.01	1.573935E-7	3.967285E-4	0.003	
P. 5			0.001	1.421085E-8	1.192093E-4	0.003	
P. 6			0.00001	0.000	0.000	0.005	
P. 7			0.1	0.006	0.076	0.008	
P. 8			1, 4, 40,40	0.001	1.455192E-7	3.814697E-4	0.010
P. 9				0.00001	2.22045E-12	1.490116E-6	0.015
P. 10	<i>infeasible</i> ( $\approx 1$ unit infeasibility)	1, 4, 40,40	0.1	0.502	0.531	0.022	
P. 11			0.01	0.500	0.507	0.010	
P. 12			0.0001	0.500	0.500	0.013	
P. 13	<i>infeasible</i> ( $\approx 1$ unit infeasibility)	6/5, 2, 12, 10	0.001	0.500	0.500	0.005	
P. 14			0.1	0.505	0.502	0.003	
P. 15			1	0.781	0.781	0.003	

**Table 4.2 Some Various Settings for MMFP Algorithm**

Representing Some Good and Bad Settings for MMFP Algorithm for a Small Feasible Instance with Density 1 ( $m=n=40$ ), and Minimum and Maximum Flow Requirement 100 and 900.						
<i>The planning horizon is one day, i.e., T = One day.  K =4 , U<sub>max</sub> =6000 , t/e=1/5.</i>						
<i>Problem</i>	$\delta_0$	$\varepsilon$	<i>Total Cost by MMFP Alg.</i>	<i>Excess Cost</i>	<i>Maximum Violation</i>	<i>Execution Time</i>
<b>P.1</b>	100000	0.00001	1.45804E-11	1.45804E-11	3.818423E-6	<b>0.015 SECONDS</b>
<b>P. 2</b>	1000	0.00001	3.55271E-11	3.55271E-11	5.960464E-6	<b>0.014 SECONDS</b>
<b>P. 3</b>	10000	0.00001	2.22045E-12	2.22045E-12	1.490116E-6	<b>0.014 SECONDS</b>
<b>P. 4</b>	90000	0.999	0.244	0.244	0.494	<b>0.008 SECONDS</b>
<b>P. 5</b>	1000	0.9	0.153	0.153	0.391	<b>0.006 SECONDS</b>
<b>P. 6</b>	10000	0.9	0.117	0.117	0.342	<b>0.007 SECONDS</b>
<b>P. 7</b>	10	0.1	0.000	0.000	0.000	<b>0.020 SECONDS</b>
<b>P. 8</b>	50	0.01	0.000	0.000	0.000	<b>0.009 SECONDS</b>
<b>P. 9</b>	50	0.0001	0.000	0.000	0.000	<b>0.012 SECONDS</b>
<b>P. 10</b>	100	0.00001	0.000	0.000	0.000	<b>0.011 SECONDS</b>

In any approximation algorithm, there are various ways of evaluating how close a given solution is to the optimality. Some useful measures in our case are the flow cost (FC), excess cost (EC), total cost (TC), and maximum excess. Once we know the optimal solution, we can compare it to the flow cost of the  $\delta\rho$ -optimal solution. Note that flow cost may be smaller than the optimal solution, since we have already relaxed the capacity constraints (see Tables 4.1-4.5). The excess cost, which is the value of the penalty term, and its proportion of the total cost (TC=FC+EC), provide us information about the deviation from feasibility.

Theoretical analysis of penalty methods show that the total cost and the flow cost increase between penalty phases when  $\rho$  increases [27][75]. This is intuitive since when  $\rho$  increases the penalty is greater. In our algorithm, however, we modify  $\delta$  and  $\rho$  simultaneously, and  $\delta\rho$  decreases. We already proved that the solution at the end of each  $\delta$ - $\rho$ -phase is  $(\delta, \delta\rho)$ -optimal, therefore, at each phase we typically have a solution that is closer to the optimal solution. Therefore, the total cost usually decreases between phases until it converges to the optimal solution. The excess cost is the difference between the total cost and the flow cost. When the penalty parameter increases we expect the excess to decrease. Sometimes this decrease is sufficient to lead to a net decrease in the total excess cost. The flow cost may go up or down. We observe that the flow cost and total cost converge to the solution value at the end of the algorithm and thus, the excess cost converges to zero (see Tables 4.3-4.11).

We also observe that the general behavior of the maximum excess is similar across the different data sets. An interesting observation may be found in following figure which is drawn for a data set of density  $4/5$  having 4 commodities ( $\delta = 1, \rho_u = 11111, \varepsilon = 0.00001$ ). The figure shows that the maximum excess (and so, the excess term) sharply decrease in the first few phases, although the value of  $\rho$  is smaller than its value in later phases. This is probably due to the linear relationship between  $\delta$  and the maximum excess, the somewhat quadratic relationship between  $\delta$  and the excess term, and the fact that  $\delta$  decreases in a faster rate than  $\rho$  increases.

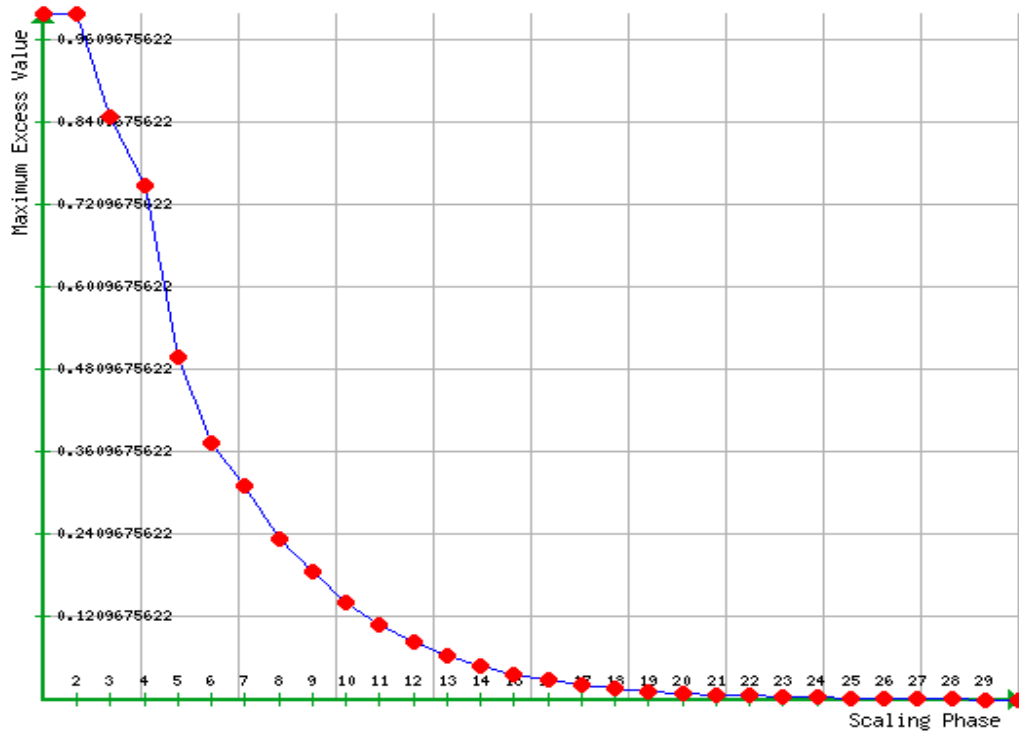


Figure 4.7 A typical maximum excess progress at the end of each scaling phase of some application of MMNF algorithm (setting:  $\delta=1, \rho_u=11111, \varepsilon=0.00001$ )

As we already proved, during each  $\delta$ - $\rho$ -phase of the MMNF algorithm the total cost decreases because flow is sent around negative cost cycles (see the last three sections). When a phase terminates we increase the penalty parameter and therefore, the total cost which corresponds to the new penalty parameter increases. The total cost decreases again during the next  $\delta$ - $\rho$ -phase. Thus, at the end of a  $\delta$ - $\rho$ -phase the total cost is not necessarily smaller than the total cost at the end of the previous phase. Nevertheless, the total cost typically decreases between phases (or it increases only by a little amount).

**Table 4.3 Parameter Settings and Tuning-Testing for Some MMNs**

For all feasible problem instances P. 1-11, the running time by GAMS $\leq 2s$ . For all feasible problem instances P. 12-16, the running time $\leq 10s$ .										
For all (real) problems, the planning horizon time is one day, i.e., T = One day.										
<i>Problem</i>	<i>Density,</i> $ K , m, n$	<i>Optimal</i> <i>Solution</i>	<i>Parl.</i> <i>Comp.</i>	$R$	$\rho_u$	$\epsilon$	<i>Total</i> <i>Cost by</i> <i>MMNF Alg.</i>	<i>Excess</i> <i>Cost</i>	<i>Max</i> <i>Excess</i>	<i>Execution</i> <i>Time (s)</i>
<b>P. 1</b>	4/5, 3, 12, 15	117700.0	no	1.7	113	0.001	117699.601	0.399	0.042	< 2
<b>P. 2</b>	6/5, 3, 18, 15	168351.0	no	1.7	213	0.0001	168350.802	0.198	0.031	< 2
<b>P. 3</b>	4/5, 6, 24, 20	199870.0	no	1.5	100	0.001	199869.877	0.122	0.035	< 2
<b>P. 4</b>	4/5, 2, 8, 10	469560.0	no	1.7	777	0.0001	469558.599	1.402	0.042	< 2
<b>P. 5</b>	4/5, 5, 20, 25	225900.0	no	1.3	70	0.01	225900.001	< 0.001	< 0.001	< 2
<b>P. 6</b>	4/5, 6, 24, 30	199870.0	no	1.3	1000	0.0001	199869.988	0.012	0.003	< 2
<b>P. 7</b>	4/5, 4, 16, 20	469560.0	no	1.3	777	0.00001	469558.598	1.402	0.042	4.156
<b>P. 8</b>	4/5, 6, 24, 30	199870.0	no	1.3	200	0.0001	199869.939	0.061	0.017	< 2
<b>P. 9</b>		311950.0	no	1.3	311	0.001	311945.542	4.458	0.078	3.062
<b>P. 10</b>			no	1.3	99	0.001	311935.996	14.003	0.245	< 2
<b>P. 11</b>			no	1.3	111	0.001	311937.510	12.490	0.218	< 2
<b>P. 12</b>	6/5, 70, 420, 350	3119500.0	yes	1.3	100	0.0001	3119478.97	21.03	0.145	11
<b>P. 13</b>			yes	1.7	317	0.00001	3119493.37	6.63	0.046	< 2
<b>P. 14</b>			yes	1.7	1113	0.00001	3119498.11	1.89	0.013	< 2
<b>P. 15</b>	4/5, 40, 160, 200	1914720.0	yes	1.3	10	0.001	1914717.75	2.25	0.150	< 3
<b>P. 16</b>			yes	1.7	11111	0.00001	1914720.00	< 0.0001	< 0.0001	< 3

As reported in Tables 4.4 and 4.5, Cplex failed to solve most of our large instances. It also did not manage to generate a proper model of our instances after allocating a huge amount of work space. In contrast, MMNF algorithm found the solution of all samples of ours in a reasonable amount of time by allocating a very small amount of work space.

**Table 4.4 Computational Results for Some Large Feasible MMNs**

An application of MMNF algorithm for some large random feasible instances with: uniform flow requirement $U(100, 900)$ , and uniform cost $U(1,100)$ , and uniform capacity $U(100*/T, 10000*/T)$ for sample problems P.1 – P.4 and uniform capacity $U(300, 900)$ for sample problems P.5 – P.6.						
For all problems, the planning horizon time is one day( $T=1$ ). <span style="float: right;"> T =24 (hrs) for P.1 and P.2.</span>						
Instances Settings	P.1	P.2	P.3	P.4	P.5	P.6
<b>Problem's Status</b>	*** Feasible ***					
<b># Node-Commodities</b>	133,056	165240	73,227	245,100	64,800	50,500
<b># Arc-Commodities</b>	1.024531E+7	1.366438E+7	2.080964E+7	6.606000E+7	1.08141E+7	1.300000E+7
<b># Commodities</b>	24	27	77	300	100	50
<b># Constraints</b>	577,731	692,415	613,737	685,500	281,082	570,500
<b># Variables</b>	1.024531E+7	1.366438E+7	2.080964E+7	6.606000E+7	1.081410E+7	1.300000E+7
<b>Network Density</b>	77	82.694	284.180	269.523	166.884	257.426
<b>R</b>	1.700	1.300	1.700	1.700	1.300	1.700
<b><math>\rho_u</math></b>	100	113	110	110	110	110
<b><math>\epsilon</math></b>	0.10	0.001	0.010	0.010	0.0010	0.010
<b>MaxExcess (MMNF)</b>	0.000	0.005	0.096	0.095	1.177	0.290
<b>Average Excess (MMNF)</b>	0.000	0.001	0.032	0.032	1.275	1.519
<b>Work Space Allocated for MMNF (reported by GAMS)</b>	93.1 Mb	135.6 Mb	156.0 Mb	620.6 Mb	18.2 Mb	14.5 Mb
<b>Work Space Allocated for CPLEX (reported by GAMS)</b>	> 2,000 Mb	> 2,300 Mb	> 2,000 Mb	> 2,200 Mb	> 1,800 Mb	> 1,800 Mb
<b>Running Time for MMNF (by GAMS)</b>	1,327 SECONDS	1572.247 SECONDS	3,630 SECONDS	21,454 SECONDS	290 SECONDS	249 SECONDS
<b>Running Time for CPLEX (by GAMS)</b>	Failed to solve in >> 2,000 Sec.	Failed to solve in >> 2,900 Sec.	Failed to solve in >> 4,000 Sec.	Failed to solve in >> 30,000 Sec.	Failed to solve in >> 1,800 Sec.	Failed to solve in >> 1,800 Sec.



As discussed earlier, MMNF algorithm not only reports the approximate amount of *violation* in the network, but it can also report the approximate cost to resolve this infeasibility, if desired. It, on the other hand, can detect for which arcs for which commodities for which time period the network has infeasibility. This reported excess cost, in this case, can be seen as the approximate necessary budget for the decision maker to get rid of the curse of infeasibility.

**Table 4.5 Computational Results for Some Infeasible MMNs**

An application of MMNF algorithm for some random infeasible instances with: uniform flow requirement $U(10000, 20000)$ , uniform capacity $U(100*/ T , 10000*/ T )$ , and uniform cost $U(1,100)$ .						
For all problems, the planning horizon time is one day ( $T=1$ ). $ T =13$ for P.1 and P.2.						
<i>Instances</i> <i>Settings</i>	<i>P.1</i>	<i>P.2</i>	<i>P.3</i>	<i>P.4</i>	<i>P.5</i>	<i>P.6</i>
<b>Problem's Status</b>	*** <i>Infeasible</i> ***					
<b># Node-Commodities</b>	42,471	28,431	507	12,397	17,745	65,667
<b># Arc-Commodities</b>	1,401,543	767,637	6,591	539,539	968,877	6,610,419
<b># Commodities</b>	33	27	13	77	91	177
<b># Constraints</b>	88209	59049	1521	26411	39039	140361
<b># Variables</b>	1,401,543	767,637	6,591	539,539	968,877	6,610,419
<b>Network Density</b>	33.000	27.000	13.000	43.522	54.600	100.666
<b><math>R</math></b>	1.300	1.300	1.300	1.300	1.700	1.300
<b><math>\rho_u</math></b>	99.000	99.000	113.000	113.000	113.000	113.000
<b><math>\epsilon</math></b>	0.001	0.001	0.0001	0.010	0.0100	0.0010
<b>Maximum Violation (reported by MMNF)</b>	315254.004	7771.000	13252.000	15549.001	12836.002	10795.000
<b>Average Violation (reported by MMNF)</b>	116230.668	5059.131	4417	5183	4278	3598
<b>Work Space Allocated for MMNF</b>	16.6 Mb	9.8 Mb	1.4 Mb	3.5 Mb	6.6 Mb	28.1 Mb
<b>Running Time for MMNF (by GAMS)</b>	118 SECONDS	52 SECONDS	0.397 SECONDS	6 SECONDS	22 SECONDS	305 SECONDS

The influence of the number of products and the number of time increments on the performance of the algorithm is more or less similar. Both the number of cycle searches and the running time are mainly influenced by the number of products and time periods, usually as increasing function (but not always). It appears that there is no obvious trend, and therefore, we cannot deduce anything regarding the influence of products and time increments on the number of iterations.

**Table 4.6 Parameter Settings for Algorithms ‘MMNF’ and ‘MMNF-Linear’**

<i>Problem</i>	<i>Optimal Solution</i>	<i>R</i>	$\rho_u$	$\varepsilon$	<i>Total Cost by (MMNF)</i>	<i>Total Cost by MMNF-Linear</i>	<i>Max Excess (MMNF)</i>	<i>Max Excess (MMNF-Linear)</i>	<i>Execution Time (MMNF)</i>	<i>Execution Time (MMNF-Linear)</i>		
<b>P. 1</b>	296736	1.3	10000	0.0001	296736.784	296736.900	0.008	0.00	11.812	49.963		
<b>P. 2</b>		1.3	1000	0.001	296736.462	296736.900	0.028	0.00	1.477	50.118		
<b>P. 3</b>		1.3	300	0.001	296735.309	296736.900	0.073	0.00	0.305	49.909		
<b>P. 4</b>		1.2	10000	0.0001	296735.559	296736.903	0.077	0.00	6.949	262.331		
<b>P. 5</b>		1.2	500	0.001	296734.120	296736.903	0.159	0.00	0.915	264.465		
<b>P. 6</b>		1.2	55	0.0001	296728.220	296736.900	0.397	1.3E-6	0.077	262.085		
<b>P. 7</b>		1.2	33	0.0001	296722.433	1.3015E+8	0.662	1400.0	0.024	30.291		
<b>P. 8</b>		80229	1.3	213	0.0001	80228.144	80229.000	0.063	0.00	0.128	2.600	
<b>P.9</b>			1.3	500	0.1	80226.060	80229.001	0.312	0.001	0.014	2.698	
<b>P. 10</b>			1.2	377	0.0001	80227.921	80229.002	0.100	0.000	3.535	45.724	
<b>P. 11</b>			1.1	300	0.0001	80214.658	3341748.97	1.180	427.00	1.698	148.105	
<b>P. 12</b>			1.1	377	0.00001	80213.861	3169451.99	1.246	427.00	19.621	1784.644	
<b>P. 13</b>			1.2	777	0.1	80203.272	8506286.33	2.371	700.00	0.027	5.494	
<b>P. 14</b>			469560	1.3	1500	0.0001	469559.274	469560.001	0.022	0.00	4.062	17.290
<b>P. 15</b>				1.4	1600	0.00001	469559.319	469560.451	0.021	0.00	1.133	2.644
<b>P. 16</b>				1.5	2000	0.001	469559.456	469971.377	0.016	0.00	2.982	1.863

As it is observed and we expected from penalty function methods [7][27][78], in general, when the value of the penalty upper bound  $\rho_u$  is too small, then the excess on some arcs at the end of the last phase is not sufficiently small. Thus, more phases will be required in order to get a good solution. The algorithm also shows the same behavior when we remove  $\rho=\rho_u$  from termination criteria. In this case, when modification rate  $R$  is too small the final value of  $\rho$  is relatively small, and excess on some arcs at the end of the last phase is not small enough.

For example, when we set  $R=0.1$ , even though  $\rho_0$  is relatively large, the final value of  $\rho$  is small. When some more phases are performed the amount of violation decreases, but the total number of iterations increases and therefore, the total time increases as well. See, for example, Table 4.7 and 4.9 - 4.11.

**Table 4.7 MMNF Sensitivity w.r.t. Penalty Parameter Upper Bound ( $\rho_u$ )**

For all feasible problem instances, the running time by GAMS $\leq 2s$ .											
<i>Problem</i>	<i>Density,  K , m, n</i>	<i>Optimal Solution</i>	<i>R</i>	<i><math>\rho_u</math></i>	<i><math>\delta_0</math></i>	<i>Total Cost by MMNF Alg.</i>	<i>Excess Cost</i>	<i>Max Excess</i>	<i>Execution Time (s)</i>		
<i>Set1</i>	4/5, 3, 12, 15	117700.0	1.5	7	$\rho_0$ 1.3	117693.554	6.542	0.684	< 2		
<i>Set2</i>			1.5	10		117695.488	4.533	0.476	< 2		
<i>Set3</i>			1.5	17		117697.346	2.680	0.281	< 2		
<i>Set4</i>			4/5, 3, 12, 15	117700.0	1.5	23	$\epsilon$ 0.1	117698.038	1.981	0.208	< 2
<i>Set5</i>					1.7	27		117698.329	1.692	0.177	< 2
<i>Set6</i>					1.7	57		117699.208	0.774	0.082	< 2
<i>Set7</i>					1.7	113		117699.601	0.413	0.043	< 2
<i>Set1</i>	6/5, 7, 42, 30	311950.0	1.3	7	$\delta_0=20$	311751.946	198.053	3.464	< 2		
<i>Set2</i>			1.3	17		311868.449	81.551	1.426	< 2		
<i>Set3</i>			6/5, 7, 42, 30	311950.0	1.3	57	$\rho_0=1.3$	311925.678	24.322	0.425	< 2
<i>Set4</i>					1.3	87		311934.065	15.935	0.279	< 2
<i>Set5</i>					1.7	99		311935.996	14.004	0.245	< 2
<i>Set6</i>					1.9	99	$\epsilon$ 0.0001	311935.996	14.004	0.245	< 2
<i>Set7</i>					1.1	99		311807.332	174.371	2.773	15.547

Not surprisingly, on the other hand, when for some combinations of  $\rho_0$  and  $R$  the penalty parameter is too large in the initial phases, large values of excess are not allowed even in the initial phases, and so the algorithm tries to find a good solution without fully exploiting the relaxation of the horizon and period capacity constraints. As a result, the algorithm converges more slowly. Such cases usually occur when  $\rho_0 > 1.7$  and  $R > 2$ .

**Table 4.8 MMNF-Linear Sensitivity w.r.t. Penalty Parameter Upper Bound ( $\rho_u$ )**

The planning horizon time is set to be one day.											
<i>Problem</i>	<i>Density,  K , m, n</i>	<i>Optimal Solution</i>	<i>R</i>	<i><math>\rho_u</math></i>	<i><math>\delta_0</math></i>	<i>Total Cost by MMNF-linear</i>	<i>Excess Cost</i>	<i>Max Excess</i>	<i>Execution Time (s)</i>		
<i>Set1</i>	1, 4, 24, 20	469560.0	1.3	99	$\rho_0=1.7$	469559.992	1.296576E-6	1.144409E-4	17.699		
<i>Set2</i>			1.3	111		469560.002	4.038156E-8	1.907349E-5	17.798		
<i>Set3</i>			1.3	150		469559.999	5.456968E-8	1.907349E-5	17.764		
<i>Set4</i>			1, 4, 24, 20	469560.0	1.3	270	$\epsilon$ 0.01	469560.002	0.000	0.000	17.777
<i>Set5</i>					1.3	390		469560.002	0.000	0.000	17.847
<i>Set6</i>					1.3	500		469560.001	0.000	0.000	17.800
<i>Set7</i>					1.3	1000		469560.001	0.000	0.000	17.651
<i>Set1</i>	4/5, 4, 16, 20	191472.0	1.1	30	$\delta_0=30$	191472.001	1.30302E-10	3.814697E-6	2.742		
<i>Set2</i>			1.1	110		191472.001	1.30302E-10	3.814697E-6	2.807		
<i>Set3</i>			4/5, 4, 16, 20	191472.0	1.1	200	$\rho_0=1.1$	191472.001	1.30302E-10	3.814697E-6	2.774
<i>Set4</i>					1.1	10000		191472.001	1.30302E-10	3.814697E-6	2.795
<i>Set5</i>					1.1	100000		$\epsilon=0.0001$	3.814697E-6	1.30302E-10	3.814697E-6

Similarly, when  $\rho_0 = 1.5$  and  $R = 2.9$ , the initial value of  $\rho$  is small enough, but the rate of modification is too large. This situation also leads to slower convergence, similar to the case in which  $\rho_0 = 2.2$  and  $R = 1.9$  (see Tables 4.7, 4.8, 4.10, 4.11). Table 4.7 and 4.8 show how  $\rho_u$  can improve or worsen the progress of the objective function. However, it should be noted that assigning a large value to  $\rho_u$  cannot guarantee a better progress for objective function, but some good trade-off between  $R$ ,  $\rho_u$ ,  $\rho_0$ , and  $\varepsilon$  can.

To examine the influence of the increasing penalty parameter  $\rho_0$  and modification rate  $R$ , we tested several real distribution networks of our samples having different densities and commodities with different parameter settings, and compared the results in the following Tables. These tables present the maximum excess and execution time as a function of  $\rho_0$  and  $R$  for two problems sets.

**Table 4.9 MMNF Sensitivity w.r.t. Initial Penalty Parameter Value ( $\rho_0$ )**

For all settings, the running time by GAMS $\leq 10$ s.										
Problem Setting	Density, $ K , m, n$	Optimal Solution	$R$	$\rho_0$	$\delta_0=20$	Total Cost by MMNF Alg.	Excess Cost	Max Excess	Computational Time (s)	
				Problem Setting 1						33
Problem Setting 1	6/5, 70, 420, 350	3119500.0	1.9	23	$\rho_u=77$	3119472.69	27.3	0.188	20	
				10		3119472.69	27.3	0.188	17	
				1.9	$\varepsilon$	3119319.95	180.05	0.315	8	
				1.7	0.01	3119319.95	180.05	0.315	8	
				1.5		3119319.95	180.05	0.315	< 8	
				1.3		3119319.95	180.05	0.315	< 7	
				1.3	27	$\delta_0=20$	3119478.97	21.03	0.145	30
					17		3119478.97	21.03	0.145	16
			9		$\rho_u$	3119478.97	21.03	0.145	11	
			1.9		100	3119361.36	138.64	0.243	10	
			1.7			3119361.36	138.64	0.243	9	
			1.3	1.5	$\varepsilon$	3119361.36	138.64	0.243	9	
				1.3	0.0001	3119361.36	138.64	0.243	13	

It is observed that when the penalty parameter is fixed, the maximum excess decreases in the first few phases due to the additional shifting possibilities, which arise when  $\delta$  decreases. After a few phases, however, the maximum excess converges to a certain value much (larger than zero), and stays at that value for the remaining phases. When the penalty increases at each phase, the maximum excess decreases and converges to zero.

**Table 4.10 MMNF-Linear Sensitivity w.r.t. Penalty Modification Rate ( $R$ )**

For all feasible problem instances, the running time by GAMS $\leq 2s$ .								
The planning horizon time is set to be two half days.								
Problem Setting 1			$R$	$\rho_u = 77$	<i>Total Cost by MMNF Alg.</i>	<i>Excess Cost</i>	<i>Max Excess</i>	<i>Computational Time (s)</i>
			5					469545.891
			2.3	$\rho_0 = 1.3$	469545.891	14.144	0.429	2.6
			1.9		469545.891	14.144	0.429	2.6
			1.7	$\delta_0 = 27$	469545.889	14.139	0.429	2.9
			1.3		469545.889	14.139	0.429	1.9
			1.2	$\varepsilon = 0.01$	469516.782	65.261	1.648	1.2
			1.1		469342.329	266.853	7.352	1.0
	Density, $ K $	Optimal Solution						
Problem Setting 2	4/5, 2	469560.0	2.9		469558.599	1.402	0.042	15.016
			2.3	$\rho_u = 777$	469558.599	1.402	0.042	22.953
			1.9		469558.599	1.402	0.042	22.641
			1.7	$\rho_0 = 1.3$	469558.599	1.402	0.042	0.218
			1.5	$\delta_0 = 57$	469558.599	1.402	0.042	0.313
			1.3		469558.599	1.402	0.042	2.907
			1.1	$\varepsilon = 0.0001$	469495.616	78.694	2.168	21.265
Optimal Settings			$\rho_0 = 1.3, \rho_u = 1110,$	$R=1.7, \varepsilon = 0.0001.$	469559.019	0.981	0.030	0.938
			$\rho_0 = 1.5, \rho_u = 2000,$	$R=1.7, \varepsilon = 0.0001.$	469559.456	0.544	0.016	0.391
			$\rho_0 = 1.3, \rho_u = 777,$	$R=1.7, \varepsilon = 0.001.$	469558.599	1.401	0.042	0.891
			$\rho_0 = 1.3, \rho_u = 1110,$	$R=1.7, \varepsilon = 0.001.$	469559.019	0.981	0.030	0.953

As it is seen, setting  $R$  without considering the other parameters does not have any good influence on the optimal solution value. Therefore, saying that there is unique good value for  $R$  is totally wrong, but a good combination of between  $R, \rho_u, \rho_0,$  and  $\varepsilon$  is true. In one word, we do not recommend linear penalty function model, as it is so very sensitive to parameters which may cause a very large error in estimation of the objective function. This error is mostly due to the low cost of the negative cost cycles found in phases and/or having not enough number of phases. This all can only be resolved by setting the parameters carefully, which is different problem to problem.

**Table 4.11 MMNF-Linear Sensitivity w.r.t. Penalty Modification Rate ( $R$ )**

The planning horizon time is set to be two half days.									
Problem Setting 1			$R$	$\delta_0 = 200$	<i>Total Cost by MMNF-Linear</i>	<i>Excess Cost</i>	<i>Max Excess</i>	<i>Computational Time (s)</i>	
			1.3	$\rho_0 = 1.7$	469560.018	0.000	0.000	<b>8.983</b>	
			1.35		469964.974	0.000	0.000	<b>3.420</b>	
			1.4	$\rho_u = 400$	469560.272	0.000	0.000	<b>1.280</b>	
			1.45		469965.155	0.000	0.000	<b>0.633</b>	
			1.5	$\epsilon = 0.01$	469560.723	0.000	0.000	<b>2.415</b>	
			1.55		469965.020	0.000	0.000	<b>0.321</b>	
			1.6		469969.806	0.000	0.000	<b>0.138</b>	
			1.65		469967.462	0.000	0.000	<b>0.166</b>	
			1.7		469969.806	0.000	0.000	<b>0.136</b>	
			1.25	$\delta_0 = 77$	469560.059	0.000	0.000	<b>147.469</b>	
			1.27		469560.059	0.000	0.000	<b>73.768</b>	
			1.3		$\rho_0 = 1.5$	469965.194	0.000	0.000	<b>35.518</b>
			1.35		$\rho_u = 300$	469966.404	0.000	0.000	<b>11.264</b>
			1.4		$\epsilon = 0.0001$	469967.269	0.000	0.000	<b>5.722</b>
			1.5		469973.059	0.000	0.000	<b>1.660</b>	
			1.7		469997.121	0.000	0.000	<b>0.436</b>	
				$\delta_0 = 170$	$\rho_0 = 1.3$	469560.035	3.28626E-11	2.98023E-7	<b>20.742</b>
				$R = 1.3$	$\rho_u = 370$				
				$\epsilon = 0.001$					
				$\delta_0 = 200$	$\rho_0 = 1.7$	469560.723	0.000	0.000	<b>2.521</b>
				$R = 1.5$	$\rho_u = 370$				
				$\delta_0 = 200$	$\rho_0 = 1.7$	469560.723	0.000	0.000	<b>2.490</b>
				$R = 1.5$	$\rho_u = 430$				
				$\delta_0 = 200$	$\rho_0 = 1.7$	469560.723	0.000	0.000	<b>2.502</b>
				$R = 1.5$	$\rho_u = 430$				
				$\epsilon = 0.0001$					

## 4.6 Summary and Concluding Remarks

This chapter addressed the multiperiod multiproduct distribution network problems, where all network parameters change over time and products. The linear minimum cost flow problem in the discrete-time settings with varying network parameters was investigated, and we used scaling and  $\delta$ -optimality, together with penalty function methods, to develop the first network-based scaling algorithm for the minimum cost multiperiod multiproduct distribution planning problems. The feasibility problem of the distribution networks in the discrete-time settings with varying parameters was also investigated, and then, a specific implementation of our scaling-based algorithm was developed.

Moreover, we analyzed the algorithms from both theoretical and practical perspectives. The practical performances supported the theoretical properties we already derived. Computational experiences and tuning were demonstrated using many actual instances corresponding to some real electricity transmission-distribution networks from our case study and many random instances. However, it should be noted that our comparisons are all approximate, and they are obtained by applying a set of typical data from our samples to provide some insight regarding the behavior of our scaling algorithm and the impact of various parameters on the performance of the algorithm.

There may be many factors that can influence the running time, such as the amount of available RAM, the compiler and the programming language, network structure and topology, way of detecting negative cost cycles etc. In addition, some other approaches may find the optimal solution to our samples, but may spend some additional time to get from an approximately optimal solution to an optimal one.

It is believed that an approximate solution which is within 0.01%-0.05% from optimality and from feasibility has the same practical value as an optimal solution.

# Chapter 5

## 5 Open Problems and Future Research

Some issues and areas for future directions of research are outlined below:

1. A very challenging, but very interesting and yet open, issue is the study of optimal routing in the multiperiod multiproduct distribution networks with uncertain costs and uncertain capacities. The purpose would be to obtain the uncertainty distribution of the total shipping cost. Uncertainty theory could be one tool to deal with indeterminacy factors in uncertain multiperiod (multiproduct) systems.
2. In the last chapter, we focused on a penalty-based scaling algorithm for MCDF problem on an MMN. However, the general scheme, yet, can be used to solve other types of multiperiod network problems, as well as general network flow problems with side constraints. Our penalty-based algorithm itself may also be extended to other linear or nonlinear production-distribution systems in general or quadratic nonlinear systems in particular. Investigating more special cases of production-distribution network flow problems with side constraints could be of interest.
3. The feasibility of the uncertain multiperiod multiproduct distribution problems can be another topic which can be handled by our algorithms by some essential modifications to cope with indeterminacy factors (e.g., uncertain capacities).
4. Developing and testing a parallel version of the algorithm presented in the last chapter can be another topic for future research. Multiperiod multiproduct flow algorithm reveals an inherent parallelism that makes it attractive for parallel implementations. Such implementations usually require a modification of existing algorithmic techniques and require different data structures.
5. Testing augmented Lagrangian methods instead of penalty methods to solve the multiperiod multiproduct flow problems could be one another issue of interest.



## 6 Bibliography

- [1] Abdelhay A. Sallam and Om P. Malik, *Electric Distribution Systems*. IEEE Computer Society Press (2011)
- [2] Abraham, I., Fiat, A., Goldberg, A., Werneck, R., Highway Dimension, Shortest Paths, and Provably Efficient Algorithms, *ACM-SIAM Symposium on Discrete Algorithms*, 782-793 (2010)
- [3] Ahuja, R.K., Magnanti M., Orlin J., *Network Flows. Theory, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs, New Jersey (1993)
- [4] Amanulla, B., Chakrabarti, S., Singh, S., Reconfiguration of power distribution Systems Considering Reliability and Power loss. *IEEE Trans. Power Deliv* 27 (2012) 918–926
- [5] Andreas D., Alf K., *Beyond Manufacturing Resource Planning (MRP II): Advanced Models and Methods for Production Planning*, Springer (1998)
- [6] Aronson, J.E., A survey of dynamic network flows. *Annals of Operations Research* 20, 1–66 (1989)
- [7] Avriel, Mordecai, *Nonlinear Programming: Analysis and Methods*, Dover, ISBN 0-486-43227-0 (2003)
- [8] Barnett, D., Binkley, J., McCarl, B., The effects of US port capacity constraints on national and world grain shipment , Techn paper, Purdue University (1982)
- [9] Bazaraa, M., Jarvis, J., Sherali, H.: *Linear Programming and Network flows*, John Wiley & Sons (2009)
- [10] Bartle, R. Sherbert, D., *Introduction to Real Analysis*. John Wiley & Sons (2011)
- [11] Bertsekas, D., *Linear Network Optimization, Algorithms and Codes*, MIT press (1991)
- [12] Boyd, S., Vandenberghe, L., *Convex Optimization*. Cambridge, Cambridge University Press (2004)
- [13] Borndorfe, R., Ferreira, C., Martin A., Decomposing matrices into blocks, *SIAM J. Opt.* 9 236-269 (1998)
- [14] Binks, B.P., Fletcher, P., Holt, B., Selective Retardation of Perfume Oil Evaporation from Oil-in-Water Emulsions Stabilized by Either Surfactant or Nanoparticles, *Langmuir J.* 26(23) 18024–18030 (2010)
- [15] Bertsimas, D. and Tsitsiklis, J., *Introduction to Linear Optimization*. Athena Scientific, Belmont, Massachusetts, USA (1997)
- [16] Cal , X., Sha, D., Wong, C., *Time-Varying network Optimization*, Springer (2007)
- [17] Chen P., Pinto, J., Lagrangean-Based Techniques for the Supply Chain Management of Flexible Process Networks. *Comp. and Chem. Eng.* 32 2505–2528 (2008)

- [18] Chen, C., M. Engquist, Primal simplex approach to pure processing networks. *Management Science* 1582-1598 (1986)
- [19] Colebrook, C., Turbulent flow in pipes, with particular reference to the transition region between smooth and rough pipe laws, *J. Inst. Civil Engrs.*, London (1939)
- [20] Cormen, T., Leiserson C., Ronald L., Clifford S., *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill ISBN 0-262-03293-7 (2001)
- [21] Dantzig, G., Wolfe, P., The Decomposition Algorithm for Linear Programs. *Econometrical* 29 767-778 (1961)
- [22] Dantzig, G., Wolfe, P., Decomposition Principle for Linear Programs. *Opns. Res.* 8 101-111 (1960)
- [23] Dinic, E., *The Method of Scaling and Transportation Problems*. *Issledovaniya po Diskretnoi Matematike*, Science, Moscow (1973)
- [24] Eugene L., *Combinatorial Optimization: Networks and Matroids*, Dover. 117–120. ISBN 0-486-41453-1 (2001)
- [25] Ferris, M. and Horn, J.D., Partitioning mathematical programs for parallel solution. *Math. Prog.* 80 35-61 (1998)
- [26] Ferris, M., Voelker, M., Slice models in general purpose modelling systems: An application to DEA. *Opt. Meth. and Soft.* 17 1009-1032 (2002)
- [27] Fiacco, A., McCormick, G., *Nonlinear programming: Sequential Unconstrained Minimization Techniques*. John-Wiley & Sons, New York (1968)
- [28] Flounders, C.A., Lin, X., Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Comp. and Chem. Eng.* 28, 2109-2129 (2004)
- [29] Fathabadi, H., Hosseini, S. A., Maximum Flow Problem on Dynamic Generative Network Flows with Time-Varying Bounds, *Journal of Applied Mathematical Modelling* 34 2136–2147 (2010)
- [30] Fleischer, L., Universally maximum flow with piecewise-constant capacities. *Networks* 38(3) 115–125 (2001)
- [31] Ford, L., Fulkerson, D., Constructing Maximal Dynamic Flows from Static Flows. *Oper. Res.* 6 419–433 (1958)
- [32] Ford, L., Fulkerson, D., *Flows in Networks*, Princeton University Press (1962)
- [33] Grigsby, L., *The Electric Power Engineering Handbook*, USA: CRC Press (2001)
- [34] Hoppe, B., Efficient dynamic network flow algorithms. PhD Thesis, Cornell University (1995)
- [35] Hosseini, S.A., Dynamic Generative Network Flows, Master Thesis, University of Tehran, Iran (2009)

- [36] Hosseini, S.A., A Model-Based Approach and Analysis for Multiperiod Networks, *J Optim Theory Appl.* 157 486–512 (2013)
- [37] Hosseini, S.A., Sahin G, Unluyurt, T., A Decomposition-Based Solution Method for the Multiperiod Multiproduct Distribution Planning Problem, *Journal of Applied Mathematics*, vol. 2014 (2014) (WOS:000343540900001)
- [38] Hosseini, S.A., The Minimum Cost Flow Problem in Dynamic Multi Generative Network Flows. DM 10 Abstracts, The SIAM Conference on Discrete Mathematics. Hyatt Regency Austin, USA (2010)
- [39] Hosseini, S.A., Fathabadi, H., Minimum Cost Flow Problem on Dynamic Multi Generative Networks, *Journal of Algorithmic Operations Research* 5 39–48 (2010)
- [40] Hosseini, S.A., An Introduction to Dynamic Generative Networks: Minimum Cost Flow, *App. Math. Model.* 35 5017-5025 (2011)
- [41] Hosseini, S.A., Dashty, F., Determining the Optimal Flows in Zero-Time Dynamic Networks. *J. Math. Model. Algorithms* 11(2) 105-117 (2012)
- [42] Hartman, J.K., Lasdon, L.S., A generalized upper bounding algorithm for multiproduct network flow problems, *Networks* 1(4) 333–354 (1971)
- [43] Hall, J.A., McKinnon, K., Hyper-sparsity in the revised simplex method and how to exploit it. *Computational Optimization and Applications* 32(3) (2005) 259-283
- [44] Ho, J.K., Loute, E., An advanced implementation of the Dantzig-Wolfe decomposition algorithm for linear programming. *Mathematical Programming* 20 303–326 (1981)
- [45] Ho, J.K., Sundarraj, R.P., DECOMP: An implementation of Dantzig-Wolfe decomposition for linear programming, *Lecture Notes in Economics and Mathematical Systems* 338, Springer-Verlag (1989)
- [46] Hughes, Thomas P., *Networks of Power: Electrification in Western Society 1880-1930*, The Johns Hopkins University Press, Baltimore (1983)
- [47] Jackson, J.R., Grossmann, I. E., Temporal Decomposition Scheme for Nonlinear-Multisite Production Planning and Distribution Models. *Ind. and Eng. Chem. Res.* 42(13) 3045 – 3055 (2003)
- [48] Jones, K.L., Lustig, I.J., Farvolden, J. M., and Powell, W. B., Multiproduct network flows: The impact of formulation on decomposition. *Mathematical Programming* 62 95–117 (1993)
- [49] Jacob O., Energy losses of superconducting power transmission cables in the grid, *IEEE Transactions on Applied Superconductivity* 11 2375 (2001)
- [50] James K. Ho, Tak C. Lee, and R. P. Sundarraj, Decomposition of linear programs using parallel computation, *Mathematical Programming* 42 391-405 (1988)

- [51] Klein, P., Rao, S., Approximation Through Multicommodity Flow, Proceeding of the 31<sup>st</sup> Annual Symposium on Foundations of Computer Science 726-727 (1990)
- [52] Kondili, E., Pantelides, C., Sargent, W., A general algorithm for short-term scheduling of batch operations-I. MILP formulation. *Comp. and Chem. Eng.* 2 211-227 (1993)
- [53] Kernighan, B.W., Lin, S., An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal* 291-307 (1970)
- [54] K. Holmberg, D. Yuan, A Multi commodity Network Flow Problem with Side Constraints on Paths Solved by Column Generation, *INFORMS Journal of Computing* 1542-57 (2003)
- [55] Lei Y., Kimmo S., Jukka H., Ou T., Chinese industry from supply chain perspective-A case study of the major Chinese players, *Int. J. Production Economics* 115 374 -387 (2008)
- [56] Lindeman, R., The trophic-dynamic aspect of ecology, *Ecology* 23 (4) 399–417 (1942)
- [57] Lozovanu, D., *Optimization and Multiobjective Control of Time-Discrete Systems: Dynamic Networks and Multilayered Structures*, Springer (2010)
- [58] Luenberger, D.G., *Linear and Nonlinear Programming*. Addison-Wesley, Reading, MA (1984)
- [59] Maravelias, C.T., Grossmann, I.E., New Continuous-Time State Task Network Formulation for the Scheduling of Multipurpose Batch Plants. *Ind. and Eng. Chem. Res.* 42 3056-3074 (2003)
- [60] Mendoza, J.E., Lopez, M.E., Coello, C.A., Lopez, E.A., Microgenetic multiobjective reconfiguration algorithm considering power losses and reliability indices for medium voltage distribution network. *IET Gener. Transm. Distrib* 3 825–840 (2009)
- [61] Merlin, A., Search for a Minimal-Loss Operating Spanning Tree Configuration in an Urban Power Distribution System, Proceeding of the 1975 Fifth Power Systems Computer Conference (PSCC), Cambridge 1–18 (1975)
- [62] Mokhatab, S., Poe, W., *Handbook of Natural Gas Transmission and Processing*. Gulf Professional Publishing (2012)
- [63] Mouret, S., Grossmann, I.E., Pectiaux, P., Time Representations and Mathematical Models for Process Scheduling Problems. *Comp. Chem. Eng.* 35 1038-1063 (2011)
- [64] Moin, N.H., Salhi, S. Aziz, N.A.B., An efficient hybrid genetic algorithm for the multiproduct multiperiod inventory routing problem, *Int. J. Prod. Eco.* 133 334–343 (2011)
- [65] Nasrabadi, E., *Dynamic Flows in Time-varying Networks*. PhD thesis, Amirkabir University of Technology, Tehran, Iran (2009)

- [66] Neiro S.M.S. and Pinto, J. M., Lagrangean decomposition applied to multiperiod planning of petroleum refineries under uncertainty. *Latin Amer. App. Res.* 36 213 – 220 (2006)
- [67] Newman, M., Barabasi, A., Watts, D., *The Structure and Dynamics of Networks.* Princeton University Press (2006)
- [68] N. Hwang, R. Houghtalen, *Fundamentals of hydraulic Engineering Systems,* Prentice Hall, Upper Saddle River, NJ (1996)
- [69] Odum, E., Barrett, G., *Fundamentals of Ecology* (5th ed.). Brooks/Cole, a part of Cengage Learning, ISBN 0-534-42066-4 (2005)
- [70] Osiadacz, Andrzej, *Simulation and analysis of gas networks, Gas Engineering - Mathematical models,* E. & F.N. Spon Ltd, ISBN 0-419-12480-2 (1987)
- [71] Osiadacz, A., *Simulation and optimization of large systems, Large scale systems - Mathematical models,* Clarendon press, ISBN 0-19-853617-8 (1988)
- [72] Pantelides, C., *Unified Frameworks for the Optimal Process Planning and Scheduling.* Proceedings of the second conference on the foundations of computer aided operations p.253, Cache Publications
- [73] Proulx, S., Promislow, D., Phillips, P., *Network thinking in ecology and evolution,* *Trends in Ecology and Evolution* 20 (6) 345–353 (2005)
- [74] Royden, H., Fitzpatrick, P., *Real Analysis.* Prentice Hall (2010)
- [75] Rockfellar, R., *Network Flows and Monotropic Optimization.* John Wiley and Sons, New York (1984)
- [76] Rios, J., A general, parallel implementation of Dantzig–Wolfe decomposition. *ACM Trans. Math. Softw.* 39 3 (2013) (DOI: <http://dx.doi.org/10.1145/2450153.2450159>)
- [77] Rutenberg, D.P., *Generalized Networks, Generalized Upper Bounding and Decomposition of the Convex Simplex Method.* *Management Science* 16 (1970) 388-401
- [78] Schneur, R., Orlin, J., *A Scaling Algorithm for Multicommodity Flow Problems,* *Operations Research* 46 36–62 (2000)
- [79] Schrijver, A., *On the history of the transportation and maximum flow problems,* *Mathematical Programming* 91 (3): 437 445. doi:10.1007/s101070100259 (2002)
- [80] Skutella, M., *An Introduction to Network Flows Over Time,* in *Research Trends in Combinatorial Optimization,* W. Cook, L. Lovasz and J. Vygen, Sprngor, Berlin (2009)
- [81] Song, H., *A Study on the Dynamic Network Flow Problems: With Applications to the Dynamic Resource Managment in Logistics Networks.* VDM Verlag (2009)
- [82] Stefansson, H., Sigmarsdottir, S., Jensson, P., Shah, N., *Discrete and continuous time representations and mathematical models for large production scheduling problems: A case study from the pharmaceutical industry.* *Eur. J. Oper. Res* 215 383–392 (2011)

- [83] Tebboth, J. R., A computational study of Dantzig–Wolfe decomposition. Ph.D. thesis, University of Buckingham (2001)
- [84] Terrazas, S., Trotter, P., Grossmann, I.E., Temporal and spatial Lagrangean decompositions in multi-site, multiperiod production planning problems with sequence-dependent changeovers. *Comp. and Chem. Eng.* 35 (2011) 2913-2928
- [85] Voelker, M., Optimization of Slice Models. PhD Thesis, University of Wisconsin, Madison (2002)
- [86] Weil, R.L., Kettler, P. C., Rearranging matrices to block-angular form for decomposition (and other) algorithms. *Manag. Sci.* 18 (1971) 98-108
- [87] Zahorik, A., Thomas, L.J., Trigeiro, W., Network Programming Models for Production Scheduling in Multi-Stage, Multi-Item Capacitated Systems, *Management Science* 30 308-325 (1984)