OPTIMIZING INTEGRATED PRODUCTION, INVENTORY AND DISTRIBUTION PROBLEMS IN SUPPLY CHAINS

By
SANDRA DUNI EKŞIOĞLU

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
UNIVERSITY OF FLORIDA
2002
This work is dedicated to my family.
ACKNOWLEDGMENTS

I would like to thank the people who helped me complete the work contained in this dissertation. The help of my supervisors Panos Pardalos and Edwin Romeijn was of great value. I would like to thank Panos Pardalos for his technical advice, encouragement and insightful comments throughout my dissertation work. I would like to thank Edwin Romeijn for working closely with me. His unconditional support in solving many details surrounding this dissertation and his valuable feedbacks are deeply appreciated.

I extend my thanks to the members of my committee Selçuk Ercengüz and Joseph Geunes for their constructive criticism concerning the material of this dissertation. I also would like to express my appreciation to all my friends at the ISE department and in Gainesville for lightening up my life every day and making graduate school more fun than I am sure it is supposed to be. In particular I would like to thank Adriana and Jorge Jimenez, Mirela and Ilir Bejleri, Ebru and Deniz Erdoğmuş, Paveena Chaovalitwongse, Bayram Yildirim, Seviye Yörük, Mary and Kevin Taaffe, Olga Perdikaki, Hülya Emir, Sergiy Butenko and Lihui Bai.

I would like to express my special thanks to my parents Leonora and Perikli Duni and my brother Dhimitraq Duni. Their understanding and faith in me and my capabilities, their love, encouragement, and eternal support have motivated me all the time. Last but not least, I would like to thank my husband Burak for his love, patience and continuous support throughout all my years here at the University of Florida.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td></td>
<td>iii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td></td>
<td>vi</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Supply Chain Management</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Framework of This Study</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>Objectives and Summary</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>MULTI-FACILITY LOT-SIZING PROBLEM</td>
<td>12</td>
</tr>
<tr>
<td>2.1</td>
<td>Introduction</td>
<td>12</td>
</tr>
<tr>
<td>2.2</td>
<td>Literature Review</td>
<td>13</td>
</tr>
<tr>
<td>2.3</td>
<td>Problem Description</td>
<td>16</td>
</tr>
<tr>
<td>2.4</td>
<td>Extended Problem Formulation</td>
<td>21</td>
</tr>
<tr>
<td>2.5</td>
<td>Primal-Dual Based Algorithm</td>
<td>26</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Intuitive Understanding of the Dual Problem</td>
<td>27</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Description of the Algorithm</td>
<td>28</td>
</tr>
<tr>
<td>2.5.3</td>
<td>Running Time of the Algorithm</td>
<td>30</td>
</tr>
<tr>
<td>2.6</td>
<td>Cutting Plane Algorithm</td>
<td>31</td>
</tr>
<tr>
<td>2.6.1</td>
<td>Valid Inequalities</td>
<td>31</td>
</tr>
<tr>
<td>2.6.2</td>
<td>Separation Algorithm</td>
<td>32</td>
</tr>
<tr>
<td>2.6.3</td>
<td>Facets of Multi-Facility Lot-Sizing Problem</td>
<td>33</td>
</tr>
<tr>
<td>2.7</td>
<td>Dynamic Programming Based Heuristic</td>
<td>35</td>
</tr>
<tr>
<td>2.7.1</td>
<td>Introduction</td>
<td>35</td>
</tr>
<tr>
<td>2.7.2</td>
<td>Description of the Algorithm</td>
<td>37</td>
</tr>
<tr>
<td>2.7.3</td>
<td>Running Time of the Algorithm</td>
<td>40</td>
</tr>
<tr>
<td>2.8</td>
<td>Computational Results</td>
<td>42</td>
</tr>
<tr>
<td>2.9</td>
<td>Conclusions</td>
<td>52</td>
</tr>
<tr>
<td>3</td>
<td>EXTENSIONS OF THE MULTI-FACILITY LOT-SIZING PROBLEM</td>
<td>54</td>
</tr>
<tr>
<td>3.1</td>
<td>Multi-Commodity, Multi-Facility Lot-Sizing Problem</td>
<td>54</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Problem Formulation</td>
<td>56</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Linear Programming Relaxation</td>
<td>59</td>
</tr>
<tr>
<td>3.1.3</td>
<td>Valid Inequalities</td>
<td>61</td>
</tr>
<tr>
<td>3.1.4</td>
<td>Lagrangean Decomposition Heuristic</td>
<td>62</td>
</tr>
<tr>
<td>3.1.5</td>
<td>Outline of the Algorithm</td>
<td>67</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>3.1.6 Managerial Interpretation of the Decomposition</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td>3.1.7 Computational Results</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>3.2 Single-Commodity, Multi-Retailer, Multi-Facility Lot-Sizing Problem</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>3.2.1 Problem Formulation</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>3.2.2 Primal-Dual Algorithm</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>3.2.3 Intuitive Understanding of the Dual Problem</td>
<td>87</td>
<td></td>
</tr>
<tr>
<td>3.2.4 Outline of the Primal-Dual Algorithm</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>3.2.5 Computational Results</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>3.3 Multi Facility Lot-Sizing Problem with Fixed Charge Transportation</td>
<td>93</td>
<td></td>
</tr>
<tr>
<td>3.4 Conclusions</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>4 PRODUCTION-DISTRIBUTION PROBLEM</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>4.2 Problem Formulation</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td>4.3 Dynamic Slope Scaling Procedure</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>4.3.1 Multi-Commodity Network Flow Problem with Fixed Charge Cost Function</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>4.3.2 Single-Commodity Case</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>4.3.3 Multi-Commodity Case</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>4.3.4 Production-Distribution Problem</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>4.3.5 Extended Problem Formulation</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>4.4 A Lagrangean Decomposition Procedure</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>4.5 Computational Results</td>
<td>114</td>
<td></td>
</tr>
<tr>
<td>4.6 Conclusions</td>
<td>126</td>
<td></td>
</tr>
<tr>
<td>5 CONCLUDING REMARKS</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>APPENDICES</td>
<td>130</td>
<td></td>
</tr>
<tr>
<td>REFERENCES</td>
<td>138</td>
<td></td>
</tr>
<tr>
<td>BIOGRAPHICAL SKETCH</td>
<td>146</td>
<td></td>
</tr>
</tbody>
</table>
OPTIMIZING INTEGRATED PRODUCTION, INVENTORY AND DISTRIBUTION PROBLEMS IN SUPPLY CHAINS

By

Sandra Duni Ekşioğlu

December 2002

Chair: Panagote M. Pardalos
Co-Chair: H. Edwin Romeijn
Major Department: Industrial and Systems Engineering

The goal of this dissertation is the study of optimization models that integrate production, inventory and transportation decisions, in search of opportunities to improve the performance of a supply chain network. We estimate the total costs of a given design of a general supply chain network, including production, inventory and transportation costs. We consider production and transportation costs to be of fixed charge type. Fixed charge cost functions are linear functions with a discontinuity at the origin.

The main focus of this dissertation is the development of solution procedures for these optimization models. Their computational complexity makes the use of heuristics solution procedures advisable. One of the heuristics we propose is a Multi-Commodity Dynamic Slope Scaling Procedure (MCDSSP). This heuristic makes use of the fact that when minimizing a concave function over a convex set, an extreme point optimal solution exists. The same holds true for linear programs. Therefore, the concave cost function is approximated by a linear function and the corresponding linear program is solved. The slope of the linear function is updated iteratively until
no better solution is found. The MCDSSP can be used to solve any multi-commodity network flow problem with fixed charge cost functions.

We also develop a Lagrangean decomposition based heuristic. The subproblems from the decomposition have a special structure. One of the subproblems is the multi-facility lot-sizing problem that we study in detail in Chapter 2. The multi-facility lot-sizing problem is an extension of the economic lot-sizing problem. We add a new dimension to the classical problem, the facility selection decision. We provide the following heuristic approaches to solve this problem: dynamic programming, a primal-dual method, a cutting plane method and a linear programming based algorithm. We propose a set of valid inequalities and show that they are facet defining. We tested the performance of the heuristics on a wide range of randomly generated problems.

We also studied other extensions of the multi-facility lot-sizing problem. In Chapter 3 we analyze and provide solution approaches to the multi-commodity and multi-retailer (single-commodity) versions of the problem.
CHAPTER 1
INTRODUCTION

1.1 Supply Chain Management

Most companies nowadays are organized into networks of manufacturing and distribution sites that procure raw materials, process them into finished goods, and distribute the finished goods to customers. The goal is to deliver the right product to the right place at the right time for the right price. These production-distribution networks are what we call “supply chains.”

Supply Chain Management is a growing area of interest for both companies and researchers. It first attracted the attention of companies in the 1990s as they started to realize the potential cost benefits of integrating decisions with other members of their supply chain. The primary cost factors within a supply chain can be put into the categories of production, transportation and inventory. The signature of supply chain management is the integration of activities. Effective supply chain members invariably integrate the wishes and concerns of their downstream members into their operations while simultaneously ensuring integration with their upstream members. We concentrate on developing optimization tools to enable companies to take advantage of opportunities to improve their supply chain.

For many years companies and researchers failed to take an integrated view of the entire supply chain. They considered only one piece of the overall problem, such as production or distribution submodels. These submodels were optimized separately and the solutions were then joined together to establish operating policies.

A number of new developments have had an impact on many companies. For example, increased market responsiveness has intensified the inter-dependencies within the supply chain (Erengüç et. al [30]); technological innovations have shortened
the life span of manufacturing equipment, which in turn increases the cost of manufacturing capacity; internet has offered high speed communication (Geunes et. al [50]). These developments combined with increased product variety and decreased product volumes prompt companies to explore new ways of running their business. Experience has shown that a firm’s ability to manage its supply chain is a major source of competitive advantage. This realization is the single most important reason for the recent emphasis on supply chain management in industry and academia. To exploit these new opportunities to improve their profitability, the companies need decision support tools that provide evaluation of alternatives using optimization models.

Several examples can be found in the literature proving that models coordinating at least two stages of the supply chain can detect new opportunities for improving the efficiency of the supply chain. Chandra and Fisher [21] investigated the effect of coordinating production and distribution on a single-plant, multi-commodity, multi-period scenario. In this scenario, the plant produces and stores the products until they are delivered to the customers using a fleet of trucks. They proposed two solution approaches. The first approach solves the production scheduling and routing problems separately and the second approach considers both, production and routing decisions to be incorporated into the model. Their computational study showed that the coordinated approach can yield up to 20% in costs savings.

Anily and Federgruen [4] considered integrating inventory control and transportation planning decisions motivated by the trade-off between the size and the frequency of delivery. Their model considered a single warehouse and multiple retailers with inventories held only at the retailers who face constant demand. Burns et al. [18] investigated distribution strategies that minimize transportation and inventory costs. Successful applications of supply chain decision coordination were reported at Xerox [94] and Hewlett Packard [44]. As a result of coordinating the
decisions on inventories throughout the supply chain, they were able to reduce their inventory levels by 25%.

### 1.2 Framework of This Study

Companies deliver products to their customers using a logistics distribution network. Such networks typically consist of product flows from the producers to the customers through distribution centers (warehouses) and retailers. Companies generally need to make decisions on production planning, inventory levels, and transportation in each level of the logistics distribution network in such a way that customer’s demand is satisfied at minimum cost.

Coordinating decisions with other members with the aim of bigger profits and better customer service is a distinctive feature of supply chain management. Bhatnagar, Chandra and Goyal [13] distinguished between two broad levels of coordination. At the most general level, coordination can be seen in terms of integrating decisions of different functions (for example, facility location, inventory planning, production planning, distribution planning, etc.). They refer to this level of coordination as “general coordination.” At another level, the problem of coordination may be addressed by linking decisions within the same function at different echelons in the organization.

They classified the research on general coordination into three categories. These categories represent integration of decision-making related to

(i) Supply and production planning
(ii) Inventory and distribution planning
(iii) Production and distribution planning

Studies on coordination between supplier and buyer focused on determining the order quantity that is jointly optimal for both. The second category addresses the problem of coordinating inventory planning with distribution planning. This problem emerges when a number of customers must be supplied from one or more warehouses. The inventory and distribution planning problem decides on the replenishment policy
at the warehouse and the distribution schedule for the customers, so that the total of inventory and distribution costs are minimized. The trade-off is reducing the inventory costs versus an increase in the transportation costs. For example, shipping in smaller quantities and with high frequency reduces the inventory level at the warehouse, but causes higher transportation costs.

The third category of research concentrates on integrating production planning and distribution planning. The production planner is concerned with determining optimal production-inventory levels for each product in every period so that the total cost of production and inventory holding is minimized. On the other hand, the distribution planner must determine schedules for distribution of products to customers so that the total transportation cost is minimized. These two activities can function independently if there is a sufficiently large inventory buffer that completely decouples the two. However, this would lead to increased holding costs and longer lead times, since on one side the distribution planer, in order to minimize transportation costs, would prefer full truck shipments and minimum number of stops; and on the other side the production planner would prefer less number of machine setups. The pressure of reducing inventory and lead times in the supply chain forced companies to explore the issue of closer coordination between production and distribution.

Our work contributes to this last research category. We consider complex supply chains with centralized decisions on production-inventory and transportation. We model these supply chains as multi-commodity network flow problems with non-convex cost functions. In particular we use fixed charge cost functions to model production or transportation costs. We present several optimization techniques to solve these problems.

Related work is the research of King and Love [67] on the coordination of production and distribution systems at Kelly Springfield, a major tire manufacturer with four factories and nine major distribution centers located throughout the
United States. The authors described a coordinated system for manufacturing plants and distribution centers. Implementation of this system resulted in substantial improvements in overall lead times, customer service and average inventory levels. Annual costs were reduced by almost $8 million.

Blumenfeld et al. [16] reported on the successful implementation of an optimization model that synchronizes scheduling of production and distribution at the Delco electronics division of General motors. Implementation resulted in a 26% reduction in logistics costs.

Brown et al. [17] presented a successful implementation of an optimization model that coordinates decisions on production, transportation and distribution at Kellogg Company. Kellogg operates five plants in the United States and Canada, and it has seven core distribution centers. The model has been used for tactical and operational decisions since 1990. The operational version of the model determines everyday production and shipping quantities. The tactical version helps to establish budget and make capacity expansion and consolidation decisions. The operational model reduced production, inventory and distribution costs by approximately $4.5 million in 1995. The tactical model recently guided a consolidation of production capacity with a projected savings of $35 to $40 million per year.

Models coordinating different stages of the supply chain can be classified as strategic, tactical or operational (Hax and Candea [55]). The model we study helps the companies to make strategic decisions related to production, inventory and transportation. Several surveys in the literature address coordination issues. Vidal and Goetschalckx [102] addressed the issue of strategic production-distribution planning with emphasis on global supply chains. Beamon [12] presented models on multi-stage supply chain design and analysis. Erengüç et al. [30] surveyed models that integrate production and distribution planning. Thomas and Griffin [97] surveyed coordination models on strategic and operational planning.
1.3 Objectives and Summary

We propose a class of optimization models that consider coordination of production, transportation and inventory decisions in a particular supply chain. The supply chain we consider consists of a number of facilities and retailers. This model helps to estimate the total cost of a given logistics distribution network, including production, inventory holding and transportation costs. The evaluation takes place over a fixed, finite planning horizon of $T$ periods.

The particular scenario presented considers a set of facilities where $K$ different product types can be produced. Products are stored at the facilities until demand occurs. Moreover, retailers are supplied by the facilities and keep no inventories. We do not allow for transportation between facilities. An example of this particular scenario is the supply chain for very expensive items (such as cars). Often, it is not affordable for the retailers to keep inventories of expensive items, therefore they order the commodities as demand occurs (for example, Ford dealers keep no inventories for Corvette).

We consider production and transportation cost functions to be of the fixed charge form. Often in the literature production costs are modeled using this type of cost function. This is because of the special nature of the problem. Thus, whenever the production of a commodity takes place, a fixed charge is paid to set up the machines, plus a variable cost for every unit produced. Transportation costs also can be modelled using fixed charge costs, since usually there is a fixed charge (for example, the cost of the paperwork necessary) to initiate a shipment plus other costs that depend on the amount shipped.

The objective is to find the production, inventory, and transportation quantities that satisfy demand at minimum cost. We formulate this problem as a multi-commodity network flow problem on a directed, single source graph consisting of $T$ layers (Figure 1–1). Each layer of the graph represents a time period. In each layer,
a bipartite graph represents the transportation network between the facilities and retailers.

The set of nodes of the network consists of \( T \) copies of the \( F \) facilities and \( R \) retailers, as well as the source node. The set of arcs consists of production arcs (between the source node and a facility at a particular time period), transportation arcs (between a facility and a retailer at a particular time period), and inventory arcs (between two nodes corresponding to a particular facility in consecutive time periods). There are bundle capacities on the transportation and production arcs, and no capacities on the inventory arcs. In multi-commodity network flow problems, the bundle capacities tie together commodities by restricting the total flow (of all commodities) on an arc.

Our problem is related to the ones studied by Wu and Golbasi [106]; and Freling et al. [43]; and Romeijn and Romero Morales [90, 91]; and Romero Morales [83]; and Balakrishnan and Geunes [6]. In contrast to our model, Wu and Golbasi [106] assume a fixed-charge cost structure for production, but assume linear transportation costs. Also, they consider in their model the product structure of each end-item. On the other hand, Freling et al. [43], Romeijn and Romero Morales [90, 91] and Romero Morales [83] consider the case of a single commodity only, but account for the presence of so-called single-sourcing constraints, where each retailer should be supplied from a single facility only. Balakrishnan and Geunes [6] addressed a dynamic requirements-planning problem for two-stage multi-product manufacturing systems with bill-of-material flexibility (i.e., with options to use substitute components or subassemblies produced by an upstream stage to meet demand in each period at the downstream stage). Their model is similar to the multi-retailer and multi-facility lot-sizing problem we discuss in Chapter 3. However, their formulation is more general and consists of extra constraints (similar to the single-sourcing constraints) and extra components in the cost function (penalties for violating the single-sourcing constraints).
The supply chain optimization model we discuss is an $NP$-hard problem. Even the special case of the multi-commodity network flow problem with fixed charge costs, the single-period and single-commodity problem is $NP$-hard (Garey and Johnson [45]). The complexity of this problem led us to consider mainly heuristic approaches.

The difficulty in solving this problem motivated us to look into some special cases of the model. The knowledge we gained from analyzing the simpler problems gave us insights about how to approach the general model. Below we describe some of the special cases we identified.

The single-retailer, single-facility problem with only fixed charge production costs, reduces to the classical lot-sizing problem (Figure 1–2). The retailer in each period has to ship his demand from the facility. Therefore, the only decision to be made in this problem is for the facility to decide on its production schedule, when and how much to produce in every period (Figure 1–3). The same holds true when the model considers a single facility with multiple retailers.
Figure 1–2: Multi-period, single-facility, single-retailer problem

Figure 1–3: Multi-period, single-facility, single-retailer problem
The single-commodity, multi-facility, single-retailer problem with only production setup costs is discussed in Chapter 2 (Figure 1–4). We refer to this problem as the multi-facility lot-sizing problem. Wu and Golbasi [106] show that the uncapacitated version of this problem with holding costs not restricted in sign is \(NP\)-complete. We propose a number of heuristic approaches to solve the problem such as a dynamic programming based algorithm, a primal-dual heuristic and a cutting plane algorithm. We propose a set of valid inequalities and show that they are facets of the convex hull of the feasible region. We present a different formulation of the problem that we refer to as the extended problem formulation. The linear programming relaxation of the extended formulation gives lower bounds that are at least as good as the bounds from the linear programming relaxation of the original formulation.

In Chapter 3 we discuss two extensions of the multi-facility lot-sizing problem. First, we discuss the multi-commodity version of the problem. Later we present a model for the single-commodity, multi-retailer, multi-facility problem. We propose a Lagrangean decomposition scheme to solve the multi-commodity problem and a primal-dual algorithm to solve the multi-retailer problem.
Finally, in Chapter 4 we discuss the general production-distribution model we presented above. We propose a heuristic approach called the multi-commodity dynamic slope scaling procedure (MCDSSP). This is a general heuristic that can be used to solve any multi-commodity network flow problem with fixed charge cost function. The MCDSSP is a linear programming based heuristic. We solve the linear programming relaxation of the extended formulation to generate lower bounds and compare the quality of the solutions from MCDSSP to these bounds. For the production-distribution problem we also present a Lagrangean decomposition based algorithm. This algorithm decomposes the problem into two subproblems. The first subproblem further decomposes by commodity into $K$ single-commodity, multi-facility, multi-retailer problems, that we discuss in Chapter 3. The performance of these algorithms is tested on a large variety of test problems. We present extensive computational results.
CHAPTER 2
MULTI-FACILITY LOT-SIZING PROBLEM

2.1 Introduction

In this chapter we analyze and provide algorithms to solve the multi-facility economic lot-sizing problem. This is an extension of the well-known Economic Lot-Sizing problem. The economic lot-sizing problem can be described as follows: Given a finite time horizon $T$ and positive demands for a single item in each production period, find a production schedule such that the total costs are minimized. The customers' demand must be satisfied from production in the current period or by inventory from the previous periods (that is no backlogging is allowed). The two kinds of costs considered are production costs and holding costs.

Different from the classical economic lot-sizing problem, the multi-facility lot-sizing problem considers that demand can be satisfied via multiple facilities. This adds an extra dimension to the classical problem, the facility selection decision. Transportation costs, together with production and inventory costs are the biggest components of total costs. Thus, in contrast with the classical lot-sizing problem, this model, in deciding on a production schedule, considers not only production and inventory costs, but also transportation costs.

The multi-facility lot-sizing problem finds an optimal production, inventory and transportation schedule that satisfies demand at minimum cost. This problem has many practical applications. For example, a manufacturing company often has multiple facilities with similar production capacities. The need for cross facility capacity management is most evident in high-tech industries that have capital intensive equipment and a short technology life cycle. These companies often struggle with their production planning problem in their complex and rapidly changing
environment. To better utilize their capital-intensive equipments they are pressured to produce a variety of products in each of their production facilities. Coordinating the decisions on production, inventory and transportation among all the facilities reduces the costs relative to having each facility make its own decisions independently.

We present various solution approaches to solve this problem. The importance of the algorithms we propose is in the fact that these algorithms can be used as subroutines to solve more complex supply chain optimization problems. In Section 2.2 we present a literature review of economic lot-sizing and related problems. Section 2.3 gives the problem formulation and Section 2.4 discusses an extended formulation of the multi-facility lot-sizing problem. Its linear programming relaxation gives close-to-optimal solutions and the corresponding dual problem has a special structure. In Section 2.5 we discuss a primal-dual based algorithm. We present a set of valid inequalities for the problem and show that they are facets defining inequalities. The valid inequalities are used in the cutting plane algorithm discussed in Section 2.6. In Section 2.7 we discuss a dynamic programming based algorithm. Finally, Section 2.8 presents some of the computational results and Section 2.9 concludes this chapter.

2.2 Literature Review

This section presents a literature review on the single-item economic lot-sizing problem. The first contribution is the Economic Order Quantity (EOQ) model proposed by Harris [54] in 1913. This model considers a single commodity with a constant demand rate, production taking place continuously over time, and does not incorporate capacity limits.

A major limitation of the above model is the restriction that the demand is continuous over time and has constant rate. Manne [81] and Wagner and Whitin [104] considered the lot-sizing problem with a finite time horizon consisting of a number of discrete periods, each with its own deterministic and independent demand. This is the classic economic lot-sizing problem. Wagner and Whitin developed a dynamic
programming algorithm for the single-commodity, uncapacitated version of the problem. Their algorithm runs in $O(T^2)$.

This problem (apparently well solved since 1958) recently revealed a variety of new results. Practical reasons exist for the interest in this model. Almost thirty years later Wagelmans et al. [103], Aggarwal and Park [1], and Federgruen and Tzur [39] showed that the running time of the dynamic programming algorithm could be reduced to $O(T \log T)$ in the general case and to $O(T)$ when the costs have a special structure ($h_t + p_t \geq p_{t+1}$), also referred to as the absence of speculative motives.

The capacitated lot-sizing problem is $NP$-hard even for many special cases (Florian et al. [41] and Bitran and Yanasse [15]). In 1971, Florian and Klein presented a remarkable result. They developed an $O(T^4)$ algorithm for solving the capacitated lot-sizing problem with equal capacities in all periods. This result uses a dynamic programming approach combined with some important properties of optimal solutions to these problems. Recently, van Hoesel and Wagelmans [99] showed that this algorithm can be improved to $O(T^3)$ if backlogging is not allowed and the holding cost functions are linear.

Several solution approaches have been proposed for $NP$-hard special cases of the capacitated lot-sizing problem. These methods are typically based on branch-and-bound (see for instance, Baker et al. [5] and Erengüç and Aksoy [29]), dynamic programming (see for instance, Kirca [68] and Chen and Lee [23]) or a combination of the two (see for instance, Chung and Lin [25] and Lofti and Yoon [75]).

The multi-commodity version of the problem has attracted much attention. Manne [81] used the zero inventory (ZIO) property to develop a column generation approach to solve this problem. Barany et al. [10] solved the multi-commodity capacitated lot-sizing problem without set-up times optimally using a cutting plane procedure followed by branch and bound.

Other extensions to the classic economic lot-sizing problem consider set-up times, backorders and other factors. Zangwill [107] extended Wagner and Whitin’s
model to allow for backlogging and concave cost functions. Veinott [101] studied an uncapacitated model with convex cost structures. Trigeiro et al. [98] showed that capacitated lot-sizing problem with set-up times is much harder to solve than capacitated lot-sizing problem without set-up times. It is easy to check if the capacitated lot-sizing problem without set-up times has a feasible solution or not. This can be done by computing cumulative demand and cumulative capacity. When set-up times are considered, the feasibility problem is \( NP \)-complete. The bin packing problem is a special case of capacitated lot-sizing problem with set-up times (Garey and Johnson [45] p.226).

Much research on lot-sizing problems focused on determining a (partial) polyhedral description of the set of the feasible solutions and applying branch-and-cut methods (Pochet et al. [89], Leung et al. [74], Barany et al. [10]). The main motivation for studying the polyhedral structure of the single item lot-sizing problem is to use the results to develop efficient algorithms for problems such as the multi-commodity economic lot-sizing problem that contains this model as a substructure. However, the branch-and-cut approach has not (yet) resulted in competitive algorithms for the single-item lot-sizing problem itself. The reason is that generating a single cut could be as time consuming as solving the whole problem.

Barany et. al [9, 10] provided a set of valid inequalities for the single-commodity lot-sizing problem, showed that these inequalities are facets of the convex hull of the feasible region and furthermore, they showed that the inequalities fully describe the convex hull of the feasible region.

Pereira and Wolsey [88] studied a family of unbounded polyhedra arising in an uncapacitated lot-sizing problem with Wagner-Whitin costs. They completely characterized the bounded faces of maximal dimension and showed that they are integral. For a problem with \( T \) periods they derived an \( O(T^2) \) algorithm to express any point within the polyhedron as a convex combination of extreme points and
extreme rays. They observed that for a given objective function the face of optimal solutions can be found in $O(T^2)$.

Shaw and Wagelmans [93] considered the capacitated lot-sizing problem with piecewise linear production costs and general holding costs. They showed that this is an $NP$-hard problem and presented an algorithm that runs in pseudo-polynomial time.

Wu and Golbasi [106] considered a multi-facility production model where a set of items is to be produced in multiple facilities over multiple periods. They analyzed in depth the product-level (single-commodity, multi-facility) subproblem. They prove that general-cost version of this uncapacitated subproblem is $NP$-complete. They developed a shortest path algorithm and showed that it achieves optimality under special cost structures.

Our study is closely related to all the work we mention in this section. The multi-facility lot-sizing problem is an extension of the lot-sizing problem. We add to the classical model the facility selection decision and other than production and inventory costs, we consider transportation costs as well. The results found and the algorithms developed for the single-commodity lot-sizing problem enlighten us in our search for solution approaches to the multi-facility lot-sizing problem.

The set of valid inequalities that we propose in Section 2.6.1 are a generalization of the valid inequalities proposed by Barany et. al [10]. The extended problem formulation presented in Section 2.4 is inspired by a similar formulation proposed by van Hoesel [100] for the lot-sizing problem. However, our study is closely related to the work of Wu and Golbasi [106]. The multi-facility lot-sizing problem is the same model as the one in Wu and Golbasi, however we propose a wider (and different) range of solution approaches.

2.3 Problem Description

The multi-facility lot-sizing problem studied in this chapter can be formulated using the following notation.
Problem Data:

- $T$: number of periods in the planning horizon
- $F$: number of facilities
- $p_{it}$: production unit cost at facility $i$ in period $t$
- $s_{it}$: production set-up cost at facility $i$ in period $t$
- $h_{it}$: inventory unit cost at facility $i$ in period $t$
- $c_{it}$: transportation unit cost at facility $i$ in period $t$
- $Q_{it}(q_{it})$: production cost function at facility $i$ in period $t$
- $b_t$: demand in period period $t$

Decision Variables:

- $q_{it}$: number of items produced at facility $i$ in period $t$
- $x_{it}$: number of items transported from facility $i$ in period $t$
- $I_{it}$: number of items in the inventory at facility $i$ in the end of period $t$

The objective is to find a minimum cost production, inventory and transportation plan to fulfill demand when the set-up costs and production, inventory, transportation unit costs can vary from period to period, from facility to facility. The model assumes no starting or ending inventory.

This problem can be formulated as a network flow problem as follows:

$$\text{minimize } \sum_{i=1}^{F} \sum_{t=1}^{T} (Q_{it}(q_{it}) + c_{it}x_{it} + h_{it}I_{it})$$

subject to

1. $I_{it-1} + q_{it} = x_{it} + I_{it}$ \quad $i = 1,\ldots,F; t = 1,\ldots,T$ (2.1)
2. $\sum_{i=1}^{F} x_{it} = b_t$ \quad $t = 1,\ldots,T$ (2.2)
3. $I_{i0} = 0$ \quad $i = 1,\ldots,F$ (2.3)
4. $q_{it}, x_{it}, I_{it} \geq 0$ \quad $i = 1,\ldots,F; t = 1,\ldots,T$ (2.4)

Equation (2.1) presents the flow conservation constraints for each facility in each time period, and (2.2) presents the flow conservation constraints for the demand in every time period. The flow conservation constraint for the source, $\sum_{i=1}^{F} \sum_{t=1}^{T} q_{it} = \sum_{t=1}^{T} b_t$, is not included in the formulation because it is implied by (2.1) and (2.2).
The structure of this network is illustrated in Figure 2–1. The set of nodes in this network consists of $T$ copies of each of the $F$ facilities and the demand node, as well as a source node. Each layer of the network represents a time period. The set of arcs consists of $F \times T$ production arcs, $F \times T$ transportation arcs and $F \times (T - 1)$ inventory arcs. The source node $s$ supplies the total demand for the planning horizon. The production arcs connect the source to each facility, in each time period. The inventory arcs connect the same facility in successive periods. Transportation arcs connect facilities to the demand node in each period. The production cost function is a fixed charge cost function, thus, if production in a time period is initiated, the set-up cost plus production cost for each unit produced has to be paid.

$Q_{it}(q_{it}) = \begin{cases} p_{it}q_{it} + s_{it} & \text{if } q_{it} > 0 \\ 0 & \text{otherwise} \end{cases}$ for $i = 1, \ldots, F; t = 1, \ldots, T$.

The standard mixed-integer linear programming (MILP) formulation of this function can be obtained by introducing a binary set-up variable $y_{it}$ corresponding to each production arc. The production cost function can then be replaced by
$$Q_{it}(q_{it}) = p_{it}q_{it} + s_{it}y_{it} \quad \text{for } i = 1, \ldots, F; t = 1, \ldots, T,$$

where,

$$y_{it} = \begin{cases} 
1 & \text{if } q_{it} > 0 \\
0 & \text{otherwise.} 
\end{cases}$$

The MILP formulation of the multi-facility lot-sizing problem then reads as follows:

$$\text{minimize } \sum_{i=1}^{F} \sum_{t=1}^{T} (p_{it}q_{it} + s_{it}y_{it} + c_{it}x_{it} + h_{it}I_{it})$$

subject to

$$I_{it-1} + q_{it} = x_{it} + I_{it} \quad i = 1, \ldots, F; t = 1, \ldots, T \quad (2.5)$$

$$\sum_{i=1}^{F} x_{it} = b_{t} \quad t = 1, \ldots, T \quad (2.6)$$

$$q_{it} \leq b_{t}y_{it} \quad i = 1, \ldots, F; t = 1, \ldots, T \quad (2.7)$$

$$q_{it}, x_{it}, I_{it} \geq 0 \quad i = 1, \ldots, F; t = 1, \ldots, T \quad (2.8)$$

$$y_{it} \in \{0, 1\} \quad i = 1, \ldots, F; t = 1, \ldots, T, \quad (2.9)$$

where, $b_{tT}$ presents the demand during the time periods $t \ (t = 1, \ldots, T)$ to $T$, (i.e. $b_{tT} = \sum_{r=t}^{T} b_{r}$). Standard solvers such as CPLEX can be used to solve formulation (MF*) of the multi-facility lot-sizing problem.

The set of constraints (2.5) show that the number of items produced in the current period together with the inventory from previous periods, should be equal to the ending inventory plus the amount shipped to the retailer. Together with (2.8) they imply

$$I_{it} = \sum_{\tau=1}^{t} (q_{i\tau} - x_{i\tau}) \quad \text{for } i = 1, \ldots, F; t = 1, \ldots, T,$$

and using (2.6),

$$\sum_{i=1}^{F} I_{it} = \sum_{i=1}^{F} \sum_{\tau=1}^{t} q_{i\tau} - b_{1t} \quad \text{for } t = 1, \ldots, T.$$
Using these facts, the inventory variables can be eliminated from the formulation, reducing the size of the problem. The following is the MILP formulation of our problem without inventory variables.

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{F} \sum_{t=1}^{T} (p_{it}'q_{it} + s_{it}y_{it} + c_{it}'x_{it}) \\
\text{subject to} & \quad (R-MF^*) \\
\sum_{i=1}^{F} \sum_{\tau=1}^{t} q_{i\tau} & \geq b_{1t} \quad t = 1, \ldots, T \quad (2.10) \\
\sum_{\tau=1}^{t} q_{i\tau} & \geq \sum_{\tau=1}^{t} x_{i\tau} \quad i = 1, \ldots, F; t = 1, \ldots, T \quad (2.11) \\
\sum_{i=1}^{F} x_{it} & = b_{it} \quad t = 1, \ldots, T \quad (2.12) \\
q_{it} & \leq b_{iT}y_{it} \quad i = 1, \ldots, F; t = 1, \ldots, T \quad (2.13) \\
x_{it}, q_{it} & \geq 0 \quad i = 1, \ldots, F; t = 1, \ldots, T \quad (2.14) \\
y_{it} & \in \{0, 1\} \quad i = 1, \ldots, F; t = 1, \ldots, T \quad (2.15)
\end{align*}
\]

where \( p_{it}' = p_{it} + \sum_{\tau=t}^{T} h_{i\tau} \) and \( c_{it}' = c_{it} - \sum_{\tau=t}^{T} h_{i\tau} \).

**Proposition 2.3.1** (Non-splitting property): There exists an optimal solution to the uncapacitated, single-commodity, multi-facility lot-sizing problem such that the demand in period \( t \) is satisfied from either production or the inventory of exactly one of the facilities.

**Proof:** The multi-facility lot-sizing problem minimizes a concave cost function over a bounded convex set, therefore its optimal solution corresponds to a vertex of the feasible region. Let \((q^*, y^*, x^*, I^*)\) be an optimal solution to the multi-facility lot-sizing problem. In an uncapacitated network flow problem, a vertex is represented by a tree solution. The tree representation of the optimal solution implies that demand in every time period will be satisfied by exactly one of the facilities. In other words, \( x_{it}^* x_{jt}^* = 0 \) for \( i \neq j \) and \( t = 1, 2, \ldots, T \). Furthermore, for each facility, in each time period if the inventory level is positive, there will be no production, and vice versa.
Thus, zero inventory policy holds for this problem \((q^*_it^*I^*_{i,t-1} = 0, \text{ for } i = 1, \ldots, F, t = 1, \ldots, T)\). This completes the proof that in an optimal solution to our problem demand is satisfied from either production or the inventory of exactly one of the facilities.

Another characteristic of an optimal solution to the multi-facility lot-sizing problem is the following: Every facility in a given time period either does not produce, or produces the demand for a number of periods (the periods do not need to be successive). This property can easily be derived from Proposition 2.3.1 and the tree representation of an optimal solution. Figure 2–8 presents an example of such an extreme flow solution, where demands \(b_1\) and \(b_3\) are satisfied from production at facility 1 in period 1 and demand \(b_2\) is satisfied from production at facility 2 in period 2. Such a production schedule happens in case that the setup cost at facility 2 in period 2 is very small (as compared to the setup cost at facility 1 in periods 1, 2, 3 and setup cost at facility 2 in periods 1 and 3), but inventory holding cost at facility 2 in period 2 is very high.

For the multi-facility lot-sizing problem there exists an exact algorithm that is polynomial in the number of facilities and exponential in the number of periods (Wu and Golbasi [106]).

- Assign demands \(b_1, \ldots, b_T\) to facilities \(i = 1, \ldots, F\). This takes \(O(F^T)\).
- Given the assignment, solve for each facility an uncapacitated, single item, single facility lot-sizing problem. This takes \(F \times O(T \log T)\).

Wu and Golbasi [106] show that for the special case when inventory holding costs are not restricted in sign, the multi-facility lot-sizing problem is an NP-complete problem. They use a reduction from the uncapacitated facility location problem.

### 2.4 Extended Problem Formulation

The linear programming relaxation of (MF*) is not very tight. This is due to the constraints \(q_{it} \leq b_{iT} y_{it}\). In the linear programming relaxation of (MF*), the variable \(y_{it}\) determines the fraction of the demand from periods \(t\) to \(T\) satisfied from
production at facility \(i\) in period \(t\). \(b_T\) is usually a very high upper bound, since the production in a period rarely equals this amount. One way to tighten the formulation is to split the production variables \(q_{it}\) by destination into variables \(q_{it\tau}\) (\(\tau = t, \ldots, T\)), where \(\tau\) denotes the period for which production takes place (van Hoesel [100]). For the new variables, a trivial and tight upper bound is the demand in period \(\tau\) (i.e. \(b_\tau\)). The split of the production variables leads to the following identities:

\[
q_{it} = \sum_{\tau=t}^{T} q_{it\tau} \quad (2.16)
\]

\[
x_{it} = \sum_{s=1}^{t} q_{ist} \quad (2.17)
\]

\[
I_{it} = \sum_{s=1}^{t} \sum_{\tau=t}^{T} q_{is\tau} - \sum_{s=1}^{t} q_{ist} \quad (2.18)
\]

Replacing the production, transportation and inventory decision variables, and after re-arranging of terms, the objective function becomes

\[
\text{minimize} \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{\tau=t}^{T} (p_{it} + c_{it\tau} + \sum_{s=t+1}^{\tau} h_{is}) q_{it\tau} + s_{it} y_{it}]
\]

Replacing the decision variables in the constraints of the formulation (MF*) with the new variables, we obtain the following:

- Constraints (2.5) transform to

\[
\sum_{s=1}^{t-1} \sum_{\tau=t-1}^{T} q_{is\tau} - \sum_{s=1}^{t-1} q_{ist-1} + \sum_{\tau=t}^{T} q_{it\tau} = \sum_{s=1}^{t} q_{ist} + \sum_{s=1}^{t} \sum_{\tau=t}^{T} q_{is\tau} - \sum_{s=1}^{t} q_{ist}
\]

for \(i = 1, \ldots, F; t = 1, \ldots, T\).

After re-arranging terms, it follows that the constraints (2.5) are redundant.

- Constraints (2.6) transform to

\[
\sum_{i=1}^{F} \sum_{s=1}^{t} q_{ist} = b_t \quad \text{for } t = 1, \ldots, T.
\]
Constraints (2.7) transform to

\[ q_{it\tau} \leq b_{\tau}y_{it} \quad \text{for } i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T. \]

Finally, constraints (2.8) transform to

\[ q_{it\tau} \geq 0, \quad \text{for } i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T. \]

The extended problem formulation is the following:

\[
\text{minimize } \sum_{i=1}^{F} \sum_{t=1}^{T} \left[ \sum_{\tau=t}^{T} \bar{c}_{it\tau}q_{it\tau} + s_{it}y_{it} \right]
\]

subject to

\[
\sum_{i=1}^{F} \sum_{t=1}^{T} q_{it\tau} = b_{\tau} \quad \tau = 1, \ldots, T \quad (2.19)
\]

\[
q_{it\tau} - b_{\tau}y_{it} \leq 0 \quad i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T \quad (2.20)
\]

\[
q_{it\tau} \geq 0 \quad i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T \quad (2.21)
\]

\[
y_{it} \in \{0, 1\} \quad i = 1, \ldots, F; t = 1, \ldots, T, \quad (2.22)
\]

where \( \bar{c}_{it\tau} = p_{it} + c_{it\tau} + \sum_{s=t+1}^{\tau} h_{is} \). The above formulation resembles the model for the uncapacitated facility location problem. The relationship with the uncapacitated facility location problem is the following: consider \( W \times T \) facilities from which customers can be served. The decision to be made is whether or not to open (whether to get a shipment from) a certain facility \( it \). The cost for opening a facility (set-up the machines) is denoted by \( s_{it} \). The cost of serving customer (satisfying demand in period) \( \tau (\tau \geq t) \) is \( \bar{c}_{it\tau} \) per unit of demand. Our problem resembles the facility location problem, however it is not exactly the same problem. The fact that the costs assigned to the same facility in different periods are related to each other, makes the multi-facility lot-sizing problem a special case of the facility location problem.
The linear programming relaxation of (Ex-MF) replaces constraints \( y \in \{0, 1\} \) with \( y \geq 0 \). The binary variables \( y \) appear only in constraints (2.20), therefore
\[
y_{it} \geq \frac{q_{it\tau}}{b_{r}} \quad \text{for } i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T.
\]
Since set-up costs \( s_{it} \geq 0 \), and (Ex-MF) is a minimization problem, in an optimal solution we will have
\[
y_{it} = \frac{q_{it\tau}}{b_{r}} \quad \text{for } i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T.
\]
We know that the most \( q_{it\tau} \) can be is \( b_{r} \), and in this case \( y = 1 \); the least \( q_{it\tau} \) can be is zero and in this case \( y = 0 \). This result makes the constraint \( y_{it} \leq 1 \) redundant.

Formulation (LP-MF) is the linear programming relaxation of (Ex-MF).

\[
\text{minimize } \sum_{i=1}^{F} \sum_{t=1}^{T} \left[ \sum_{\tau=t}^{T} q_{it\tau} + s_{it} y_{it} \right]
\]
subject to
\[
\sum_{i=1}^{F} \sum_{t=1}^{T} q_{it\tau} = b_{r} \quad \tau = 1, \ldots, T
\]
\[
q_{it\tau} - b_{r} y_{it} \leq 0 \quad i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T
\]
\[
q_{it\tau} \geq 0 \quad i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T
\]
\[
y_{it} \geq 0 \quad i = 1, \ldots, F; t = 1, \ldots, T.
\]

In the special case of only one facility, our problem reduces to the economic lot-sizing problem. The linear programming relaxation of the extended formulation of this special case always has an integer solution. This result is due to Krarup and Bilde [69]. Although this does not hold true for the multi-facility lot-sizing problem, the lower bounds generated solving (LP-MF) are very good and the solutions are close to optimal. In fact, the lower bounds found solving (LP-MF) are much tighter than the lower bounds generated solving the linear programming relaxation of the original
formulation of this problem (MF*) (Proposition 2.4.1). A network representation of the extended formulation is given in Figure 2–2.

Figure 2–2: Network representation of extended formulation of a two-period, three-facility lot-sizing problem

Formulations (MF*) and (Ex-MF) of multi-facility lot-sizing problem are equivalent to each other. The notion of equivalence requires that the optimal solution to both formulations is the same (e. g. once the problem (Ex-MF) is solved in terms of \( q_{it\tau} \) variables, this solution should yield the optimal solution to problem (MF*)). For this to hold true, two conditions must be satisfied: (i) every solution of (Ex-MF) must correspond to a solution of (MF*) (i. e. no new solutions were created by redefining the decision variables of (MF*)), and (ii) there must be a solution in the feasible region of (Ex-MF) that corresponds to every extreme point in the convex hull of the set of feasible solutions to (MF*) (i. e. no solutions to formulation (MF*) were lost by redefining the variables). Condition (i) follows directly from the definition of \( q_{it\tau} \). A solution to (Ex-MF) can be directly translated into a solution to (MF*) using equations (2.16), (2.17) and (2.18). Condition (ii) is more difficult to argue. An extreme point of the convex hull of the feasible solutions to (MF*) is such that (a) it satisfies the zero inventory property (\( q_{itI_{i,t-1}} = 0 \) for \( i = 1, \ldots, F \) and \( t = 1, \ldots, T \));
(b) demand in period \( t (t = 1, \ldots, T) \) is satisfied from exactly one facility \( x_{it}x_{jt} = 0 \) for \( i, j = 1, \ldots, F, i \neq j \); (c) in period \( t (t = 1, \ldots, T) \) a facility either does not produce, or produces the demand for a number of periods, the periods do not need to be successive (Proposition 2.3.1). One can easily see (Figure 2-2) that an extreme point to (Ex-MF) satisfies exactly the same conditions. The correspondence between the extreme points of the two formulations shows that there is an extreme flow (tree solution) in the formulation (Ex-MF) for every extreme flow of the formulation (MF*).

**Proposition 2.4.1** The optimal cost of linear programming relaxation of the extended formulation of multi-facility lot-sizing problem (Ex-MF) is at least as high as the optimal cost of linear programming relaxation of original formulation (MF*).

**Proof:** Every feasible solution to the linear programming relaxation of extended formulation of multi-facility lot-sizing problem (Ex-MF) can be transformed to a solution to linear programming relaxation of original problem formulation (MF*) using equations (2.16), (2.17) and (2.18). It follows that the optimal solution of linear programming relaxation of (Ex-MF) can be transformed to a feasible solution (not necessary the optimal solution) to linear programming relaxation of (MF*). \( \square \)

### 2.5 Primal-Dual Based Algorithm

The dual of (LP-MF) has a special structure that allows us to develop a primal-dual based algorithm. The following is the formulation of the dual problem:

\[
\text{maximize } \sum_{t=1}^{T} b_t v_t
\]

subject to

\[
\sum_{\tau=t}^{T} b_\tau w_{it\tau} \leq s_{it} \quad i = 1, \ldots, F; t = 1, \ldots, T
\]

\[
v_\tau - w_{it\tau} \leq c_{it\tau} \quad i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T
\]

\[
w_{it\tau} \geq 0 \quad i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T.
\]

In an optimal solution to (D-MF) both constraints \( w_{it\tau} \geq 0 \) and \( w_{it\tau} \geq v_\tau - c_{it\tau} \) should be satisfied. Since \( w_{it\tau} \) is not in the objective function of (D-MF), we can
replace it with $w_{itr} = \max(0, v_{\tau} - c_{itr})$. This leads to the following condensed dual formulation:

$$\text{maximize } \sum_{t=1}^{T} b_t v_t$$

subject to

$$\sum_{\tau=1}^{T} b_{\tau} \max(0, v_{\tau} - c_{itr}) \leq s_{it} \quad i = 1, \ldots, F; t = 1, \ldots, T. \quad (D-MF^*)$$

Recall that the extended formulation of the multi-retailer lot-sizing problem is a special case of the facility location problem. The primal-dual scheme we discuss, in principle, is similar to the primal-dual scheme proposed by Erlenkotter [38] for the facility location problem. However, the implementation of the algorithm is different. Wagelmans et al. [103] consider the extended formulation of lot-sizing problem. They solve the corresponding dual problem in $O(T \log T)$. They show that the dual variables have the following property: $v_t \geq v_{t+1}$ for $t = 1, \ldots, T$. This property of the dual variables is used to show the dual ascent algorithm that they propose gives the optimal solution to the economic lot-sizing problem.

2.5.1 Intuitive Understanding of the Dual Problem

In this section we give an intuitive interpretation of the relationship between the primal-dual solutions of (Ex-MF). Suppose the linear programming relaxation of extended formulation (LP-MF) has an optimal solution $(q^*, y^*)$ that is integral. Let $\Theta = \{(i, t)|y^*_{it} = 1\}$ and let $(v^*, w^*)$ denote an optimal dual solution.

The complementary slackness conditions for this problem are

$$\begin{align*}
(C_1) \quad & y^*_{it} [s_{it} - \sum_{\tau=t}^{T} b_{\tau} w_{itr}] = 0 \quad \text{for } i = 1, \ldots, F; t = 1, \ldots, T \\
(C_2) \quad & q^*_{itr} [c_{itr} - v^*_{\tau} + w^*_{itr}] = 0 \quad \text{for } i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T \\
(C_3) \quad & w^*_{itr} [q^*_{itr} - b_{\tau} y^*_{it}] = 0 \quad \text{for } i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T \\
(C_4) \quad & v^*_{t} [b_{t} - \sum_i \sum_{\tau=1}^{t} q^*_{itr}] = 0 \quad \text{for } t = 1, \ldots, T.
\end{align*}$$
By conditions \((C_1)\), if a facility produces in a particular time period, the set-up cost must be fully paid (i. e. if \((i, t) \in \Theta\), then \(s_{it} = \sum_{\tau=t}^{T} b_{\tau} w_{\tau t}\)). Consider conditions \((C_3)\). Now, if facility \(i\) produces in period \(t\), but demand in that period is satisfied from inventory from a previous period \((q_{it}^* = 0 \text{ and } q_{it}^* - b_{it} y_{it}^* \neq 0)\), then \(w_{it}^* = 0\), which implies that the price paid for the product in a time period will contribute to the set-up cost of only the period in which the product is produced.

By conditions \((C_2)\), if \(q_{it}^* > 0\), then \(v_{it}^* = \tau_{it} + w_{it}^*\). Thus, we can think of \(v_{it}^*\) as the total cost (per unit of demand) in time \(\tau\); of this, \(\tau_{it}\) goes to pay for production and inventory holding costs, and \(w_{it}^*\) is the contribution to the production set-up cost.

### 2.5.2 Description of the Algorithm

The simple structure of the dual problem can be exploited to obtain near optimal feasible solutions by inspection. Suppose that the optimal values of the first \(k - 1\) dual variables \(v_1^*, \ldots, v_{k-1}^*\) are known. Then, to be feasible \(v_k\) must satisfy the following constraints:

\[
\begin{align*}
b_k \max(0, v_k - \tau_{it}) & \leq M_{it,k-1} = s_{it} - \sum_{\tau=t}^{k-1} b_{\tau} \max(0, v_{\tau}^* - \tau_{it}) \\
& \quad \text{for all } i = 1, \ldots, F \text{ and } t = 1, \ldots, k.
\end{align*}
\]

(2.23)

In order to maximize the dual problem we should assign to \(v_k\) the largest value satisfying these constraints. When \(b_k > 0\), this value is

\[
v_k = \min_{i,t \leq k} \left\{ \tau_{it} + \frac{M_{it,k-1}}{b_k} \right\}
\]

(2.24)

Note that \(M_{it,k-1} \geq 0\) implies \(v_k \geq \tau_{it}\). A dual feasible solution can be obtained simply by calculating the value of the dual variables sequentially using equation (2.24) (Figure 2–3), and a backward construction algorithm can then be used to generate primal feasible solutions (Figure 2–4). For the primal-dual set of solutions to be optimal, the complimentary slackness conditions should be satisfied. However, the primal and dual solutions found do not necessarily satisfy the complimentary slackness conditions.
\[ M_{it, t-1} = s_{it}, \ i = 1, \ldots, F, \ t = 1, 2, \ldots, T \]

for \( \tau = 1 \) to \( T \) do

if \( b_{\tau} = 0 \) then \( v_{\tau} = 0 \)
else

\[ v_{\tau} = \min_{i, t \leq \tau} \{ \bar{e}_{itr} + M_{it, \tau-1} / b_{\tau} \} \]

for \( i = 1 \) to \( F \) do

\[ w_{itr} = \max \{ 0, v_{\tau} - \bar{e}_{itr} \} \]

\[ M_{itr} = \max \{ 0, M_{itr-1} - b_{\tau} w_{itr} \} \]
enddo
enddo
endo

Figure 2–3: Dual algorithm

**Proposition 2.5.1** The solutions obtained using the primal and dual algorithms are feasible and they always satisfy the complimentary slackness conditions \((C_1)\) and \((C_2)\).

**Proof:** It is clear that primal and dual solutions generated are feasible by construction. Since the primal algorithm sets \( q_{itr} > 0 \) only when \( \bar{e}_{itr} - v_{\tau} - w_{itr} = 0 \), the solution satisfies conditions \((C_2)\). The dual algorithm constructs solutions by making sure that equations (2.23) are satisfied. Therefore, the dual solutions are always such that \( b_{\tau+1} w_{itr+1} \leq M_{itr} \). If \( M_{itr} = 0 \), then \( w_{itr+1} = 0 \), and since the dual algorithm sets \( M_{itr+1} = M_{itr} - b_{\tau+1} w_{itr+1} \), we also have \( M_{itr+1} = 0 \). Continuing this way it is clear that if at some point in the calculation we get \( M_{itr} = 0 \), we subsequently obtain the following:

\[ M_{itr} = M_{itr+1} = \ldots = M_{iT} = 0 \]

and

\[ w_{itr} = w_{itr+1} = \ldots = w_{iT} = 0. \]

The primal algorithm sets \( y_{it} = 1 \) only when \( M_{itr} = 0 \); this implies that conditions \((C_1)\) will always be satisfied.

Hence, in order to determine if the solution obtained using the primal-dual algorithm is optimal, one can check conditions \((C_3)\). Another way to check if the
\begin{align*}
y_{it} &= 0, q_{i\tau t} = 0, i = 1, \ldots, F, t = 1, \ldots, T; \tau \geq t \\
P &= \{t | b_t > 0, t = 1, \ldots, T\}
\end{align*}

**Start**: \(j = \max t \in P, t = 0\)

**Step 1**: for \(i = 1\) to \(F\) do

\begin{align*}
\text{repeat } t &= t + 1 \\
\text{until } M_{itj} &= 0 \text{ and } \bar{c}_{itj} - v_j - w_{itj} = 0 \\
y_{it} &= 1, i^* = i, t^* = t, \text{ go to Step 2}
\end{align*}

```
go to Step 3```

**Step 2**: for \(l = t^*\) to \(j\) do

\begin{align*}
\text{if } \bar{c}_{i^*tl} - v_l - w_{i^*tl} &= 0 \\
\quad P &= P - \{l\}, q_{i^*tl} = b_l
\end{align*}

```
endo```

**Step 3**: if \(P \neq \emptyset\) then go to **start**

```
else Stop
```

---

**Figure 2–4**: Primal algorithm

Solution is optimal is by comparing the objective function values from the primal and dual algorithms. Since the dual algorithm gives a dual feasible solution and the primal algorithm gives a primal feasible solution, at optimality, the two objective functions should be equal.

**2.5.3 Running Time of the Algorithm**

Here we discuss the running time of the primal-dual algorithm. The total number of logical and arithmetical operations performed in the dual algorithm is

- \(FT\) assignments during initialization
- \(2T + FT + FT^2\) comparisons
- \(FT + FT^2\) multiplications
- \(FT^2\) additions
- \(FT^2 + T\) assignments
- \(2FT^2\) subtractions

The total number of logical and arithmetical operations in the primal algorithm is

- \(FT + FT^2\) assignments during initialization
- \(T + 2FT\) comparisons
- \(T + 2FT\) assignments
- \(2T + FT^2\) subtractions

Thus, the running time of the primal-dual algorithm is \(O(FT^2)\).
Step 1: Solve LP relaxation of MF$^*$.

Step 2:
if $y^*$ is integral,
then STOP, the solution is optimal
else
solve the separation problem
if no violated inequality is found
STOP
else
add the valid inequality to (LP-MF), go to Step 1.

Figure 2–5: Cutting plane algorithm

2.6 Cutting Plane Algorithm

In this section we derive a set of valid inequalities for the multi-facility economic lot-sizing problem. We show that these inequalities are facets of the feasible region. The inequalities are then used in a cutting plane algorithm that finds tight lower bounds.

Figure 2–5 presents the steps of the cutting plane algorithm. The algorithm stops when either (i) the optimal solution to (MF$^*$) is found, or (ii) if no more valid inequalities can be generated. Next we discuss in detail the valid inequalities, the separation algorithm, and we show that these inequalities are facets of the feasible region of (MF$^*$).

2.6.1 Valid Inequalities

Theorem 2.6.1: For any $1 \leq l \leq T$, $L = \{1, \ldots, l\}$, and $S \subseteq L$:

$$\sum_{i=1}^{F} \left( \sum_{t \in S} q_{it} + \sum_{t \in L \setminus S} b_t y_{it} \right) \geq b_l$$

(2.25)

are valid inequalities for the multi-facility economic lot-sizing problem.

Proof: In order to prove that (2.25) are valid inequalities we should prove that they are satisfied by all feasible solutions with integer $y$. Consider formulation (MF$^*$). Given a solution $(q, x, I, y)$, let $y_{it} = 0$ for $t \in L \setminus S$. Inequality $q_{it} \leq b_t y_{it}$ implies that if $y_{it} = 0$, then $q_{it} = 0$. Thus,
Consider constraints (2.5). Summing up these constraints over all \( i = 1, \ldots, F \) we get

\[
\sum_{i=1}^{F} \left( \sum_{t \in S} q_{it} + \sum_{t \in L \setminus S} b_{lt} y_{it} \right) = \sum_{i=1}^{F} \sum_{t \in S} q_{it} = \sum_{i=1}^{F} \sum_{t=1}^{l} q_{it}.
\]

Since \( I_{lt} \geq 0 \) for all \( l = 1, \ldots, T \), and \( \sum_{i=1}^{F} x_{it} = b_t \) for all \( t = 1, \ldots, T \)

\[
\sum_{i=1}^{F} \sum_{t=1}^{l} q_{it} \geq b_{lt} \quad \text{for} \quad l = 1, \ldots, T.
\]

As a results, solutions such that \( y_{it} = 0 \) for \( t \in L \setminus S \), satisfy the inequality (2.25). Let \( t' = \arg\min \{ t \in L \setminus S, y_{it} = 1 \} \). Then, \( y_{it} = q_{it} = 0 \) for \( t \in (L \setminus S) \cap \{1, \ldots, t' - 1\} \).

Hence,

\[
\sum_{i=1}^{F} \left( \sum_{t \in S} q_{it} + \sum_{t \in L \setminus S} b_{lt} y_{it} \right) \geq \sum_{i=1}^{F} \sum_{t=1}^{t'-1} q_{it} + \sum_{i=1}^{F} \sum_{t=1}^{l} q_{it} \geq b_{1,t' - 1} + b_{lt} = b_{lt}.
\]

This completes the proof that every feasible solution with integer \( y \) will satisfy the valid inequalities (2.25).

The intuition behind the inequalities (2.25) is as follows: Assume that no production takes place in the periods in \( L \setminus S \). Then the full demand \( b_{lt} \) has to be produced in the periods in \( S \), giving \( \sum_{i=1}^{F} \sum_{t \in S} q_{it} \geq b_{lt} \). Now, suppose that we do produce in some of the periods in \( L \setminus S \), and let period \( k \) be the first such period. The production for periods \( 1, \ldots, k - 1 \) then has to be done in periods in \( S \). It is possible that the remaining demands \( b_{kl} \) is produced in a single period in \( L \setminus S \), which explains the coefficients of the \( y \) variables.

### 2.6.2 Separation Algorithm

Let \((q^*, x^*, I^*, y^*)\) be the solution to the linear programming relaxation of \((MF^*)\). If this solution satisfies the integrality constraints \((y \in \{0,1\})\), this is the solution to \((MF^*)\) as well. However, if it does not, we have to identify a valid inequality that cuts
off this solution from the set of feasible solutions to linear programming relaxation of \( (MF^*) \). A challenging problem is to identify the sets \( S \) and \( L \setminus S \) for which the valid inequality \( (2.25) \) is violated. An exponential number of sets \( S \) and \( L \setminus S \) exists, however, the following separation algorithm that runs in polynomial time \( (O(FT^2)) \) takes us through the steps needed to identify the sets \( S \) and \( L \setminus S \). Note that, for a given time period \( l \) \((l = 1, \ldots, T)\), this procedure identifies the valid inequality that is the most violated by the current solution.

For \( l = 1, \ldots, T \)

1. for \( i = 1, \ldots, F \) find \( S_l \subseteq L = \{1, \ldots, l\} \) where \( t \in S_l \) if \( \sum_{i=1}^{F} q_{it} \leq \sum_{i=1}^{F} b_{it} y_{it} \) and \( t \in L \setminus S_l \) if \( \sum_{i=1}^{F} q_{it} > \sum_{i=1}^{F} b_{it} y_{it} \)

2. check if
   \[
   \sum_{i=1}^{F} \left( \sum_{t \in S_i} q_{it}^* + \sum_{t \in L \setminus S_i} b_{it} y_{it}^* \right) < b_{1l}.
   \]

If so, the inequality is violated; otherwise for each \( l \), the following holds:

\[
\min_{S \subseteq L} \left\{ \sum_{i=1}^{F} \left( \sum_{t \in S} q_{it}^* + \sum_{t \in L \setminus S} b_{it} y_{it}^* \right) - b_{1l} \right\} = \sum_{i=1}^{F} \left( \sum_{t \in S_i} q_{it}^* + \sum_{t \in L \setminus S_i} b_{it} y_{it}^* \right) - b_{1l} > 0
\]

If no violation can be found, all valid inequalities of the form \( (2.25) \) are satisfied by the current solution \( (q^*, x^*, I^*, y^*) \). Therefore, either this solution is already an integer solution, or if this is not the case, the valid inequalities are not able to cut-off this non-integer solution from the feasible region of formulation \( (MF^*) \). There are two reasons for that to happen, either our valid inequalities are not facets of the convex hull of the feasible region, or in case that they are facets, they may not describe the whole feasible region, and other facets are needed as well. In Section 2.6.3 we will show that the valid inequalities \( (2.25) \) are indeed facets of the convex hull of the feasible region.

2.6.3 Facets of Multi-Facility Lot-Sizing Problem

Let \( \Phi \) be the feasible region of the multi-facility lot-sizing problem, let \( co(\Phi) \) be the corresponding convex hull, and let \( G = \{q, y \in co(\Phi) : \sum_{i=1}^{F} (\sum_{t \in S} q_{it} + \)
$$\sum_{t \in L \setminus S} b_{lt} y_{lt} = b_{lt}.$$  

$G$ consists of all the points in the convex hull of the feasible region that satisfy the valid inequality as equality. In other words $G$ consists of all the points in the convex hull of the feasible region that lie on the plane defined by the inequality (2.25). A common approach to show whether an inequality defines a facet of $\text{co}(\Phi)$ is to show that there exist precisely $\dim(\text{co}(\Phi))$ vectors on the boundary of $G$ that are affinely independent. Note that if the boundary of $G$ does not contain zero, this is equivalent to showing that there exist precisely $\dim(\text{co}(\Phi))$ vectors on the boundary of $G$ that are linearly independent (Nemhauser and Wolsey [86], Wolsey [105]). In our problem zero is not a feasible solution.

The following is a procedure that will help us to see whether inequalities (2.25) are facets of $\text{co}(\Phi)$. This means these inequalities are necessary if we wish to describe $\text{co}(\Phi)$ by a system of linear inequalities.

**Proposition 2.6.1**: If $b_t > 0$, $t = 1, \ldots, T$, the dimension of the feasible set of multi-facility economic lot-sizing problem is $\dim(\Phi) = 3TF - F - T - 1$.

**Proof**: We show that the dimension of the feasible region of the multi-facility lot-sizing problem is $3TF - F - T - 1$ in both cases, when formulations (MF*) or (R-MF*) are considered.

Consider formulation (MF*):

- A feasible solution in $\Phi$ has a total of $3TF$ of $q, x, y$ variables and $F(T - 1)$ of $I$ variables.
- The assumption that $b_t > 0$ implies that $b_1 > 0$ and $y_{1i} = 1$ for at least one $i = 1, \ldots, F$. This reduces the dimension of $\Phi$ by one.
- The equations $\sum_{i=1}^F x_{it} = b_t$ for $t = 1, \ldots, T$ reduce the dimension of $\Phi$ by $T$.
- The equations $I_{i,t-1} + g_{it} = x_{it} + I_{it}$ for $i = 1, \ldots, F$, and $t = 1, \ldots, T$ reduce the dimension of $\Phi$ by $TF$.

This shows that the dimension of the feasible region to formulation (MF*) is at most equal to $3TF - F - T - 1$.

Consider formulation (R-MF*):

- A feasible solution in $\Phi$ has a total of $3TF$ of $q, x, y$ variables.
- The assumption that $b_t > 0$, for $t = 1, \ldots, T$ implies that $b_1 > 0$ and $y_{1i} = 1$ for at least one $i = 1, \ldots, F$. Therefore, the dimension of $\Phi$ is reduced by one.
The equations \( \sum_{i=1}^{F} x_{it} = b_t \) for \( t = 1, \ldots, T \) reduce the dimension of \( \Phi \) by \( T \).

The equations \( \sum_{t=1}^{T} q_{it} = \sum_{t=1}^{T} x_{it} \) for \( i = 1, \ldots, F \) reduce dimension of \( \Phi \) by \( F \).

The dimension of the feasible region of formulation (R-MF*) is at most equal to \( 3TF - F - T - 1 \). However, in Theorem 2.6.2 we show that there exist \( 3TF - F - T \) affinely independent points in the convex hull of the feasible region of the multi-facility lot-sizing problem that satisfy the valid inequalities (2.25) to equality. This indicates that there exist \( 3TF - F - T \) affinely independent points in the feasible region of the multi-facility lot-sizing problem. This concludes our proof that the dimension of the feasible region of the multi-facility lot-sizing problem is \( 3TF - F - T - 1 \).

To illustrate that the above result is correct, we consider the following simple example. Barany et al. [10] show that the dimension of the feasible region to the economic lot-sizing problem is \( 2T - 2 \). The economic lot-sizing problem is a special case of the multi-facility problem with \( F = 1 \); therefore its dimension is \( 3T - 1 - T - 1 = 2T - 2 \).

The next step is to show that there are \( 3TF - F - T - 1 \) affinely independent points in \( \Phi \) that satisfy the valid inequality to equality.

**Theorem 2.6.2** If \( b_t > 0 \), for \( t = 1, \ldots, T \), the \((l, S)\) inequality defines a facet of \( \Phi \) whenever \( l < T, l \in S \) and \( L \setminus S \neq \emptyset \).

**Proof:** See Appendix.

### 2.7 Dynamic Programming Based Heuristic

#### 2.7.1 Introduction

Dynamic programming provides a framework for decomposing certain optimization problems into a nested family of subproblems. The nested substructure suggests a recursive approach for solving the original problem from the solutions of the subproblems. The recursion expresses an intuitive principle of optimality for sequential decision processes; that is, once we have reached a particular state, a necessary condition for optimality is that the remaining decisions must be chosen optimally with respect to that state (Nemhauser and Wolsey [86]).
Dynamic programming was originally developed for the optimization of sequential decision processes. A typical example that is used in the literature to explain how the dynamic programming algorithm works is the economic lot-sizing problem (Nemhauser and Wolsey [86] and Wolsey [105]). Consider the lot-sizing problem with $T$ time periods ($t=1,\ldots,T$). At the beginning of period $t$, the process is in state $s_{t-1}$, which depends only on the initial state $s_0$ (initial inventory $I_0$) and the decision variables $y_t$ and $q_t$ for $t=1,\ldots,t-1$. The contribution of the current state $t$ to the objective function depends on $I_{t-1}$. Let us denote by $v_t$ the value of the optimal decisions in periods $t,\ldots,T$

$$v_t(I_{t-1}) = \min_{q_t,y_t}(p_t q_t + s_t y_t + h_t (I_{t-1} + q_t - b_t) + v_{t+1}(I_{t-1} + q_t - b_t)) \quad (2.26)$$

The difficulty with this algorithm is that since demand in period $t$ is satisfied by production in periods $\tau \leq t$, it follows that the level of inventory in the end of period $t-1$ can be as large as $b_T$, and it appears that a large number of combinations of $(q_t, I_{t-1})$ must be considered to solve the problem.

Fortunately, the following properties of an optimal solution to the economic lot-sizing problem make the problem easier.

**Theorem 2.7.1** (Nemhauser and Wolsey [86]) An optimal solution to economic lot-sizing problem satisfies the following:

- $q_t I_{t-1} = 0$ for $t = 1, \ldots, T$
- If $q_t > 0$, then $q_t = \sum_{\tau=t}^{t'} b_{\tau}$ for $t \leq t' \leq T$
- If $I_{t-1} > 0$, then $I_{t-1} = \sum_{\tau=t}^{t'} b_{\tau}$ for $1 \leq t' \leq t$

From Theorem 2.7.1 it follows that $2(T-t)$ combinations of $(q_t, I_{t-1})$ must be considered to solve the recursive function (2.26). Thus the overall running time is $O(T^2)$, and recursive optimization yields a polynomial-time algorithm for the uncapacitated economic lot-sizing problem.
2.7.2 Description of the Algorithm

In this section we present a dynamic programming procedure to solve the multi-facility lot-sizing problem. This procedure is a heuristic, and therefore does not capture all solutions, possibly including the optimal solution to the problem.

In Section 2.3 we discussed the non-splitting property of optimal solutions to the multi-facility lot-sizing problem. It is important to note that a production, inventory and transportation plan is optimal if and only if the corresponding arcs with positive flow form an arborescence (rooted tree) in the network. Important implications of this result are the following: in an optimal production plan, the demand $b_t$ for a given period ($t = 1, \ldots, T$) will be produced in a single facility in a single time period; in every time period a facility either will not produce, or will produce the demand for a number of periods, and these time periods need not to be successive. Let $\Upsilon_t$ be the set of all periods covered by production at facility $i$ in period $t$ in an optimal arborescence. Then the optimal production plan is

$$q_{it} = \sum_{\tau=t}^{T} q_{it\tau} = \sum_{\tau \in \Upsilon_t} b_{\tau}, \quad i = 1, \ldots, F; t = 1, \ldots, T.$$  

The bold arcs in Figures 2–7 to 2–10 represent the extreme flows. As illustrated in Figures 2–7 and 2–8, an extreme flow to the multi-facility lot-sizing problem is not necessarily sequential. The example in Figure 2–8 shows an extreme point solution in which demands $b_1$ and $b_3$ are satisfied from the production at facility 1 in period 1, and the demand $b_2$ is satisfied from production at facility 2 in period 2.

Figure 2–6: Network representation of multi-facility lot-sizing problem with $T = 4$
Using this information we can now simplify the multi-facility lot-sizing problem to a shortest path problem in an acyclic network, say $G'$. We build $G'$ in the following way: let the total number of nodes in $G'$ be equal to $(T+1)$, one for each time period along with a dummy node $T+1$. Traversing arc $(\tau, \tau') \in G'$ represents the choice of producing in a single facility in a single time period $t = 1, \ldots, \tau$ to satisfy the demand in periods $\tau, \tau + 1, \ldots, \tau' - 1$. The cost of arc $(\tau, \tau')$ is calculated using the following cost function:

$$
g_{\tau, \tau'} = \min_{i=1, \ldots, F; 1 \leq t \leq \tau} s_{it} + c_{it, \tau} b_{\tau} + c_{it, \tau+1} b_{\tau+1} + \cdots + c_{it, \tau'-1} b_{\tau'-1}. \quad (2.27)
$$

Let $v_\tau$ be the minimum cost of a solution for period $\tau = 1, \ldots, T-1$ and $\tau' \geq \tau$. The recursion function for the multi-facility lot-sizing problem is

$$
v_\tau\left(\sum_{i=1}^{F} I_{i, \tau-1}\right) = \min_{\tau'}\left\{ g_{\tau, \tau'} + v_{\tau'}\left(\sum_{i=1}^{F} I_{i, \tau'-1}\right) \right\} \quad (2.28)
$$

and

$$
v_T\left(\sum_{i=1}^{F} I_{i, T-1}\right) = g_{T, T+1} \quad (2.29)
$$
The total number of arcs in $G'$ is equal to $T(T + 1)/2$. Given the costs $g_{\tau, \tau'}$ for every arc $(\tau, \tau') \in G'$, the recursive functions (2.28) and (2.29) will provide the optimal solution in $O(T^2)$. Every (directed) path in $G'$ that connects node 1 to $T + 1$, corresponds to a feasible solution to the original problem. The network $G'$ for a 4-period problem is presented in Figure 2–6.

**Theorem 2.7.2** For every path on $G'$ that connects node 1 to $T + 1$, there exists a corresponding an extreme point solution in the extended problem formulation (Ex-MF), and every sequential extreme point of (Ex-MF) is represented by a path on $G'$.

**Proof:** See Appendix.

Wu and Golbasi [106] propose a similar shortest path algorithm to solve the multi-facility lot-sizing problem. They showed that their algorithm gives the optimal solution to the problem if the following conditions hold: (i) no simultaneous production over more than one facility can take place in a given period. In other words $q_{it}q_{jt} = 0$ for $i, j = 1, \ldots, F$, $i \neq j$ and $t = 1, \ldots, T$. (ii) no production will be
scheduled at all if there is inventory carried over from previous period in one of the facilities. In other words \( q_{it}I_{jt-1} = 0 \) for \( i, j = 1, \ldots, F \) and \( t = 1, \ldots, T \).

\[
\sum_{t=1}^{T} b_t = 0
\]

Figure 2–9: Non-sequential flow: Case 1

These conditions obviously restrict the search for a solution to only sequential extreme flows. Furthermore, they investigate only part of the sequential extreme flows, the ones that satisfy the above conditions. Different from Wu and Golbasi, our procedure considers a wider range of extreme flows. We consider all the sequential extreme flows, although some of them may not satisfy conditions (i) and (ii). Figure 2–9 presents a sequential extreme flow that violates condition (i) and Figure 2–10 presents a sequential extreme flow that violates condition (ii).

2.7.3 Running Time of the Algorithm

The above dynamic programming algorithm to find the shortest path in the acyclic graph \( G' \) has running time of \( O(m) \), where \( m \) is the number of arcs in \( G' \).

Since the graph is complete,

\[
m = \sum_{t=1}^{T} t = \frac{T(T + 1)}{2}.
\]
Therefore, the running time of our algorithm will be \( O(c + T^2) \), where \( c \) presents the time it takes to calculate the costs of all arcs in \( G' \).

![Figure 2–10: Non-sequential flow: Case 2](image)

In calculating the cost of a particular arc \((\tau, \tau') \in G'\), we need to perform a certain number of comparisons, additions and multiplications.

- For an arc \((\tau, \tau')\) (where \(\tau = 1, \ldots, T\) and \(\tau' = \tau, \ldots, T + 1\)), the number of comparisons to be made in order to calculate its cost is equal to

\[
\tau F - 1.
\]

The total number of comparisons needed to generate all the arc costs for a problem with \(T\) time periods is

\[
Comp. = \sum_{\tau=1}^{T} \tau(\tau F - 1)
\]

thus, \( O(FT^3) \).

- The number of additions required to calculate the cost for arc \((\tau, \tau')\) is

\[
(\tau' - \tau + 1)\tau F.
\]
The total number of additions is

\[ \text{Add.} = \sum_{\tau=1}^{T} \sum_{\tau'=\tau+1}^{T+1} (\tau' - \tau + 1) \tau F \]

thus, \( O(FT^4) \).

- The number of multiplications required to calculate the cost for arc \((\tau, \tau')\) is

\[ (\tau' - \tau) \tau F. \]

The total number of multiplications is

\[ \text{Mlt.} = \sum_{\tau=1}^{T} \sum_{\tau'=\tau+1}^{T+1} (\tau' - \tau) \tau F \]

thus, \( O(FT^4) \).

This shows that the time complexity to solve the problem using the above dynamic programming algorithm is \( O(FT^4) \).

2.8 Computational Results

To test the performance of the algorithms discussed in this chapter we randomly generated a set of test problems and compared the computation times and solution quality to the general purpose solver CPLEX. The algorithms were compiled and executed on an IBM computer with 2 Power3 PC processors, 200 Mhz CPUs each.

The scope of our experiments, other than comparing the algorithms is to see how different factors (such as the ratio of set-up to variable cost, number of facilities, etc.) affect their performance. We first generate a nominal case problem as follows:

- Production set-up costs \(s_{it} \sim U[1200, 1500]\)
- Production variable costs \(p_{it} \sim U[5, 15]\)
- Holding costs \(h_{it} \sim U[5, 15]\)
- Demand \(b_t \sim U[5, 15]\)
- Number of facilities \(F = 150\)
- Number of periods \(T = 30\)

Most of the above parameters are the same as the ones used in Wu and Golbasi [106] for a related problem. To generate meaningful transportation variable costs, we
randomly generated from uniformly distributed points on a $[0, 10]^2$ square the facility and demand point locations, and calculated corresponding Euclidean distances. We assumed one to one correspondence between the Euclidean distances and the unit transportation costs.

Varying one or more factors from the nominal case, we generated five groups of test problems. In the first group of problems we change the level of production set-up costs from the nominal case to the following: $s_{it} \sim U[200, 300]$, $s_{it} \sim U[200, 900]$, $s_{it} \sim U[600, 900]$, $s_{it} \sim U[900, 1500]$, $s_{it} \sim U[1500, 2000]$, $s_{it} \sim U[2000, 3000]$, $s_{it} \sim U[3000, 6000]$, and $s_{it} \sim U[5000, 10000]$. These together with the nominal case problem give a total of 10 problem classes (problem classes 1 to 10).

In the second group of problems we change the length of the time horizon to 5, 10, 15, 20, 25, 35, 40 (problem classes 11 to 17). In the third group we change the number of facilities to 120, 130, 140, 160, 170, 180, 190, and 200 (problem classes 18 to 25). In the fourth group of problems the level of demand is changed to $b_t \sim U[20, 50]$, $b_t \sim U[50, 100]$, $b_t \sim U[100, 200]$, $b_t \sim U[200, 400]$, and $b_t \sim U[400, 1000]$ (problem classes 26 to 30). Finally, in the fifth group the level of holding costs is changed to $h_t \sim U[-20, -10]$, $h_t \sim U[-10, 10]$, $h_t \sim U[10, 20]$, $h_t \sim U[20, 40]$, and $h_t \sim U[40, 100]$ (problem classes 31 to 35).

For each problem class we generate 20 instances. The errors and running times we present for each problem class are the averages over the 20 problem instances. A summary of the results from the experiments are presented in Tables 2–2 to 2–5. We do not present the results from implementing the cutting plane algorithm, since in almost all of the problems CPLEX outperformed our algorithm in terms of solution quality and running times.

We would like to emphasize that the linear programming relaxation of the extended formulation and dual algorithm give lower bounds, while dynamic programming and primal algorithms give feasible solutions to the problems. The
Table 2–1: Problem characteristics

<table>
<thead>
<tr>
<th>Problems</th>
<th>Nodes</th>
<th>Arcs*</th>
<th>Arcs**</th>
<th>Arcs***</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,…,10</td>
<td>4,531</td>
<td>13,350</td>
<td>74,250</td>
<td>465</td>
</tr>
<tr>
<td>11</td>
<td>756</td>
<td>2,100</td>
<td>3,000</td>
<td>15</td>
</tr>
<tr>
<td>12</td>
<td>1,511</td>
<td>4,350</td>
<td>9,750</td>
<td>55</td>
</tr>
<tr>
<td>13</td>
<td>2,266</td>
<td>6,600</td>
<td>20,250</td>
<td>120</td>
</tr>
<tr>
<td>14</td>
<td>3,021</td>
<td>8,850</td>
<td>34,500</td>
<td>210</td>
</tr>
<tr>
<td>15</td>
<td>3,776</td>
<td>11,100</td>
<td>52,500</td>
<td>325</td>
</tr>
<tr>
<td>16</td>
<td>5,286</td>
<td>15,600</td>
<td>99,750</td>
<td>630</td>
</tr>
<tr>
<td>17</td>
<td>6,041</td>
<td>17,850</td>
<td>129,000</td>
<td>820</td>
</tr>
<tr>
<td>18</td>
<td>3,631</td>
<td>10,680</td>
<td>59,400</td>
<td>465</td>
</tr>
<tr>
<td>19</td>
<td>3,931</td>
<td>11,570</td>
<td>64,350</td>
<td>465</td>
</tr>
<tr>
<td>20</td>
<td>4,231</td>
<td>12,460</td>
<td>69,300</td>
<td>465</td>
</tr>
<tr>
<td>21</td>
<td>4,831</td>
<td>14,240</td>
<td>79,200</td>
<td>465</td>
</tr>
<tr>
<td>22</td>
<td>5,131</td>
<td>15,130</td>
<td>84,150</td>
<td>465</td>
</tr>
<tr>
<td>23</td>
<td>5,431</td>
<td>16,020</td>
<td>89,100</td>
<td>465</td>
</tr>
<tr>
<td>24</td>
<td>5,731</td>
<td>16,910</td>
<td>94,050</td>
<td>465</td>
</tr>
<tr>
<td>25</td>
<td>6,031</td>
<td>17,800</td>
<td>99,000</td>
<td>465</td>
</tr>
<tr>
<td>26,…,35</td>
<td>4,531</td>
<td>13,350</td>
<td>74,250</td>
<td>465</td>
</tr>
<tr>
<td>36,…,45</td>
<td>4,531</td>
<td>13,350</td>
<td>74,250</td>
<td>465</td>
</tr>
</tbody>
</table>

* Original formulation
** Extended formulation
*** Dynamic programming formulation
Table 2-2: Results from upper bound procedures for problem Groups 1 and 2

<table>
<thead>
<tr>
<th>Problem</th>
<th>Dynamic Prog.</th>
<th></th>
<th>Primal Algor.</th>
<th></th>
<th>CPLEX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error (%)</td>
<td>Time (sec)</td>
<td>Error (%)</td>
<td>Time (sec)</td>
<td>Time (sec)</td>
</tr>
<tr>
<td>1</td>
<td>0.00</td>
<td>1.39</td>
<td>1.15</td>
<td>0.17</td>
<td>29.55</td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
<td>1.39</td>
<td>0.85</td>
<td>0.17</td>
<td>31.57</td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
<td>1.39</td>
<td>5.34</td>
<td>0.17</td>
<td>64.56</td>
</tr>
<tr>
<td>4</td>
<td>0.00</td>
<td>1.39</td>
<td>6.25</td>
<td>0.17</td>
<td>86.63</td>
</tr>
<tr>
<td>5</td>
<td>0.00</td>
<td>1.39</td>
<td>10.26</td>
<td>0.17</td>
<td>99.35</td>
</tr>
<tr>
<td>6</td>
<td>0.00</td>
<td>1.39</td>
<td>15.14</td>
<td>0.17</td>
<td>120.37</td>
</tr>
<tr>
<td>7</td>
<td>0.00</td>
<td>1.39</td>
<td>15.15</td>
<td>0.16</td>
<td>143.32</td>
</tr>
<tr>
<td>8</td>
<td>0.00</td>
<td>1.39</td>
<td>14.03</td>
<td>0.17</td>
<td>192.07</td>
</tr>
<tr>
<td>9</td>
<td>0.00</td>
<td>1.39</td>
<td>16.44</td>
<td>0.16</td>
<td>255.40</td>
</tr>
<tr>
<td>10</td>
<td>0.00</td>
<td>1.39</td>
<td>33.06</td>
<td>0.16</td>
<td>376.69</td>
</tr>
<tr>
<td>11</td>
<td>0.00</td>
<td>0.01</td>
<td>47.37</td>
<td>0.01</td>
<td>0.93</td>
</tr>
<tr>
<td>12</td>
<td>0.00</td>
<td>0.03</td>
<td>30.61</td>
<td>0.02</td>
<td>6.30</td>
</tr>
<tr>
<td>13</td>
<td>0.00</td>
<td>0.13</td>
<td>17.72</td>
<td>0.04</td>
<td>16.88</td>
</tr>
<tr>
<td>14</td>
<td>0.00</td>
<td>0.34</td>
<td>15.25</td>
<td>0.05</td>
<td>36.66</td>
</tr>
<tr>
<td>15</td>
<td>0.00</td>
<td>0.75</td>
<td>12.25</td>
<td>0.07</td>
<td>65.33</td>
</tr>
<tr>
<td>16</td>
<td>0.00</td>
<td>2.63</td>
<td>11.17</td>
<td>0.13</td>
<td>144.59</td>
</tr>
<tr>
<td>17</td>
<td>0.00</td>
<td>4.31</td>
<td>10.58</td>
<td>0.17</td>
<td>203.35</td>
</tr>
</tbody>
</table>

Errors that we present give the deviation of the heuristics and lower bounds from the optimal solution found from solving the MILP formulation using CPLEX. The linear programming relaxation of extended formulation is solved using CPLEX callable libraries as well.

We measure the tightness of the lower bounds as follows:

\[
Error(\%) = \frac{\text{CPLEX} - \text{Lower bound}}{\text{CPLEX}} \times 100,
\]

and the quality of the heuristics using the following:

\[
Error(\%) = \frac{\text{Upper bound} - \text{CPLEX}}{\text{CPLEX}} \times 100.
\]

For all problem classes, except problem class 32, the dynamic programming algorithm gave the optimal solution in all 20 instances. The running time of this algorithm was at most 4.31 cpu seconds. On the other hand, the primal algorithm did not perform
well, as the errors went up to 52% for problem class 32. However, its running time for all problems was less than 1 cpu second. Both algorithms are much faster than CPLEX.

Table 2–3: Results from upper bound procedures for problem Groups 3, 4 and 5

<table>
<thead>
<tr>
<th>Problem</th>
<th>Dynamic Prog.</th>
<th>Primal Algor.</th>
<th>CPLEX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error (%)</td>
<td>Time (sec)</td>
<td>Error (%)</td>
</tr>
<tr>
<td>18</td>
<td>0.00</td>
<td>1.17</td>
<td>11.04</td>
</tr>
<tr>
<td>19</td>
<td>0.00</td>
<td>1.27</td>
<td>12.59</td>
</tr>
<tr>
<td>20</td>
<td>0.00</td>
<td>1.36</td>
<td>8.34</td>
</tr>
<tr>
<td>21</td>
<td>0.00</td>
<td>1.56</td>
<td>12.42</td>
</tr>
<tr>
<td>22</td>
<td>0.00</td>
<td>1.67</td>
<td>12.93</td>
</tr>
<tr>
<td>23</td>
<td>0.00</td>
<td>1.75</td>
<td>10.37</td>
</tr>
<tr>
<td>24</td>
<td>0.00</td>
<td>1.85</td>
<td>11.03</td>
</tr>
<tr>
<td>25</td>
<td>0.00</td>
<td>1.95</td>
<td>15.34</td>
</tr>
<tr>
<td>26</td>
<td>0.00</td>
<td>1.39</td>
<td>2.66</td>
</tr>
<tr>
<td>27</td>
<td>0.00</td>
<td>1.39</td>
<td>0.77</td>
</tr>
<tr>
<td>28</td>
<td>0.00</td>
<td>1.39</td>
<td>0.00</td>
</tr>
<tr>
<td>29</td>
<td>0.00</td>
<td>1.39</td>
<td>0.00</td>
</tr>
<tr>
<td>30</td>
<td>0.00</td>
<td>1.39</td>
<td>0.00</td>
</tr>
<tr>
<td>31</td>
<td>0.00</td>
<td>1.39</td>
<td>6.44</td>
</tr>
<tr>
<td>32</td>
<td>1.24</td>
<td>1.39</td>
<td>51.84</td>
</tr>
<tr>
<td>33</td>
<td>0.00</td>
<td>1.39</td>
<td>9.35</td>
</tr>
<tr>
<td>34</td>
<td>0.00</td>
<td>1.39</td>
<td>1.92</td>
</tr>
<tr>
<td>35</td>
<td>0.00</td>
<td>1.39</td>
<td>0.44</td>
</tr>
</tbody>
</table>

The linear programming relaxation of the extended formulation generated solutions that are less than 0.1% from optimal for all problem classes. The dual algorithm provided tight lower bounds as well. For all problem classes except problem class 32, it generated bounds that are less than 2% from optimal. However, the linear programming relaxation of the extended formulation took almost as much time as solving MILP formulation. The running time of the dual algorithm for all problem classes was less then 1 cpu second.

Based on the results presented in Tables 2–2 to 2–5, we can see that an effective algorithm to solve our problem would combine the dynamic programming algorithm to generate upper bounds and the dual algorithm to generate lower bounds.
Table 2–4: Results from lower bound procedures for problem Groups 1 and 2

<table>
<thead>
<tr>
<th>Problem</th>
<th>Linear Prog.</th>
<th>Dual Algor.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error (%)</td>
<td>Time (sec)</td>
</tr>
<tr>
<td>1</td>
<td>0.01</td>
<td>24.88</td>
</tr>
<tr>
<td>2</td>
<td>0.01</td>
<td>26.32</td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
<td>65.41</td>
</tr>
<tr>
<td>5</td>
<td>0.00</td>
<td>138.10</td>
</tr>
<tr>
<td>6</td>
<td>0.00</td>
<td>175.31</td>
</tr>
<tr>
<td>7</td>
<td>0.00</td>
<td>217.25</td>
</tr>
<tr>
<td>8</td>
<td>0.00</td>
<td>312.41</td>
</tr>
<tr>
<td>9</td>
<td>0.00</td>
<td>484.41</td>
</tr>
<tr>
<td>10</td>
<td>0.00</td>
<td>688.55</td>
</tr>
<tr>
<td>11</td>
<td>0.00</td>
<td>0.49</td>
</tr>
<tr>
<td>12</td>
<td>0.00</td>
<td>6.00</td>
</tr>
<tr>
<td>13</td>
<td>0.00</td>
<td>19.39</td>
</tr>
<tr>
<td>14</td>
<td>0.02</td>
<td>45.46</td>
</tr>
<tr>
<td>15</td>
<td>0.01</td>
<td>87.04</td>
</tr>
<tr>
<td>16</td>
<td>0.00</td>
<td>193.17</td>
</tr>
<tr>
<td>17</td>
<td>0.00</td>
<td>282.01</td>
</tr>
</tbody>
</table>

Now, we want to provide more insights into the problem characteristics that affect the performance of the heuristics. It has been shown in other studies (Hochbaum and Segev [60], Ekşioğlu et al. [34]) that the problem difficulty depends on the values of the fixed costs. In problem classes 1 to 10 the level of production set-up costs is increased (while everything else is the same). Thus, the time it took solving the MILP formulation of the problem, as well as the time it took solving the linear programming relaxation of extended formulation, increased as the level of fixed charge costs increased. However, the computational time of the dual-primal and dynamic programming algorithms were not affected by the change in setup costs. This can be explained by noting that their running time depends only on the number of facilities and the length of the time period.

From the results for problem groups 2 and 3, one can see that as the number of time periods and the number of facilities increase, the running time of the dynamic programming algorithm increased. Problem classes 17 and 25 have almost the same
Table 2–5: Results from lower bound procedures for problem Groups 3, 4 and 5

<table>
<thead>
<tr>
<th>Problem</th>
<th>Linear Prog.</th>
<th>Dual Algor.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error (%)</td>
<td>Time (sec)</td>
</tr>
<tr>
<td>18</td>
<td>0.00</td>
<td>85.30</td>
</tr>
<tr>
<td>19</td>
<td>0.00</td>
<td>92.83</td>
</tr>
<tr>
<td>20</td>
<td>0.00</td>
<td>113.25</td>
</tr>
<tr>
<td>21</td>
<td>0.01</td>
<td>153.80</td>
</tr>
<tr>
<td>22</td>
<td>0.00</td>
<td>164.49</td>
</tr>
<tr>
<td>23</td>
<td>0.00</td>
<td>190.94</td>
</tr>
<tr>
<td>24</td>
<td>0.01</td>
<td>213.88</td>
</tr>
<tr>
<td>25</td>
<td>0.01</td>
<td>248.09</td>
</tr>
<tr>
<td>26</td>
<td>0.01</td>
<td>38.39</td>
</tr>
<tr>
<td>27</td>
<td>0.01</td>
<td>23.56</td>
</tr>
<tr>
<td>28</td>
<td>0.00</td>
<td>16.75</td>
</tr>
<tr>
<td>29</td>
<td>0.00</td>
<td>14.13</td>
</tr>
<tr>
<td>30</td>
<td>0.00</td>
<td>12.57</td>
</tr>
<tr>
<td>31</td>
<td>0.01</td>
<td>17.31</td>
</tr>
<tr>
<td>32</td>
<td>0.16</td>
<td>19.55</td>
</tr>
<tr>
<td>33</td>
<td>0.00</td>
<td>99.44</td>
</tr>
<tr>
<td>34</td>
<td>0.00</td>
<td>50.44</td>
</tr>
<tr>
<td>35</td>
<td>0.00</td>
<td>25.14</td>
</tr>
</tbody>
</table>

number of arcs; however the time it took to solve problem class 17 is almost twice the time it took to solve problem class 25. The reason is that the number of time periods in problem class 17 is higher (the running time of this algorithm is $O(FT^4)$).

Another interesting observation is that all algorithms performed very poorly in solving problem class 32. In this problem class we have generated the holding costs in the interval $[-10, 10]$. For this problem class, the quality of both lower and upper bounds generated is worse when compared to the rest of the problems. The dynamic programming algorithms gave solutions that are 1.24% from optimal and primal algorithm gave solutions 13.19% from optimal. The lower bounds generated using linear programming relaxation of the extended formulation were 0.16% from optimal and the dual algorithm gave an average error of 51.84%. The reason for such a performance is the holding costs being not restricted in sign. Wu and Golbasi [106]
show that the multi-facility lot-sizing problem is $NP$-hard in case that the holding costs are not restricted in sign.

We wanted to further investigate the performance of the algorithms for the case when the holding costs are not restricted in sign. We generated a sixth group of problems that have holding costs uniformly distributed in the interval $[-10, 10]$. We re-ran problem classes 1 to 10 creating 10 new class problems (class problems 36 to 45).

Table 2–7: Results from lower bound procedures for problem Group 6

<table>
<thead>
<tr>
<th>Problem</th>
<th>Linear Prog.</th>
<th>Dual Algor.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error (%)</td>
<td>Time (sec)</td>
</tr>
<tr>
<td>36</td>
<td>0.00</td>
<td>12.33</td>
</tr>
<tr>
<td>37</td>
<td>0.00</td>
<td>13.67</td>
</tr>
<tr>
<td>38</td>
<td>0.05</td>
<td>15.68</td>
</tr>
<tr>
<td>39</td>
<td>0.00</td>
<td>18.03</td>
</tr>
<tr>
<td>40</td>
<td>0.16</td>
<td>18.98</td>
</tr>
<tr>
<td>41</td>
<td>0.10</td>
<td>18.04</td>
</tr>
<tr>
<td>42</td>
<td>0.16</td>
<td>17.18</td>
</tr>
<tr>
<td>43</td>
<td>0.39</td>
<td>22.82</td>
</tr>
<tr>
<td>44</td>
<td>0.23</td>
<td>47.38</td>
</tr>
<tr>
<td>45</td>
<td>0.00</td>
<td>83.41</td>
</tr>
</tbody>
</table>
Tables 2–6 and 2–7 present the results for the sixth group of problems. For this set of problems, the dynamic programming algorithm performed well. Its maximum error was less than 1.54% from optimal and the running time averaged 1.40 CPU seconds. The primal algorithm performed very poorly for this group of problems.

The linear programming relaxation of the extended formulation gave tight lower bounds. The maximum error presented for this group of problems is 0.39%. However, the running time of this algorithm is comparable to the time it took CPLEX to solve the corresponding MILP formulation of the problem. The dual algorithm also performed poorly.

In our final group of problems (problem group seven), we consider the demand to show seasonality pattern. Demands are generated as follows:

\[ b_t = 200 + \sigma z_t + \alpha \sin \left( \frac{2\pi}{d} (t + d/4) \right) \]

where,
- \( \sigma \) is the standard error of demand
- \( z_t \) i.i.d. standard normal random variable
- \( \alpha \) amplitude of the seasonal component
- \( d \) is the length of a seasonal cycle in periods

These demands are generated in the same way as in Baker et al. [5] and Chen et al. [22]. In our test problems we take \( \sigma = 67 \), \( \alpha = 125 \) and \( d = 12 \). Table 2–8 presents the characteristics of the problems generated. The errors presented in Table 2–9 are calculated as follows:

\[
\text{Gap (\%)} = \frac{\text{Upper Bound} - \text{Lower Bound from Dual Alg.}}{\text{Lower Bound from Dual Alg.}} \times 100.
\]

For problem classes 46 to 56 we compare the solutions from the dynamic programming and primal algorithms with the corresponding optimal solutions from CPLEX. For these problems, dynamic programming performed very well, as the maximum error presented is 0.065%. However, the running time of this algorithm is higher than the running time of CPLEX. It took CPLEX on average 53 cpu seconds
Table 2–8: Characteristics of problem Group 7

<table>
<thead>
<tr>
<th>Problem</th>
<th>Facilities</th>
<th>Periods</th>
<th>Nodes</th>
<th>Arcs*</th>
<th>Arcs**</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td>20</td>
<td>24</td>
<td>505</td>
<td>6,480</td>
<td>300</td>
</tr>
<tr>
<td>47</td>
<td>30</td>
<td>24</td>
<td>745</td>
<td>9,720</td>
<td>300</td>
</tr>
<tr>
<td>48</td>
<td>40</td>
<td>24</td>
<td>985</td>
<td>12,960</td>
<td>300</td>
</tr>
<tr>
<td>49</td>
<td>20</td>
<td>48</td>
<td>1,009</td>
<td>24,480</td>
<td>1,176</td>
</tr>
<tr>
<td>50</td>
<td>30</td>
<td>48</td>
<td>1,489</td>
<td>36,720</td>
<td>1,176</td>
</tr>
<tr>
<td>51</td>
<td>40</td>
<td>48</td>
<td>1,969</td>
<td>48,960</td>
<td>1,176</td>
</tr>
<tr>
<td>52</td>
<td>20</td>
<td>96</td>
<td>2,017</td>
<td>95,040</td>
<td>4,656</td>
</tr>
<tr>
<td>53</td>
<td>30</td>
<td>96</td>
<td>2,977</td>
<td>142,560</td>
<td>4,656</td>
</tr>
<tr>
<td>54</td>
<td>40</td>
<td>96</td>
<td>3,937</td>
<td>190,080</td>
<td>4,656</td>
</tr>
<tr>
<td>55</td>
<td>20</td>
<td>192</td>
<td>4,033</td>
<td>374,400</td>
<td>18,528</td>
</tr>
<tr>
<td>56</td>
<td>30</td>
<td>192</td>
<td>5,953</td>
<td>561,600</td>
<td>18,528</td>
</tr>
<tr>
<td>57</td>
<td>40</td>
<td>192</td>
<td>7,873</td>
<td>748,800</td>
<td>18,528</td>
</tr>
<tr>
<td>58</td>
<td>20</td>
<td>384</td>
<td>8,065</td>
<td>1,486,080</td>
<td>73,920</td>
</tr>
<tr>
<td>59</td>
<td>30</td>
<td>384</td>
<td>11,905</td>
<td>2,229,120</td>
<td>73,920</td>
</tr>
<tr>
<td>60</td>
<td>40</td>
<td>384</td>
<td>15,745</td>
<td>2,972,160</td>
<td>73,920</td>
</tr>
<tr>
<td>61</td>
<td>20</td>
<td>768</td>
<td>16,129</td>
<td>5,921,280</td>
<td>295,296</td>
</tr>
<tr>
<td>62</td>
<td>30</td>
<td>768</td>
<td>23,809</td>
<td>8,881,920</td>
<td>295,296</td>
</tr>
<tr>
<td>63</td>
<td>40</td>
<td>768</td>
<td>31,489</td>
<td>11,842,560</td>
<td>295,296</td>
</tr>
</tbody>
</table>

* Extended formulation
** Dynamic programming formulation

to run problem class 56, and for the same problem it took the dynamic programming algorithm on average 354 cpu seconds. The primal algorithm also performed very well. The maximum error presented is 0.352%. The running time of the primal algorithm was less then 1 cpu second for this set of problems.

For problem classes 57 to 63 we do not have the optimal solutions. CPLEX failed to solve these problems, because of their size. Therefore, we use the lower bounds generated from dual algorithm to calculate the error gaps. The dynamic programming algorithm gaves very good solutions. The maximum error gap was 0.129%, but the running time of the algorithm went as high as 115,453 cpu seconds. The primal algorithm gave a maximum gap of 0.268% and maximum running time 14.05 cpu seconds.
Table 2–9: Results for problem Group 7

<table>
<thead>
<tr>
<th>Problem</th>
<th>Dynamic Prog.</th>
<th>Primal Algor.</th>
<th>CPLEX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error (%)</td>
<td>Gap (%)</td>
<td>Time (sec)</td>
</tr>
<tr>
<td>46</td>
<td>0.000</td>
<td>0.028</td>
<td>0.08</td>
</tr>
<tr>
<td>47</td>
<td>0.000</td>
<td>0.016</td>
<td>0.12</td>
</tr>
<tr>
<td>48</td>
<td>0.004</td>
<td>0.005</td>
<td>0.17</td>
</tr>
<tr>
<td>49</td>
<td>0.001</td>
<td>0.018</td>
<td>1.08</td>
</tr>
<tr>
<td>50</td>
<td>0.002</td>
<td>0.013</td>
<td>1.63</td>
</tr>
<tr>
<td>51</td>
<td>0.008</td>
<td>0.009</td>
<td>2.17</td>
</tr>
<tr>
<td>52</td>
<td>0.004</td>
<td>0.005</td>
<td>15.49</td>
</tr>
<tr>
<td>53</td>
<td>0.004</td>
<td>0.021</td>
<td>23.25</td>
</tr>
<tr>
<td>54</td>
<td>0.002</td>
<td>0.027</td>
<td>31.16</td>
</tr>
<tr>
<td>55</td>
<td>0.065</td>
<td>0.067</td>
<td>235.32</td>
</tr>
<tr>
<td>56</td>
<td>0.048</td>
<td>0.054</td>
<td>353.44</td>
</tr>
<tr>
<td>57</td>
<td>N/A</td>
<td>0.066</td>
<td>473.03</td>
</tr>
<tr>
<td>58</td>
<td>N/A</td>
<td>0.121</td>
<td>3,654.24</td>
</tr>
<tr>
<td>59</td>
<td>N/A</td>
<td>0.129</td>
<td>5,502.14</td>
</tr>
<tr>
<td>60</td>
<td>N/A</td>
<td>0.096</td>
<td>7,359.58</td>
</tr>
<tr>
<td>61</td>
<td>N/A</td>
<td>0.081</td>
<td>57,550.00</td>
</tr>
<tr>
<td>62</td>
<td>N/A</td>
<td>0.093</td>
<td>86,604.00</td>
</tr>
<tr>
<td>63</td>
<td>N/A</td>
<td>0.108</td>
<td>115,453.00</td>
</tr>
</tbody>
</table>

The results of Table 2–9 show that the quality of the solutions generated and the quality of lower bounds is very good. In particular the running times of primal algorithm are very small.

2.9 Conclusions

In this chapter we discuss the multi-facility lot-sizing problem. We propose the following heuristic approaches to solve the problem: dynamic programming algorithm, a cutting plane algorithm and a primal-dual algorithm. For this problem we also give a different formulation that we refer to as the extended problem formulation. The linear programming relaxation of extended formulation gives lower bounds that are at least as high as the lower bounds from linear programming relaxation of “original” problem formulation. We present a set of valid inequalities for the multi-facility lot-sizing problem and show that they are facet defining inequalities.
We tested the performance of the heuristics on a wide range of randomly generated problems. The dynamic programming algorithm gave good quality solutions for all problem instances. The error (calculated with respect to the optimal solution or with respect to a lower bound in case that the optimal solution does not exists) reported is less than 1.6%. The running time of the dynamic programming algorithm is $O(FT^4)$. This explains the relatively high running times of this algorithm for the seventh group of problems. Linear programming relaxation of extended formulation gave high quality lower bounds. The maximum error reported for problem classes 1 to 45 is 0.39% from optimal.

It has been shown that the multi-facility lot-sizing problem is $NP$-hard when holding costs are not restricted in sign (Wu and Golbasi [106]). This explains the fact that all algorithms gave their worst results for problem classes 32 and 36 to 45. In particular primal-dual algorithm performed poorly for these type of problems.

For problems 57 to 63 CPLEX ran out of memory and failed to provide an integer solution. Primal-dual algorithm however gave solutions within 0.377% error gap in less than 14 CPU seconds.
CHAPTER 3
EXTENSIONS OF THE MULTI-FACILITY LOT-SIZING PROBLEM

3.1 Multi-Commodity, Multi-Facility Lot-Sizing Problem

In the previous chapter we discussed the single-commodity, multi-facility lot-sizing problem. In practice, management of production, inventory and transportation in a plant typically involves coordinating decisions for a number of commodities. In this section we analyze and propose solution approaches for the multi-commodity, multi-facility lot-sizing problem.

The multi-commodity, multi-facility lot-sizing problem consists of finding a production and transportation schedule for a number of commodities over a finite time horizon to satisfy known demand requirements without allowing backlogging. The schedule is such that the total production, inventory holding, transportation and set-up costs are minimized. These costs may vary by product, facility and time period. Production capacities and joint order costs tie together different commodities and necessitate careful coordination of their production schedules. It is easy to see that without the presence of joint order costs or production capacities, this problem can be handled by solving each commodity subproblem separately. In this section we consider the multi-commodity problem with only production capacities (no transportation capacities and no joint order costs) and refer to it as the capacitated multi-commodity, multi-facility lot-sizing problem.

A large amount of work has been devoted to the capacitated multi-commodity, single-facility lot-sizing problem since it is the core problem in the Aggregated Production Planning models. Solutions to lot-sizing problems are often inputs to Master Production Schedules and subsequently to Materials Requirements Planning.
(MRP) systems in a “push” type manufacturing environment (see Bhatnagar et al. [13] and Nahmias [85] for a review of these models).

It has been shown that, even in the single-commodity case, the capacitated lot-sizing problem is \( NP \)-hard [42]. Bitran and Yanasse [15] showed that the capacitated multi-commodity problem is \( NP \)-hard and Chen and Thizy [24] showed that the problem is strongly \( NP \)-hard. Heuristic approaches to solve the problem were developed by Lambrecht and VanderVeken [71], Dixon and Silver [31] and Maes and van Wassenhove [77, 78]. The method proposed by Barany et al. [10] solves the single-facility, capacitated problem optimally using a cutting plane algorithm followed by a branch and bound procedure. These algorithms do not consider setup times.

Three groups of researchers pioneered work on the capacitated economic lot-sizing problem with setup times. Manne [81] used a linear programming model; Dzielinski and Gomory [33] used Dantzig-Wolfe decomposition; and Lasdon and Terjung [72] used a generalized upper bounding procedure.

Eppen and Martin [36] provided an alternative formulation of the capacitated, multi-commodity lot-sizing problem known as the shortest path formulation. They showed that the linear programming relaxation of the shortest path formulation is very effective in generating lower bounds, and the bounds are equal to those that could be generated using Lagrangean relaxation or column generation.

So far, promising heuristic approaches to solve the capacitated multi-commodity lot-sizing problem seem to be those based on Lagrangean relaxation. Thizy and van Wassenhove [96] and Trigeiro et al. [98] proposed a Lagrangean relaxation of the capacity constraints. They updated the Lagrangean multipliers using a subgradient approach and proposed a heuristic to obtain feasible solutions. Merle et al. [32] used a Lagrangean relaxation approach as well; however, they updated the Lagrangean multipliers using a cutting plane method. Chen and Thizy [24] analyzed and compared the quality of different lower bounds calculated using relaxation methods such as Lagrangean relaxation with respect to different sets of constraints.
and linear programming relaxation. Millar and Yang [82] proposed a Lagrangean decomposition procedure to solve the capacitated multi-commodity lot-sizing problem. Their approach decomposes the problem into a transportation problem and $K$ independent single-commodity lot-sizing problems. Thizy [95] analyzed the quality of solutions from Lagrangean decomposition on original problem formulation and shortest path formulation using polyhedral arguments.

3.1.1 Problem Formulation

In many practical situations, coordination of production, inventory and transportation decisions involves different commodities. This complicates the problem considerably. In this section we discuss the multi-commodity, multi-facility lot-sizing problem. This is a generalization of the classical capacitated, multi-commodity lot-sizing problem. We add to the classical problem a new dimension, the facility selection problem. In addition, we consider transportation costs and their effect on lot-sizing decisions.

Assume that there are $K$ commodities that need to be produced. Each commodity faces a known demand during each of the next $T$ periods. Note that commodities share a common production resource with item specific setup costs. The goal is to decide on the production schedule for each commodity, such that production, transportation and inventory costs in all the facilities are minimized, demand is satisfied and capacity constraints are not violated. For each commodity $k$, we define the following input data:

- $p_{itk}$: production unit cost for commodity $k$ at facility $i$ in period $t$
- $s_{itk}$: production set-up cost for commodity $k$ at facility $i$ in period $t$
- $h_{itk}$: inventory unit cost for commodity $k$ at facility $i$ in period $t$
- $c_{itk}$: transportation unit cost for commodity $k$ at facility $i$ in period $t$
- $b_{tk}$: demand in period $t$ for commodity $k$
- $v_{it}$: production capacity at facility $i$ in period $t$

We introduce the following decision variables:

- $q_{itk}$: production quantity for commodity $k$ at facility $i$ in period $t$
- $x_{itk}$: amount of commodity $k$ transported from facility $i$ in period $t$
\[ I_{itk} \] amount of commodity \( k \) in the inventory at the facility \( i \) in the end of period \( t \).

\[ y_{itk} \] a binary variable that equals 1 if there is a production set-up for commodity \( k \) at the facility \( i \) in period \( t \).

An MILP formulation of the problem is the following:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} (p_{itk}q_{itk} + s_{itk}y_{itk} + c_{itk}x_{itk} + h_{itk}I_{itk}) \\
\text{subject to} & \quad (MC) \\
I_{i,t-1,k} + q_{itk} &= x_{itk} + I_{itk} \quad i = 1, \ldots, F; t = 1, \ldots, T; k = 1, \ldots, K \quad (3.1) \\
\sum_{i=1}^{F} x_{itk} &= b_{tk} \quad t = 1, \ldots, T; k = 1, \ldots, K \quad (3.2) \\
\sum_{k=1}^{K} q_{itk} &\leq v_{it} \quad i = 1, \ldots, F; t = 1, \ldots, T \quad (3.3) \\
q_{itk} &\leq b_{tTk}y_{itk} \quad i = 1, \ldots, F; t = 1, \ldots, T; k = 1, \ldots, K \quad (3.4) \\
n_{itk}, I_{itk}, x_{itk} &\geq 0 \quad i = 1, \ldots, F; t = 1, \ldots, T; k = 1, \ldots, K \quad (3.5) \\
y_{itk} &\in \{0, 1\} \quad i = 1, \ldots, F; t = 1, \ldots, T; k = 1, \ldots, K \quad (3.6)
\end{align*}
\]

Note that \( b_{tTk} \) is the total demand for commodity \( k \) from period \( t \) to \( T \). Constraints (3.1) and (3.2) are the flow conservation constraints at the production and demand points respectively. Constraints (3.3) are the production capacity constraints. These constraints reflect the multi-commodity nature of our problem. If they are absent, the problem can be decomposed into \( K \) single commodity problems.

We assume that initial and ending inventories are zero for all items. There is no loss of generality in this assumption, since we can reset for each commodity the given level of initial (ending) inventory \( I_{i1k} (I_{iTk}) \) at zero by removing \( I_{i1k} \) from the demand of the first periods (adding \( I_{iTk} \) to the demand of the last period), thus obtaining the adjusted demands for \( t = 1, \ldots, T \) and \( k = 1, \ldots, K \).

We propose a Lagrangean decomposition-based heuristic to solve the multi-commodity problem. The decomposition is performed on the extended problem.
formulation. The extended formulation, similar to the single commodity problem, requires splitting the production variables \( q_{itk} \) by destination into variables \( q_{it\tau k} \) \((\tau = t, \ldots, T)\), where \( \tau \) denotes the period for which production takes place. Given this, we have

\[
q_{itk} = \sum_{\tau=t}^{T} q_{it\tau k}
\]

\[
x_{itk} = \sum_{s=1}^{t} q_{istk}
\]

\[
I_{itk} = \sum_{s=1}^{t} \sum_{\tau=t}^{T} q_{ist\tau k} - \sum_{s=1}^{t} q_{istk}
\]

\( q_{it\tau k} \) denotes the production quantity for commodity \( k \) from facility \( i \), in time period \( t \) for period \( \tau \). The extended formulation of (MC) is the following:

\[
\text{minimize } \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} (\sum_{\tau=t}^{T} \bar{c}_{it\tau k} q_{it\tau k} + s_{itk} y_{itk})
\]

subject to

\[
\sum_{i=1}^{F} \sum_{t=1}^{\tau} q_{it\tau k} = d_{\tau k} \quad \tau = 1, \ldots, T; k = 1, \ldots, K \tag{3.7}
\]

\[
\sum_{k=1}^{K} \sum_{\tau=t}^{T} q_{it\tau k} \leq v_{it} \quad i = 1, \ldots, F; t = 1, \ldots, T \tag{3.8}
\]

\[
q_{it\tau k} - b_{\tau k} y_{itk} \leq 0 \quad i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T; k = 1, \ldots, K \tag{3.9}
\]

\[
q_{it\tau k} \geq 0 \quad i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T; k = 1, \ldots, K \tag{3.10}
\]

\[
y_{itk} \in \{0, 1\} \quad i = 1, \ldots, F; t = 1, \ldots, T; k = 1, \ldots, K \tag{3.11}
\]

\( \bar{c}_{it\tau k} \) consists of the unit production cost of commodity \( k \) at facility \( i \) in period \( t \), the unit transportation cost from facility \( i \) in period \( \tau \), as well as the total unit cost of holding the item from period \( t \) to \( \tau \) at facility \( i \) \((\bar{c}_{it\tau k} = p_{itk} + c_{it\tau k} + \sum_{s=t+1}^{\tau} h_{isk})\).
3.1.2 Linear Programming Relaxation

The linear programming relaxation (LP) of (MC) is obtained by replacing constraints (3.6) with

\[ y_{itk} \geq 0 \text{ for } i = 1, \ldots, F; t = 1, \ldots, T; k = 1, \ldots, K. \]

The solution to this relaxation is generally far from optimal. We want to give a measure of the quality of the linear relaxation, and specifically display a worst case for the error bound. Note that \( v(LP) \) denotes the optimal objective function value of (LP) and \( v(MC) \) denotes the optimal objective function value of (MC).

**Theorem 3.1.1**

\[
v(LP) \geq \sum_{k=1}^{K} \min_{i=1,\ldots,F; t=1,\ldots,T} s_{itk} \quad (3.12)
\]

\[
v(MC) - v(LP) \leq \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} s_{itk} - \sum_{k=1}^{K} \min_{i=1,\ldots,F; t=1,\ldots,T} s_{itk} \quad (3.13)
\]

We present a class of problems for which these bounds are attained asymptotically with respect to \( \epsilon \).

**Proof:** The fixed charge production cost \( s_{itk} \) is assumed to be positive, thus there exists an optimal value of problem (LP) such that \( y_{itk} = q_{itk}/b_{tTk} \). Since,

\[
\sum_{i=1}^{F} \sum_{t=1}^{T} q_{itk} \geq \frac{1}{b_{1Tk}} \sum_{i=1}^{F} \sum_{t=1}^{T} q_{itk} = 1
\]

the total set-up cost of this solution will be greater than

\[
\sum_{k=1}^{K} \min_{i=1,\ldots,F; t=1,\ldots,T} s_{itk}.
\]

This proves (3.12).

In order to prove (3.13), let \((q, I, x, y)\) be an optimal solution to (LP) and \((q, I, x, [y])\) a feasible solution to (MC) obtained by fixing the fractional components of \( y \) to 1. Then, we have the following relationship:
\[ v(MC) - v(LP) \leq \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} (s_{itk} [y] - s_{itk} \frac{q_{itk}}{b_{itTk}}) \]

\[ \leq \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} s_{itk} - \sum_{k=1}^{K} \min_{i=1,\ldots,F; t=1,\ldots,T} s_{itk} \sum_{i=1}^{F} \sum_{t=1}^{T} \frac{q_{itk}}{b_{itTk}} \]

\[ \leq \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} s_{itk} - \sum_{k=1}^{K} \min_{i=1,\ldots,F; t=1,\ldots,T} s_{itk} \quad \text{(due to (3.14))} \]

Now we present a class of problems for which the bounds derived in (3.12) and (3.13) are obtained asymptotically with respect to \( \epsilon \). The class of problems we consider has the following properties:

**Demands:**

\[ b_{tk} = \epsilon, b_{Tk} = 1/\epsilon, \text{ for } t = 1, \ldots, T-1 \text{ and } k = 1, \ldots, K. \]

**Costs:**

\[ s_{1tk} = 1, p_{1tk} = 0, h_{1tk} > 1, c_{1tk} = 0 \text{ for } t = 1, \ldots, T; k = 1, \ldots, K. \]

\[ s_{itk} > 1, p_{itk} > 0, h_{itk} > 0, x_{itk} > 0 \text{ for } i = 2, \ldots, F; t = 1, \ldots, T; k = 1, \ldots, K. \]

**Capacities:**

\[ v_{1t} = K \epsilon, \text{ for } t = 1, \ldots, T-1 \text{ and } v_{1T} = K/\epsilon. \]

The optimal solution for this class of problems is such that

\[ q_{itk} = b_{tk}, y_{itk} = 1, x_{1tk} = b_{tk} \text{ for } t = 1, \ldots, T; k = 1, \ldots, K, \]

and an optimal solution to the corresponding linear programming relaxation is

\[ q_{itk} = b_{tk}, y_{itk} = \frac{b_{tk}}{b_{itTk}} \text{ for } t = 1, \ldots, T; k = 1, \ldots, K. \]
Thus,
\[
y_{1tk} = \begin{cases} 
1 & \text{if } t = T; \ k = 1, \ldots, K \\
\frac{\epsilon}{1/\epsilon + (T-t)\epsilon} & \text{for } t = 1, \ldots, T-1; \ k = 1, \ldots, K.
\end{cases}
\]

As $\epsilon$ approaches to zero, we have the following equation:
\[
y_{1tk} = \begin{cases} 
1 & \text{if } t = T; \ k = 1, \ldots, K \\
0 & \text{for } t = 1, \ldots, T-1; \ k = 1, \ldots, K.
\end{cases}
\]

Therefore,
\[
\lim_{\epsilon \to 0} v(LP) = \sum_{k=1}^{K} s_{1Tk} = \sum_{k=1}^{K} \min_{i=1,\ldots,F; t=1,\ldots,K} s_{itk}
\]
and
\[
v(MC) = \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} s_{itk}.
\]

We have showed that, for a class of multi-retailer, multi-facility lot-sizing problems the right hand sides of (3.12) and (3.13) are attained asymptotically with respect to $\epsilon$. \qed

3.1.3 Valid Inequalities

Consider the formulation (MC) of the multi-commodity problem. Let $\Phi_k$ denote the set of feasible solutions to the $k$-th single-commodity problem and let $LP_k$ denote the set of the feasible solutions to the linear programming relaxation of the $k$-th single-commodity problem. We can re-state the capacitated multi-commodity problem as follows:

\[
\text{minimize } \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} (p_{itk}q_{itk} + s_{itk}y_{itk} + c_{itk}x_{itk} + h_{itk}I_{itk})
\]
subject to
\[
(q_k, x_k, I_k, y_k) \in \Phi_k, \ k = 1, \ldots, K
\]
\[
\sum_{k=1}^{K} q_{itk} \leq v_{it} \quad t = 1, \ldots, T; i = 1, \ldots, F
\]

Or equivalently,
\[
\text{minimize } \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} (p_{itk}q_{itk} + s_{itk}y_{itk} + c_{itk}x_{itk} + h_{itk}I_{itk})
\]
subject to (MC*)

\[(q_k, x_k, I_k, y_k) \in LP_k \quad k = 1, \ldots, K \quad (3.15)\]

\[\sum_{k=1}^{K} q_{ikt} \leq v_{it} \quad i = 1, \ldots, F; t = 1, \ldots, T \quad (3.16)\]

\[y_{ikt} \in \{0, 1\} \quad i = 1, \ldots, F; t = 1, \ldots, T; k = 1, \ldots, K \quad (3.17)\]

where \(LP_k \cap (y \in \{0, 1\}) = \Phi_k\).

Given a fractional point \((q_1, x_1, I_1, y_1), \ldots, (q_K, x_K, I_K, y_K)\), the goal is to find a valid inequality that cuts off this non integer point from the feasible region of linear programming relaxation of (MC*). Ignoring constraints (3.16), the problem decomposes into \(K\) single-commodity, multi-facility lot-sizing problems. For the single-commodity problem, we have proposed in Section 2.6.1 a set of valid inequalities. Let \(\Phi\) be the feasible region of the multi-commodity flow problem, then

\[co(\Phi) \subseteq \cap_{k=1}^{K} co(\Phi_k)\]

This implies that the valid inequalities for the single commodity problem, are valid for the multi-commodity problem as well. Thus, one can use these inequalities to check for each commodity \(k\) if point \((q_k, x_k, I_k, y_k)\) can be cut off from \(\Phi\).

3.1.4 Lagrangean Decomposition Heuristic

In this section we discuss a Lagrangean decomposition-based heuristic that we used to solve the capacitated multi-commodity, multi-facility lot-sizing problem.

Lagrangean relaxation is a classical method for solving integer programming problems (Geoffrion [49], Wolsey [105]). This method has been used to solve various network flow problems. Held and Karp [56, 57] successfully used Lagrangean relaxation to solve the traveling salesman problem; Fisher [40] used this method to solve a machine scheduling problem; Ross and Soland [92] applied this method to the general assignment problem. Holmberg and Yuan [62], Holmberg and Hellstrand [61], and Balakrishnan et al. [8] used Lagrangean relaxation based approaches for network design problems.
We start our discussion with a review of the Lagrangean relaxation approach and its extension, Lagrangean decomposition. We continue then with a detailed description of the Lagrangean decomposition based heuristic we have used to generate upper and lower bounds for the multi-commodity, multi-facility lot-sizing problem.

Review of the method. Geoffrion [49] formally defines a relaxation of an optimization problem \((P)\) as follows:

\[
\min \{ f(x) | x \in X \},
\]

as a problem \((RP)\) over the same decision variable \(x\):

\[
\min \{ g(x) | x \in Y \}
\]

such that \((i)\) the feasible set of \((RP)\) contains that of \((P)\), that is \(X \subseteq Y\), and \((ii)\) over the feasible set of \((P)\), the objective function of \((RP)\) dominates (is at least as good as) that of \((P)\), that is

\[
g(x) \leq f(x), \quad \forall x \in X.
\]

The importance of a relaxation is the fact that it provides bounds on the optimal value of difficult problems. The solution of the relaxation, although usually infeasible for the original problem, can often be used as a starting point for specialized heuristics.

Lagrangean relaxation has shown to be a powerful family of tools for solving integer programming problems approximately. Assume that problem \((P)\) is of the form

\[
\min_{x} \{ f(x) | Ax \geq b, Cx \geq d, x \in X \},
\]

where \(X\) contains only the integrality constraints. The reason for distinguishing between the two types of constraints is that the first of these \((Ax \geq b)\) is typically
complicated, in the sense that problem \((P)\) without this set of constraints would be much easier to solve.

Let \(\lambda\) be a nonnegative vector of weights called Lagrange multipliers. The Lagrangean relaxation \(LR(x, \lambda)\) of \((P)\) is the problem

\[
\min_x \{ f(x) + \lambda(b - Ax)|Cx \geq d, x \in X \},
\]

in which the slack values of the complicating constraints \(Ax \geq b\) have been added to the objective function with weights \(\lambda\), and the constraints \(Ax \geq b\) have been dropped. \(LR(x, \lambda)\) is a relaxation of \((P)\), since \((i)\) the feasible region of \(LR(x, \lambda)\) contains the feasible region of \((P)\), and \((ii)\) for any \(x\) feasible for \((P)\), \(f(x) + \lambda(b - Ax)\) is less than or equal to \(f(x)\). Thus for all \(\lambda \geq 0\), the optimal objective function value of \(LR(x, \lambda)\), which depends on \(\lambda\), is a lower bound on the optimal value of \((P)\). The problem

\[
\max_{\lambda \geq 0} v(LR(x, \lambda))
\]

of finding the highest lower bound is called the Lagrangean dual \((D)\) of \((P)\) with respect to the complicating constraints \(Ax \geq b\).

Let \(v(*)\) denote the optimal objective function value of problem \((*)\). The following theorem by Geoffrion is an important result:

**Theorem 3.1.2** The Lagrangean dual \((D)\) is equivalent to the following primal relaxation \((PR)\):

\[
\min\{f(x)|Ax \geq b, x \in Co\{x \in X|Cx \geq d\}\},
\]

in the sense that \(v(D) = v(PR)\).

This result is based on linear programming duality and properties of optimal solutions of linear programs. Let denote by \((LP)\) the linear programming relaxation of problem \((P)\). Then the following holds:

(i) If \(Co\{x|Cx \geq d, x \in X\} = \{x|Cx \geq d\}\), then \(v(P) \geq v(PR) = v(D) = v(LP)\).
In this case the Lagrangean relaxation has the integrality property, and the \((D)\) bound is equal to the \((LP)\) bound.

(ii) If \(Co\{x|Cx \geq d, x \in X\} \subset \{x|Cx \geq d\}\), then \(v(P) \geq v(PR) = v(D) \geq v(LP)\), and it is possible that the Lagrangean bound is strictly better than the \((LP)\) bound.

This suggests that in deciding on how to implement the Lagrangean relaxation method, one should consider the following properties of the set \(\Xi = \{x|Cx \geq d\}\):

(i) \(\Xi\) should be simple enough that the resulting optimization subproblems are not computationally intractable (usually \(\Xi\) decomposes into simpler subsets, \(\Xi = \Pi_{j \in J} \Xi_j\)),

(ii) \(\Xi\) should be complex enough, such that the subsets \(\Xi_j\) do not have the integrality property. Otherwise, the lower bounds generated would be equal to the lower bounds from the corresponding linear programming relaxation.

The Lagrangean function \(z(\lambda) = v(LR(x, \lambda))\) is an implicit function of \(\lambda\), and \(z(\lambda)\), the lower envelope of a family of linear functions of \(\lambda\), is a concave function of \(\lambda\), with break-points where it is not differentiable.

An extension to Lagrangean relaxation is the Lagrangean decomposition method introduced by Guignard and Kim [52]. Different than Lagrangean relaxation, Lagrangean decomposition does not remove the complicating constraints, but decomposes the problem into two subproblems that collectively share the constraints of the original problem. This is achieved by introducing a new set of variables \(z\), such that \(x = z\). Then, problem \((P)\) reads

\[
\min_{x,z}\{f(x)| Ax \geq b, Cz \geq d, x \in X, z = x, z \in X\}.
\]

Relaxing the “copy” constraints \(z = x\) yields a decomposable problem, justifying the name “Lagrangean decomposition” \((LD(x, z, \lambda))\) (Lagrangean relaxation refers to the case when only one subset of constraints is relaxed), which is given by

\[
\min_{x,z}\{f(x) + \lambda(z - x)| Ax \geq b, Cz \geq d, x \in X, z \in X\}
\]
\[= \min_x \{ f(x) - \lambda x | Ax \geq b, x \in X \} + \min_z \{ \lambda z | Cz \geq d, z \in X \}.\]

Guignard and Kim [52] show that Lagrangean decomposition can in some cases yield bounds substantially better than “traditional” Lagrangean relaxation. The Lagrangean dual problem (D) is the following:

\[\max_\lambda \{ \min_{x,z} LD(x, z, \lambda) \} .\]

The following theorem by Guignard and Kim is an important result:

**Theorem 3.1.3**

\[v(D) = \min \{ f(x) | x \in Co \{ x | Ax \geq b, x \in X \} \cap Co \{ x | Cx \geq d, x \in X \} \}.\]

If one of the subproblems has the integrality property, then \(v(D)\) is equal to the better Lagrangean relaxation bound. If both have the integrality property, then \(v(D) = v(LP)\). Note that finding a lower bound for problem (P) using a Lagrangean relaxation/decomposition algorithm requires optimally solving the inner minimization problem and the outer maximization problem. Since the Lagrangean function \(z(\lambda) = v(LD(x, z, \lambda))\) is a piecewise concave function, we use a subgradient optimization algorithm to maximize it. In implementing the subgradient algorithm, an important issue is choosing a step size to move along the subgradient direction that guarantees convergence to the \(\lambda\) that maximizes the dual function (D).

Significant for the decomposition is that the inner minimization problem is easier to solve than the original problem (P). Recall that Lagrangean relaxation/decomposition is an iterative method, the inner minimization problem needing to possibly be solved several times. In the case that the inner minimization problem is still difficult and requires a considerable amount of computational efforts to solve to optimality, a practice that can be followed is finding good lower bounds instead. Obviously this will affect the quality of the lower bound from the Lagrangean
relaxation/decomposition. Letting $v(P)$ be the optimal solution to problem (P), then $v(P) \geq v(D)$.

Let $\omega(*)$ denote a lower bound and $\phi(*)$ an upper bound for problem $(*)$. If, for every $\lambda$ we do not solve the Lagrangean decomposition problem $LD(x, z, \lambda)$ optimally, but we rather provide a lower bound, the following holds:

$$v(P) \geq v(D) = \max_{\lambda} v(LD(x, z, \lambda)) \geq \max_{\lambda} \omega(LD(x, z, \lambda)).$$

$max_{\lambda} \omega(LD(x, z, \lambda))$ will still be a lower bound for problem (P), however not as good bound as $v(D)$. In case that we use a heuristic procedure to find a feasible solution to the inner optimization problem, then $v(D) \leq \max_{\lambda} \phi(LD(x, z, \lambda))$.

However, we are not sure anymore if $\max_{\lambda} \phi(LD(x, z, \lambda))$ gives a lower bound for problem (P) since one of the following may happen: $v(P) \geq \max_{\lambda} \phi(LD(x, z, \lambda))$ or $v(P) \leq \max_{\lambda} \phi(LD(x, z, \lambda))$.

In the case when a lower bounding procedure is used instead of solving the inner minimization problem, it is important to identify the quality of these bounds compared to the lower bounds from the linear programming relaxation. If the lower bound guarantees that

$$\max_{\lambda} \omega(LD(x, z, \lambda)) \geq v(LP)$$

and the running time of this procedure outperforms linear programming relaxation, one is better off using the Lagrangean relaxation/decomposition method to find lower bounds to problem (P).

In the next section we describe the Lagrangean decomposition algorithm we used to solve the multi-commodity, multi-facility lot-sizing problem.

3.1.5 Outline of the Algorithm

Consider the extended problem formulation (Ex-MC). The basic idea of our decomposition is to separate the capacitated, multi-commodity problem into subproblems that are computationally easier to solve than the original problem.
There are many ways one can do that; however we aim to decompose the problem in such a way that it has interesting managerial implications as well.

We decompose the problem into two subproblems. The first subproblem consists of the flow conservation constraints and the integrality constraints. This subproblem can be further decomposed by commodity. The single commodity sub-subproblems have the special structure of the single-commodity, multi-facility lot-sizing problem analyzed in Chapter 2. The second subproblem consists of the flow conservation constraints and the capacity constraints. In this decomposition, the first subproblem consists of a collection of MILPs and the second subproblem is a linear program.

Below we give an equivalent formulation of the capacitated multi-commodity, multi-facility lot-sizing problem. We introduce the continuous variables $z_{it\tau k}$ that are simply “copies” of the production variables $q_{it\tau k}$. This allows for the duplication of some of the constraints.

\[
\minimize \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} \left[ \sum_{\tau=t}^{T} \bar{c}_{it\tau k} q_{it\tau k} + s_{itk} y_{itk} \right]
\]

subject to

\begin{align*}
q_{it\tau k} &= z_{it\tau k} & i &= 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T; k = 1, \ldots, K \quad (3.18) \\
\sum_{i=1}^{F} \sum_{t=1}^{T} z_{it\tau k} &= b_{\tau k} & \tau &= 1, \ldots, T; k = 1, \ldots, K \quad (3.19) \\
\sum_{k=1}^{K} \sum_{\tau=t}^{T} z_{it\tau k} &\leq v_{it} & i &= 1, \ldots, F; t = 1, \ldots, T \quad (3.20) \\
z_{it\tau k} &\geq 0 & i &= 1, \ldots, F; t = 1, \ldots, T; k = 1, \ldots, K \quad (3.21)
\end{align*}

It is clear that the above formulation is equivalent to formulation (Ex-MC). Relaxing the “copy” constraints (3.18) and moving them to the objective function yields the following Lagrangian decomposition problem:

\[
\minimize \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} \left[ \sum_{\tau=t}^{T} (\bar{c}_{it\tau k} + \lambda_{it\tau k}) q_{it\tau k} + s_{itk} y_{itk} - \sum_{\tau=t}^{T} \lambda_{it\tau k} z_{it\tau k} \right]
\]
subject to \((LD(x,z,\lambda))\)

\[(3.7), (3.9), (3.10), (3.11), (3.19), (3.20), \text{ and } (3.21).\]

The Lagrangean decomposition \((LD(x,z,\lambda))\) problem can now be separated into the following two subproblems:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} \left[ \sum_{\tau=t}^{T} (v_{it\tau k} + \lambda_{it\tau k}) q_{it\tau k} + s_{itk} y_{itk} \right] \\
\text{subject to} & \quad (3.7), (3.9), (3.10), (3.11), \text{ and } (SP_1)
\end{align*}
\]

and

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} \sum_{\tau=t}^{T} -\lambda_{it\tau k} z_{it\tau k} \\
\text{subject to} & \quad (3.19), (3.20), \text{ and } (SP_2)
\end{align*}
\]

The corresponding Lagrangean dual \((LD)\) problem is the following:

\[
\max_{\lambda} v(LD(x,z,\lambda))
\]

Note that under the general framework of Lagrangean decomposition method, constraints \((3.7)\) do not need to be duplicated for both subproblems. However, computational experience indicates that as long as the added constraints do not add computational burden to the subproblems, constraint duplication improves the speed of the convergence and yields better lower bounds (Guignard and Kim [52]).

The potential computational savings from using the dual algorithm is significant since the subgradient search requires solving \(K\) single commodity subproblems in each iteration. If a problem instance considers 30 commodities for example, this results in up to 15,000 calls to the subproblem (the maximum number of iterations we use is 500). In Section 3.1.7 we compare the performance of the Lagrangean decomposition approach when the subproblems are solved to optimality versus the case when the dual algorithm is used.
Subproblem (SP$_1$) can be decomposed by commodity. Each sub-subproblem $k$ has the following MILP formulation:

$$\text{minimize} \ \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{\tau=t}^{T} (c_{it\tau} + \lambda_{it\tau})q_{it\tau} + s_{it}y_{it}$$

subject to

$$\sum_{i=1}^{F} \sum_{t=1}^{T} q_{it\tau} = d_{\tau}, \quad \tau = 1, \ldots, T$$

$$q_{it\tau} - b_{\tau} y_{it} \leq 0 \quad i = 1, \ldots, F; t = 1, \ldots, T; \tau \geq t$$

$$q_{it\tau} \geq 0 \quad i = 1, \ldots, F; t = 1, \ldots, T; \tau \geq t$$

$$y_{it} \in \{0, 1\} \quad i = 1, \ldots, F; t = 1, \ldots, T$$

One can see that this formulation is the same as the formulation of multi-facility lot-sizing problem we discussed in the previous chapter. Since for most of the test problems discussed in Chapter 2, the dual algorithm gave close to optimal lower bounds and its running time was much smaller than the running time of the linear programming relaxation, we use the dual algorithm to find good lower bounds for the subproblems (SP$_{1k}$) for all $k = 1, \ldots, K$. However, it should be recognized that the quality of the lower bounds from the decomposition may not always be as good as in the case that we solve the subproblems optimally.

On the other hand, subproblem (SP$_2$) is simply a linear program. We use CPLEX callable libraries to solve this problem. The solution of (SP$_2$) is feasible for the (Ex-MC) problem. However, since the set-up costs are not considered in the formulation, we use a simple procedure to calculate an upper bound (Figure 3–1).

**Proposition 3.1.1** The Lagrangean decomposition of (Ex-MC) gives lower bounds that are at least as high as the corresponding linear programming relaxation of extended formulation.

**Proof:** Let $\Phi$ denote the set all feasible solutions to the extended formulation of the multi-commodity, multi-retailer problem (Ex-MC). Let $\Phi_1$ be the set of the
Let $z^{**}$ be the optimal solution to subproblem (SP$_2$) in iteration $s$.

Initialize $UB^s = 0$; $\text{sum}_{itk} = 0$ for $i = 1, \ldots, F$; $t = 1, \ldots, T$; $k = 1, \ldots, K$.

for $i = 1, \ldots, F$; $t = 1, \ldots, T$; $k = 1, \ldots, K$

for $\tau = t, \ldots, T$
do

if $z^{**}_{itrk} \geq 0$ then $UB^s = UB^s + \bar{c}_{itrk} z^{**}_{itrk}$

$\text{sum}_{itk} = \text{sum}_{itk} + z^{**}_{itrk}$

if $\text{sum}_{itk} > 0$ then $UB^s = UB^s + s_{itk}$.

Figure 3–1: Upper bound procedure

feasible solutions corresponding to linear programming relaxation of subproblem (SP$_1$) and $\Phi_2$ be the set of feasible solutions to linear programming relaxation of subproblem (SP$_2$).

Guignard and Kim [52] show that “Optimizing the Lagrangean decomposition dual is equivalent to optimizing the primal objective function on the intersection of the convex hulls of the constraint sets” (Theorem 3.1.3). Therefore, optimizing (LD) is the same as optimizing

$$
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} \left[ \sum_{\tau=t}^{T} \bar{c}_{itrk} q_{itrk} + s_{itk} y_{itk} \right] \\
\text{subject to} & \quad q, y \in Co\{q, y|q, y \in \Phi_1 \cap y \in \{0, 1\} \} \cap q \in Co\{q|q \in \Phi_2\}
\end{align*}
$$

The linear programming relaxation of (Ex-MC) minimizes the same objective function over the intersection of $\Phi_1$ and $\Phi_2$.

$$
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} \left[ \sum_{\tau=t}^{T} \bar{c}_{itrk} q_{itrk} + s_{itk} y_{itk} \right] \\
\text{subject to} & \quad q, y \in \Phi_1 \cap q \in \Phi_2
\end{align*}
$$

(LP-Ex)
Since $q$ are continuous variables

$$\text{Co}\{q|q \in \Phi_2\} = \Phi_2,$$

however,

$$\text{Co}\{q,y|q, y \in \Phi_1 \cap y \in \{0,1\}\} \subseteq \Phi_1.$$

This shows that the feasible region of the dual problem (LD) is smaller than the feasible region of the linear programming relaxation (LP-Ex). Therefore, the objective function value we get solving (LD) will be at least as high as the objective function value of the linear programming relaxation. □

**Proposition 3.1.2** The Lagrangean decomposition of (Ex-MC) gives lower bounds that are equal to the bounds from Lagrangean relaxation of (Ex-MC) with respect to the capacity constraints (3.8).

**Proof:** Let $\Phi_1$ be the set of solutions satisfying constraints (3.7),(3.9),(3.10) and (3.11) and $\Phi_2$ be the set of solutions satisfying constraints (3.8) and (3.11). Let $\Phi^*$ be the intersection of $\Phi_2$ with the convex hull of $\Phi_1$. Solving the Lagrangean relaxation with respect to the capacity constraints (3.8) is equivalent to minimizing the objective function of (Ex-MC) over $\Phi^*$ (Theorem 3.1.2). On the other hand, solving the Lagrangean decomposition of (Ex-MC) is equivalent to minimizing the objective function over the intersection of $\Phi_2^* = \Phi_2 \cap (3.7)$ with the convex hull of $\Phi_1$ (Theorem 3.1.3), or in other words minimizing the objective function of (Ex-MC) over the intersection of $\Phi^*$ with (3.7). Note that all the solutions in $\Phi^*$ satisfy constraints (3.7) since these constraints are contained in $\Phi_1$. Therefore, solving the Lagrangean decomposition problem is equivalent to minimizing the objective function of (Ex-MC) over $\Phi^*$. This concludes the proof that both the Lagrangean relaxation with respect to the capacity constraints and the Lagrangean decomposition scheme we propose give the same lower bounds for (Ex-MC) □
The lower bounds generated from the Lagrangean decomposition heuristic are the same as the bounds from Lagrangean relaxation with respect to the capacity constraints, however we chose to implement the Lagrangean decomposition approach. There are two reasons we choose to do so (i) The decomposition scheme provides feasible solutions to problem (Ex-MC) at every iteration (ii) The decomposition converges faster.

Subgradient optimization algorithm. It is well-known (Nemhauser and Wolsey [86]) that the Lagrangean dual function is concave and nondifferentiable. To maximize it and consequently derive the best Lagrangean lower bound we use a subgradient optimization method. For more details on the subgradient optimization method see Held, Wolfe and Crowder [58] and Crowder [28], and for a survey of nondifferentiable optimization techniques see Lemaréchal [73].

Subgradient optimization is an iterative method in which steps are taken along the negative of the subgradient of the Lagrangean function \( z(\lambda) \) \( (z(\lambda) = v(LD(x, z, \lambda)) \). At each iteration \( s \), we calculate the Lagrangean multipliers \( \lambda_{itrk} \) using the following equation:

\[
\lambda_{itrk}^{s+1} = \lambda_{itrk}^s + u^s(q_{itrk} - z_{itrk}),
\]

where

\[
u^s = \frac{\gamma^s (\min_{UB} - \max_{LB})}{\sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{\tau=t}^{T} \sum_{k=1}^{K} (q_{itrk} - z_{itrk})^2}.
\]

\( \gamma^s \) is a number greater than 0 and less or equal to 2. \( \gamma^s \) is reduced if the lower bound fails to improve after a fixed number of iterations; \( \max_{LB} \) is the best lower bound found up to iteration \( s \) and \( \min_{UB} \) is the best upper bound. Calculating the step size \( \lambda \) using equation (3.22) is a common heuristic rule that has been used in the literature. The step size updating using rule (3.22) generally allows convergence to the \( \lambda^* \) that maximizes the Lagrangean function \( z(\lambda) \) (Nemhauser and Wolsey [86]).

In order to find a subgradient direction at each step of the Lagrangean decomposition procedure, we need to find a feasible solution to subproblem (SP1).
Step 1: Initialize $\lambda$, $\min_{UB}$, $\max_{LB}$, $s$, $u$, $\gamma$, $\epsilon$, $\text{count}$.

Step 2: Solve the subproblems (SP$_1$) and (SP$_2$). Compute the lower bound:

$$LB^s = \sum_{k=1}^{K} \omega (\text{SP}_{1k}(\lambda^s)) + v(\text{SP}_2(\lambda^s))$$

for the current iteration $s$

$\text{count} = \text{count} + 1$

If $LB^s > \max_{LB}$, then $\max_{LB} = LB^s$

Step 3: Compute an upper bound $UB^s$

If $UB^s < \min_{UB}$, then $\min_{UB} = UB^s$

If $\min_{UB} = \max_{LB}$, then STOP

If $\gamma \leq \epsilon$, then STOP

Step 4: Update the multipliers using equation (3.22)

Step 5: Stop if the number of iterations reach the prespecified limit ($\text{count}$)

Otherwise go to Step 2

Figure 3–2: Lagrangean decomposition algorithm

The dual algorithm provides only a lower bound to (Ex-MC), but does not provide a feasible primal solution. Therefore, we use the primal algorithm (Section 2.5) to find a feasible solution to subproblems (SP$_{1k}$).

In our computational experiments, we terminate the algorithm if one of the following happens: $(i)$ the best lower bound is equal to the best upper bound (the optimal solution is found), $(ii)$ the number of iterations reaches a prespecified bound, $(iii)$ the scalar $\gamma^s$ is less than or equal to $\epsilon$ (a small number close to zero). Figure 3–2 presents the steps of the Lagrangean decomposition algorithm.

3.1.6 Managerial Interpretation of the Decomposition

In this section we discuss the managerial insights of the decomposition procedure proposed in Section 3.1.5. The choice of the Lagrangean decomposition scheme we present is motivated not only by its computation capability, but also by its interesting managerial implications. Several studies (Burton and Obel [19] and Jörnsten and Leisten [64]) have recognized that mathematical decomposition often leads to insights for general modelling strategies and even new decision structures. In this discussion
we refer to subproblem (SP\(_1\)) as the product (commodity) subproblem and (SP\(_2\)) as the resource subproblem.

The Lagrangean decomposition we propose helps understanding and solving managerial issues that arise in multi-facility manufacturing planning. Suppose we consider the resource subproblem as a decision problem for a production manager who supervises multiple facilities, and each product subproblem as a decision problem for a product manager. Therefore, the decomposition can be viewed as a decision system where product managers compete for resource capacity available from manufacturing facilities. The production manager, on the other side, represents the interests of efficiently allocating resources from multiple facilities to the products. Often the solutions proposed by the production manager will not agree with the individual solutions of product managers. A search based on the Lagrangean multipliers basically penalizes their differences, while adjusting the penalty vector iteratively. This process stops when the degree of disagreement (the duality gap) is acceptably low, or when further improvement is unlikely. See Wu and Golbasi [106] for a similar discussion on a related problem.

### 3.1.7 Computational Results

In this section we have tested the performance of the Lagrangean Decomposition algorithm on a large group of randomly generated problems. We use the CPLEX callable libraries to solve the MILP formulation (Ex-MC). The CPLEX runs were stopped whenever a guaranteed error bound of 1% or less was achieved, allowing for a maximum CPU time of 5,000 seconds (or 10,000 seconds depending on problem size). We use CPLEX to solve the linear programming relaxation of (Ex-MC) and subproblem (SP\(_2\)). One of the factors that affects the problem complexity is the tightness of the upper bounds on production arcs. If the arcs are very tight, there exist only a few feasible solutions, and this makes the search for the optimal solution easy. On the other hand, if the arc capacities are so loose we could remove them from
Table 3–1: Problem characteristics

<table>
<thead>
<tr>
<th>Problem</th>
<th>Nodes</th>
<th>Arcs</th>
<th>Problem</th>
<th>Nodes</th>
<th>Arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,680</td>
<td>6,000</td>
<td>14</td>
<td>2,280</td>
<td>8,400</td>
</tr>
<tr>
<td>2</td>
<td>2,240</td>
<td>8,000</td>
<td>15</td>
<td>2,430</td>
<td>9,000</td>
</tr>
<tr>
<td>3</td>
<td>2,800</td>
<td>10,000</td>
<td>16</td>
<td>3,330</td>
<td>19,500</td>
</tr>
<tr>
<td>4</td>
<td>3,360</td>
<td>12,000</td>
<td>17</td>
<td>4,980</td>
<td>40,500</td>
</tr>
<tr>
<td>5</td>
<td>3,920</td>
<td>14,000</td>
<td>18</td>
<td>6,630</td>
<td>69,000</td>
</tr>
<tr>
<td>6</td>
<td>4,480</td>
<td>16,000</td>
<td>19</td>
<td>8,280</td>
<td>105,000</td>
</tr>
<tr>
<td>7, . . . , 10</td>
<td>1,680</td>
<td>6,000</td>
<td>20</td>
<td>9,930</td>
<td>148,500</td>
</tr>
<tr>
<td>11</td>
<td>1,830</td>
<td>6,600</td>
<td>21</td>
<td>11,580</td>
<td>199,500</td>
</tr>
<tr>
<td>12</td>
<td>1,980</td>
<td>7,200</td>
<td>22, . . . , 25</td>
<td>1,680</td>
<td>6,000</td>
</tr>
<tr>
<td>13</td>
<td>2,130</td>
<td>7,800</td>
<td>23</td>
<td>9,930</td>
<td>148,500</td>
</tr>
</tbody>
</table>

the problem formulation, the problem loses its multi-commodity nature, and it can be decomposed into $K$ single-commodity problems.

In order to create challenging test problems, we generated the upper bounds in the following way: A necessary condition for feasibility of the problem is

$$
\sum_{i=1}^{F} \sum_{\tau=1}^{t} v_{i\tau} \geq \sum_{k=1}^{K} \sum_{\tau=1}^{t} b_{\tau k} \quad \forall t = 1, \ldots, T.
$$

(3.23)

Under the assumption that all facilities in any time period have the same production capacity $\bar{v}$, we can replace

$$
\sum_{i=1}^{F} \sum_{\tau=1}^{t} v_{i\tau} = Ft\bar{v},
$$

thus,

$$
\bar{v} \geq \frac{1}{Ft} \sum_{k=1}^{K} \sum_{\tau=1}^{t} b_{\tau k} \quad \forall t = 1, \ldots, T.
$$

In fact $\bar{v}$ should be such that the following is satisfied:

$$
\bar{v} \geq \frac{1}{Ft} \max_t \sum_{k=1}^{K} \sum_{\tau=1}^{t} b_{\tau k}.
$$

Letting $\delta (\geq 1)$ be the capacity tightness coefficient, then

$$
\bar{v} = \frac{\delta}{Ft} \max_t \sum_{k=1}^{K} \sum_{\tau=1}^{t} b_{\tau k}.
$$

Using this procedure we generated challenging (but feasible) test problems with tight capacity constraints. We start our computational experiments by generating a nominal case problem with the following characteristics:

- Production set-up costs $s_{it} \sim U[200, 900]$
- Production variable costs $p_{it} \sim U[5, 15]$
- Holding costs $h_{it} \sim U[5, 15]$
- Demand $b_t \sim U[5, 15]$
- Number of facilities $F = 10$
- Number of periods $T = 5$
- Number of commodities $K = 30$
- Capacity tightness $\delta = 1.3$

We alter the nominal case by changing problem characteristics to generate additional problems. For each problem class we generate 20 instances. First we change the number of commodities from 30 to 40, 50, 60, 70, and 80 generating six problem classes (problem classes 1 to 6; problem class 1 is the nominal case).

In the second group of problems we change the capacity tightness coefficient ($\delta$) to 1.1, 1.2, 1.4, and 1.5 (problem classes 7 to 10). In the third group of problems we change the number of facilities from 10 (the nominal case) to 11, 12, 13, 14 and 15 (problem classes 11 to 15).

We also ran the program for problems with different lengths of the time horizon. In problem classes 16 to 21 we change the number of periods to 10, 15, 20, 25, 30, and 35. Finally, we change the level of the fixed charge cost to $s_{it} \sim U[200, 300], U[600, 900], U[900, 1500]$ and $U[1200, 1500]$ (problem classes 22 to 25).

In implementing the Lagrangean decomposition algorithm, for all problem instances we set $\gamma = 1.8$. $\gamma$ is reduced by 20% if there is no improvement in the last 5 iterations. We set a limit of 500 iterations for the Lagrangean decomposition algorithm. As we have mentioned earlier in this section, we are interested in the effectiveness of the primal-dual algorithm as a heuristic in the context of the subgradient search algorithm. Therefore, we present results from the Lagrangean decomposition algorithm where
Table 3–2: Quality of upper bounds (in %) from Lagrangean decomposition

<table>
<thead>
<tr>
<th>Problem</th>
<th>CPLEX</th>
<th>Scheme1</th>
<th>Scheme2</th>
<th>Scheme3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.06</td>
<td>1.92</td>
<td>1.79</td>
<td>1.82</td>
</tr>
<tr>
<td>2</td>
<td>0.05</td>
<td>1.22</td>
<td>1.14</td>
<td>1.16</td>
</tr>
<tr>
<td>3</td>
<td>0.01</td>
<td>0.83</td>
<td>0.76</td>
<td>0.76</td>
</tr>
<tr>
<td>4</td>
<td>0.01</td>
<td>0.62</td>
<td>0.50</td>
<td>0.52</td>
</tr>
<tr>
<td>5</td>
<td>0.01</td>
<td>0.57</td>
<td>0.40</td>
<td>0.42</td>
</tr>
<tr>
<td>6</td>
<td>0.01</td>
<td>0.53</td>
<td>0.32</td>
<td>0.32</td>
</tr>
<tr>
<td>7</td>
<td>1.44</td>
<td>3.67</td>
<td>3.61</td>
<td>3.61</td>
</tr>
<tr>
<td>8</td>
<td>0.38</td>
<td>2.61</td>
<td>2.60</td>
<td>2.53</td>
</tr>
<tr>
<td>9</td>
<td>0.01</td>
<td>1.36</td>
<td>1.25</td>
<td>1.24</td>
</tr>
<tr>
<td>10</td>
<td>0.01</td>
<td>1.01</td>
<td>0.83</td>
<td>0.83</td>
</tr>
<tr>
<td>11</td>
<td>0.08</td>
<td>2.54</td>
<td>2.49</td>
<td>2.49</td>
</tr>
<tr>
<td>12</td>
<td>0.17</td>
<td>2.86</td>
<td>2.84</td>
<td>2.79</td>
</tr>
<tr>
<td>13</td>
<td>0.21</td>
<td>3.53</td>
<td>3.49</td>
<td>3.50</td>
</tr>
<tr>
<td>14</td>
<td>0.34</td>
<td>4.56</td>
<td>4.52</td>
<td>4.49</td>
</tr>
<tr>
<td>15</td>
<td>0.50</td>
<td>4.86</td>
<td>4.76</td>
<td>4.76</td>
</tr>
<tr>
<td>16</td>
<td>0.24</td>
<td>2.03</td>
<td>1.93</td>
<td>1.95</td>
</tr>
<tr>
<td>17</td>
<td>0.37</td>
<td>2.20</td>
<td>2.12</td>
<td>2.12</td>
</tr>
<tr>
<td>18</td>
<td>0.48</td>
<td>2.28</td>
<td>2.20</td>
<td>2.22</td>
</tr>
<tr>
<td>19</td>
<td>0.53</td>
<td>2.28</td>
<td>2.21</td>
<td>2.20</td>
</tr>
<tr>
<td>20</td>
<td>0.57</td>
<td>2.34</td>
<td>2.28</td>
<td>2.29</td>
</tr>
<tr>
<td>21</td>
<td>0.55</td>
<td>2.41</td>
<td>2.35</td>
<td>2.32</td>
</tr>
<tr>
<td>22</td>
<td>0.02</td>
<td>1.25</td>
<td>1.20</td>
<td>1.20</td>
</tr>
<tr>
<td>23</td>
<td>0.04</td>
<td>3.37</td>
<td>3.13</td>
<td>3.21</td>
</tr>
<tr>
<td>24</td>
<td>0.22</td>
<td>4.96</td>
<td>4.85</td>
<td>4.75</td>
</tr>
<tr>
<td>25</td>
<td>0.77</td>
<td>6.54</td>
<td>6.38</td>
<td>6.40</td>
</tr>
</tbody>
</table>
**Scheme 1:** a lower bound to subproblem \((SP_{1k})\) is found using the dual algorithm

**Scheme 2:** subproblem \((SP_{1k})\) is solved to optimality using the exact MILP formulation

**Scheme 3:** a lower bound to subproblem \((SP_{1k})\) is found solving its linear programming relaxation.

The error bounds from the three schemes as well as the linear programming relaxation of \((Ex-MC)\), are presented in Tables 3–2 and 3–3. The quality of the upper bounds we generated using the Lagrangean decomposition algorithm is measured as follows:

\[
\text{Error (\%)} = \frac{\text{Upper Bound} - \text{CPLEX Lower Bound}}{\text{CPLEX Lower Bound}} \times 100,
\]

and the quality of the lower bounds is calculated as follows:

\[
\text{Error (\%)} = \frac{\text{Lower Bound} - \text{CPLEX Upper Bound}}{\text{CPLEX Upper Bound}} \times 100.
\]

The results indicated that the quality of the upper bounds generated when we solve the subproblems to optimality (Scheme 2) is almost the same as the quality of the upper bounds when we use the dual algorithm (Scheme 1) or the linear programming relaxation (Scheme 3). The difference in the quality of the solutions was no more than 0.2\%. A main incentive for using the primal-dual algorithm to solve the single-commodity, multi-facility lot-sizing problems \((SP_{1k})\) is the potential computational savings when solving the multi-commodity problem.

The results from Section 2.8 showed that primal-dual algorithm took little time to solve each single-commodity problem; however, we were interested in gauging the real savings for the multi-commodity, multi-retailer lot-sizing problem. Table 3–4 presents the CPU running times of the three algorithms. The running time of Scheme 1 were much smaller in all cases. These savings are due to the performance of the dual algorithm.

The quality of the lower bounds generated using either scheme of the Lagrangean decomposition algorithm or the linear programming relaxation of extended problem
Table 3-3: Quality of lower bounds (in %) from Lagrangean decomposition

<table>
<thead>
<tr>
<th>Problem</th>
<th>LP-Ex</th>
<th>Scheme1</th>
<th>Scheme2</th>
<th>Scheme3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.67</td>
<td>0.68</td>
<td>0.67</td>
<td>0.67</td>
</tr>
<tr>
<td>2</td>
<td>0.38</td>
<td>0.39</td>
<td>0.38</td>
<td>0.38</td>
</tr>
<tr>
<td>3</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
</tr>
<tr>
<td>4</td>
<td>0.11</td>
<td>0.11</td>
<td>0.11</td>
<td>0.11</td>
</tr>
<tr>
<td>5</td>
<td>0.07</td>
<td>0.09</td>
<td>0.07</td>
<td>0.07</td>
</tr>
<tr>
<td>6</td>
<td>0.06</td>
<td>0.10</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>7</td>
<td>2.47</td>
<td>2.50</td>
<td>2.48</td>
<td>2.48</td>
</tr>
<tr>
<td>8</td>
<td>1.18</td>
<td>1.20</td>
<td>1.18</td>
<td>1.18</td>
</tr>
<tr>
<td>9</td>
<td>0.46</td>
<td>0.46</td>
<td>0.46</td>
<td>0.46</td>
</tr>
<tr>
<td>10</td>
<td>0.30</td>
<td>0.31</td>
<td>0.31</td>
<td>0.31</td>
</tr>
<tr>
<td>11</td>
<td>1.19</td>
<td>1.20</td>
<td>1.19</td>
<td>1.19</td>
</tr>
<tr>
<td>12</td>
<td>2.24</td>
<td>2.24</td>
<td>2.24</td>
<td>2.24</td>
</tr>
<tr>
<td>13</td>
<td>2.36</td>
<td>2.36</td>
<td>2.36</td>
<td>2.36</td>
</tr>
<tr>
<td>14</td>
<td>2.97</td>
<td>2.97</td>
<td>2.97</td>
<td>2.97</td>
</tr>
<tr>
<td>15</td>
<td>3.18</td>
<td>3.19</td>
<td>3.18</td>
<td>3.18</td>
</tr>
<tr>
<td>16</td>
<td>0.75</td>
<td>0.76</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>17</td>
<td>0.81</td>
<td>0.82</td>
<td>0.81</td>
<td>0.81</td>
</tr>
<tr>
<td>18</td>
<td>0.87</td>
<td>0.89</td>
<td>0.87</td>
<td>0.87</td>
</tr>
<tr>
<td>19</td>
<td>0.88</td>
<td>0.90</td>
<td>0.88</td>
<td>0.88</td>
</tr>
<tr>
<td>20</td>
<td>0.92</td>
<td>0.93</td>
<td>0.92</td>
<td>0.92</td>
</tr>
<tr>
<td>21</td>
<td>0.86</td>
<td>0.88</td>
<td>0.86</td>
<td>0.86</td>
</tr>
<tr>
<td>22</td>
<td>0.56</td>
<td>0.57</td>
<td>0.56</td>
<td>0.56</td>
</tr>
<tr>
<td>23</td>
<td>0.58</td>
<td>0.59</td>
<td>0.58</td>
<td>0.58</td>
</tr>
<tr>
<td>24</td>
<td>0.85</td>
<td>0.87</td>
<td>0.85</td>
<td>0.85</td>
</tr>
<tr>
<td>25</td>
<td>1.34</td>
<td>1.39</td>
<td>1.34</td>
<td>1.35</td>
</tr>
</tbody>
</table>
formulation was almost identical. The results indicated that as the number of commodities increased, the error reported from CPLEX and Lagrangean decomposition decreased (problem classes 1 to 10). For these problem classes the

solution from Lagrangean decomposition was less than 2% from optimal and the running times of Scheme1 were smaller compared to CPLEX. However, the increase in the number of commodities affected the running time of Lagrangean decomposition algorithm because of the increase in the number of subproblems (SP_{1k}) to be solved at every iteration.

Increasing the capacity tightness coefficient δ (problem classes 7 to 10) made the problems easier. It is well-known that the uncapacitated lot-sizing problem is easier
to solve (capacitated lot-sizing problem, different than the uncapacitated problem is NP-hard).

The results showed that problem classes 11 to 15 were quite challenging. As the number of facilities increases we observed a monotonic increase in the duality gap. The results suggested that adding the facility selection dimension to the classical multi-commodity lot-sizing problem has quite an effect on problem complexity.

The running times of all algorithms for problem classes 16 to 21 were the highest. One of the reasons for this to happen is the size of the network for these problems (Table 3–1). Setup costs appeared to have a significant impact on the duality gap (problem classes 22 to 25). Problem class 22 presented an average error of 1.20% compared to 6.44% for the problem class 25. This result is not surprising since increased setup costs widen the gap between (SP$_2$), which ignores the setup costs and (SP$_1$). Moreover, since our original problem is a MILP with binary setup variables, as the setup costs increase the problem behaves closer to a combinatorial problem than a linear programming problem.

3.2 Single-Commodity, Multi-Retailer, Multi-Facility Lot-Sizing Problem

The satisfaction of the demand for products of a set of customers involves several complex processes. In the past, this often forced practitioners and researchers to investigate these processes separately. As mentioned by Erengüc et al. [30], the traditional way of managing operations in a competitive market place suggested that companies competing on price will sacrifice their flexibility in offering new products or satisfying new demands from their customers. The competition and the evolution of hardware and software capabilities has offered companies the possibility of considering coordinated decisions between processes in the supply chain.

In this section we propose a class of optimization models that consider the integration of decisions on production, transportation and inventory in a dynamic supply chain consisting of a number of retailers and facilities. We call this model the single-commodity, multi-retailer, multi-facility lot-sizing problem. This model
estimates the total cost of a given logistics distribution network, including production, inventory holding, and transportation costs. The evaluation is performed for a typical planning period in the future.

This problem considers a set of plants where a single product type is produced. This is a special case of the supply chain optimization problems we discussed in Chapter 1. Chapter 4 considers the more general problem, where a number of commodities are produced in the plants and the plants face production and transportation capacity constraints.

3.2.1 Problem Formulation

In this section we consider a class of multi-facility, multi-retailer production-distribution problems. Let \( R \) denote the number of retailers. Demand of retailer \( j \) in period \( t \) is given by \( b_{jt} \). The unit transportation costs from facility \( i \) to retailer \( j \) in period \( t \) are \( c_{ijt} \). In this discussion we assume that transportation and inventory cost functions are linear, and the production cost function is of the fixed charge type.

The multi-facility, multi-retailer problem can be formulated as follows:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{F} \sum_{t=1}^{T} (p_{it}q_{it} + s_{it}y_{it} + h_{it}I_{it} + \sum_{j=1}^{R} c_{ijt}x_{ijt}) \\
\text{subject to} & \quad (MR) \\
q_{it} + I_{i,t-1} &= \sum_{j=1}^{R} x_{ijt} + I_{it} \quad i = 1, \ldots, F; t = 1, \ldots, T \quad (3.24) \\
\sum_{i=1}^{F} x_{ijt} &= b_{jt} \quad j = 1, \ldots, R; t = 1, \ldots, T \quad (3.25) \\
q_{it} &\leq \sum_{j=1}^{R} b_{j,t}y_{it} \quad i = 1, \ldots, F; t = 1, \ldots, T \quad (3.26) \\
x_{ijt}, I_{it}, q_{it} &\geq 0 \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T \quad (3.27) \\
y_{it} &\in \{0, 1\} \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T \quad (3.28)
\end{align*}
\]
Decision variables $x_{ijt}$ represents the quantity transported from facility $i$ to retailer $j$ in time period $t$. Constraints (3.24) model the balance between the inflow, storage, and outflow at facility $i$ in period $t$. Constraints (3.25) make sure that retailer’s demand is satisfied. Constraints (3.26) relate the fixed and variable production costs (the production can be initiated once the setup cost is paid). Figure 1–1 gives a network representation of this problem.

The multi-retailer, multi-facility model we propose helps managers to answer questions that arise in managing the production and distribution network. Obviously, the most accurate answers are given when the problem is solved to optimality. However, this is a difficult task since the problem we present is $NP$-hard. This problem can be classified as a network flow problem with fixed charge cost functions. Several special cases of the single-commodity network flow problem have been shown to be $NP$-hard: for bipartite networks (Johnson et al. [63]), for single-source networks and constant fixed-to-variable cost ratio (Hochbaum and Segev [60]), and the case of zero variable costs (Lozovanu [76]). For the special case of this model when there is only one retailer, Wu and Golbasi [106] show that the problem is $NP$-hard when the holding costs are not restricted in sign.

We present the extended formulation of this problem as well. We do this by splitting the variables $q_{it}$ by destination into variables $q_{ijt\tau}$ ($\tau = t, \ldots, T$), where $\tau$ denotes the period and $j$ represents the facility for which production takes place. The split of the production variables implies the following:

$$q_{it} = \sum_{j=1}^{R} \sum_{\tau=t}^{T} q_{ijt\tau}$$  \hspace{1cm} (3.29)$$

$$x_{ijt} = \sum_{s=1}^{t} q_{ijst}$$ \hspace{1cm} (3.30)$$

$$I_{it} = \sum_{\tau=1}^{t} q_{i\tau} - \sum_{j=1}^{R} \sum_{\tau=1}^{T} x_{ijt\tau} = \sum_{j=1}^{R} \sum_{s=1}^{t} \sum_{\tau=t}^{T} q_{ijst\tau} - \sum_{s=1}^{t} q_{ijst}$$ \hspace{1cm} (3.31)$$
We can re-formulate problem (MR) as follows:

\[
\text{minimize } \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{j=1}^{R} \sum_{\tau=t}^{T} c_{ij\tau} q_{ij\tau} + s_{it} y_{it} \\
\text{subject to (Ex-MR)}
\]

\[
\sum_{i=1}^{F} \sum_{t=1}^{\tau} q_{ij\tau} = b_{j\tau} \quad j = 1, \ldots, R; \tau = 1, \ldots T \\
q_{ij\tau} \leq b_{j\tau} y_{it} \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots T; t \leq \tau \leq T \\
q_{ij\tau} \geq 0 \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots T; t \leq \tau \leq T \\
y_{it} \in \{0, 1\} \quad i = 1, \ldots, F; t = 1, \ldots T
\]

where \( c_{ij\tau} = p_{it} + c_{ij\tau} + \sum_{s=t+1}^{\tau} h_{is} \). The variable unit cost on the arcs of the extended network consist of the production unit cost at facility \( i \) in period \( t \), the transportation unit cost from facility \( i \) to retailer \( j \) in period \( \tau \), and the total unit holding cost at facility \( i \) from production period \( t \) to the shipping period \( \tau \).

**Proposition 3.2.1** The optimal cost of linear programming relaxation of the extended formulation of multi-facility, multi-retailer lot-sizing problem (Ex-MR) is at least as high as the optimal cost of linear programming relaxation of original formulation (MR).

**Proof:** Every feasible solution to the linear programming relaxation of extended formulation of multi-facility, multi-retailer lot-sizing problem (Ex-MR) can be transformed to a solution to linear programming relaxation of formulation (MR) using equations (3.29), (3.30) and (3.31). It follows that the optimal solution of linear programming relaxation of (Ex-MR) can be transformed to a feasible solution (not necessary the optimal solution) to linear programming relaxation of (MR). Computational results show in fact that the linear programming relaxation of (Ex-MR) gives solutions that are close to optimal. \( \square \)
3.2.2 Primal-Dual Algorithm

The dual of the linear programming relaxation of (Ex-MR) has a special structure. Below we present the linear programming relaxation and the corresponding dual of the formulation (Ex-MR).

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{F} \sum_{t=1}^{T} \left[ \sum_{j=1}^{R} \sum_{\tau=t}^{T} c_{ijt\tau} q_{ijt\tau} + s_{it} y_{it} \right] \\
\text{subject to} & \quad (LP-MR) \\
& \quad (3.32), (3.33), (3.34) \\
& \quad y_{it} \geq 0 \quad i = 1, \ldots, F; t = 1, \ldots, T \\
& \quad (3.36)
\end{align*}
\]

The dual problem reads

\[
\begin{align*}
\text{maximize} & \quad \sum_{t=1}^{T} \sum_{j=1}^{R} b_{jt} v_{jt} \\
\text{subject to} & \quad (D-MR) \\
& \quad \sum_{\tau=t}^{T} \sum_{j=1}^{R} b_{jt} w_{ijt\tau} \leq s_{it} \quad i = 1, \ldots, F; t = 1, \ldots, T \\
& \quad v_{jt\tau} - w_{ijt\tau} \leq \tau_{ijt\tau} \quad i = 1, \ldots, F; t = 1, \ldots, T; t \leq \tau \leq T \\
& \quad w_{ijt\tau} \geq 0 \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T \\
& \quad t \leq \tau \leq T.
\end{align*}
\]

In an optimal solution to (D-MR), both constraints \( w_{ijt\tau} \geq 0 \) and \( w_{ijt\tau} \geq v_{jt\tau} - \tau_{ijt\tau} \) should be satisfied. Since \( w_{ijt\tau} \) is not in the objective function, we can replace it with \( w_{ijt\tau} = \max(0, v_{jt\tau} - \tau_{ijt\tau}) \). This leads to the following condensed dual formulation:

\[
\begin{align*}
\text{maximize} & \quad \sum_{t=1}^{T} \sum_{j=1}^{R} b_{jt} v_{jt} \\
\end{align*}
\]
subject to \((D-MR^*)\)

\[
\sum_{\tau=t}^{T} \sum_{j=1}^{R} b_{j\tau} \max(0, v_{j\tau} - \bar{c}_{ij\tau}) \leq s_{it} \quad i = 1, \ldots, F; t = 1, \ldots, T.
\]

3.2.3 Intuitive Understanding of the Dual Problem

In this section we give an intuitive interpretation of the relationship between primal-dual solutions of \((Ex-MR)\). Suppose the linear programming relaxation of \((Ex-MR)\) has an optimal solution \((q^*, y^*)\) that is integral. Let \(\Theta = \{(i, t)|y_{it}^* = 1\}\) and let \((v^*, w^*)\) denote an optimal dual solution. The complimentary slackness conditions for this problem are as follows:

\((C_1)\) \(y_{it}^*[s_{it} - \sum_{j=1}^{R} \sum_{\tau=t}^{T} b_{j\tau} w_{ij\tau}] = 0\) for \(i = 1, \ldots, F; t = 1, \ldots, T\)

\((C_2)\) \(q_{ij\tau}^*[\bar{c}_{ij\tau} - v_{j\tau}^* + w_{ij\tau}^*] = 0\) for \(i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T; t \leq \tau \leq T\)

\((C_3)\) \(w_{ij\tau}^*[q_{ij\tau}^* - b_{j\tau} y_{it}^*] = 0\) for \(i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T; t \leq \tau \leq T\)

\((C_4)\) \(v_{j\tau}^*[b_{j\tau} - \sum_{i} \sum_{\tau=1}^{t} q_{ij\tau}^*] = 0\) for \(j = 1, \ldots, R; t = 1, \ldots, T\).

By conditions \((C_1)\), if a facility produces in a particular time period, the set-up cost must be fully paid (i.e., if \((i, t) \in \Theta\), then \(s_{it} = \sum_{j=1}^{R} \sum_{\tau=t}^{T} b_{j\tau} w_{ij\tau}\)). Consider conditions \((C_3)\). Now, if facility \(i\) produces in period \(t\), but demand of retailer \(j\) in that period is satisfied from the inventory from a previous period or from production (inventory) at a different facility \((q_{ijtt}^* = 0\) and \(q_{ijtt}^* - b_{j\tau} y_{it}^* \neq 0\), then \(w_{ijtt}^* = 0\). This implies that the price paid for the product will contribute to set-up cost of only the period when the product is produced.

By conditions \((C_2)\), if \(q_{ij\tau}^* > 0\), then \(v_{j\tau}^* = \bar{c}_{ij\tau} + w_{ij\tau}^*\). Thus, we can think of \(v_{j\tau}^*\) as the total cost (per unit of demand) in period \(\tau\) for retailer \(j\). Of this amount, \(\bar{c}_{ij\tau}\)
Figure 3–3: Dual algorithm

goes to pay for production and inventory holding costs, and \( w^*_{ijt\tau} \) is the contribution to the production set-up cost.

3.2.4 Outline of the Primal-Dual Algorithm

Suppose that the optimal values of the first \( k - 1 \) dual variables of (D-MR) are known. Let the index \( k \) be such that \( k = (\tau - 1) \times R + l \), where \( \tau = 1, \ldots, T \) and \( l = 1, \ldots, R \). Then, to be feasible, the \( k \)-th dual variable \( (v_{l\tau}) \) must satisfy the following constraints:

\[
\begin{align*}
    b_{l\tau} \max(0, v_{l\tau} - \bar{c}_{ilt\tau}) & \leq M_{ilt,\tau-1} = s_{it} - \\
    \sum_{j=1}^{R} \sum_{s=t}^{\tau-1} b_{js} \max(0, v_{js}^* - \bar{c}_{ijts}) & - \sum_{j=1}^{t-1} b_{jt} \max(0, v_{jt}^* - \bar{c}_{ijt\tau})
\end{align*}
\]  

(3.37)

for all \( i = 1, \ldots, F \) and \( t = 1, \ldots, \tau \). In order to maximize the dual problem we should assign \( v_{l\tau} \) the largest value satisfying these constraints. When, \( b_{l\tau} > 0 \), this value is

\[
v_{l\tau} = \min_{i=1,\ldots,F;\tau \geq t} \{ \bar{c}_{ilt\tau} + \frac{M_{ilt,\tau-1}}{b_{l\tau}} \} \tag{3.38}
\]

Note that if \( M_{ilt\tau-1} \geq 0 \) implies \( v_{l\tau} \geq \bar{c}_{ilt\tau} \). The dual solution found using equation (3.38) may not necessarily satisfy the complimentary slackness conditions. However, a dual feasible solution can be obtained simply by calculating the value of the dual
\[ y_{ik} = 0, \quad q_{ijk\tau} = 0, \quad i = 1, \ldots, F; \quad j = 1, \ldots, R; \quad k = 1, \ldots, T; \quad \tau \geq k \]

\[ P = \{(j, k)|b_{jk} > 0, \quad for \quad j = 1, \ldots, R; \quad k = 1, \ldots, T\} \]

**Start**: \( \tau = \max k \in P, \quad k = 0 \)

**Step 1**: for \( i = 1 \) to \( F \) do

  for \( j = 1 \) to \( R \) do

    repeat \( k = k + 1 \)

    until \( M_{ijk\tau} = 0 \) and \( \bar{c}_{ijk\tau} - v_{j\tau} - w_{ijk\tau} = 0 \)

    \( y_{ik} = 1 \), and \( i^* = i, \quad k^* = k \), go to Step 3

  enddo

enddo

**Step 2**: for \( i = k^* \) to \( T \) do

  for \( j = 0 \) to \( R \) do

    if \( \bar{c}_{i^*j^*k^*t} - v_{jt} - w_{i^*j^*k^*t} = 0 \)

    then \( q_{i^*j^*k^*t} = b_{jt}, \quad P = P - (j, t) \)

  enddo

enddo

**Step 3**: if \( P \neq \emptyset \) then go to Start

---

Figure 3–4: Primal algorithm

variables sequentially (Figure 3–3). A backward construction algorithm can then be used to generate primal feasible solutions (Figure 3–4). For the primal-dual set of solutions to be optimal, the complimentary slackness conditions should be satisfied.

**Proposition 3.2.2** The solutions obtained with the primal and dual algorithms are feasible and they always satisfy the complimentary slackness conditions \((C_1)\) and \((C_2)\).

**Proof**: It is clear that the primal and dual solutions generated are feasible by construction. Since the primal algorithm sets \( q_{ijtr} > 0 \) only when \( \bar{c}_{ijtr} - v_{jt} - w_{ijtr} = 0 \), the solution satisfies conditions \((C_2)\). The dual algorithm constructs solutions by making sure that equation (2.23) is satisfied. Therefore, the dual solutions are always such that \( b_{j,\tau+1}w_{ijt,\tau+1} \leq M_{ijtr} \). If \( M_{ijtr} = 0 \), then \( w_{ijt,\tau+1} = 0 \), and, since the dual algorithm sets \( M_{ijt,\tau+1} = M_{ijtr} - b_{j,\tau+1}w_{ijt,\tau+1} \), we also have \( M_{ijt,\tau+1} = 0 \). Continuing this way it is clear that if at some point in the calculation we get \( M_{ijtr} = 0 \), we subsequently obtain...
\[ M_{ijt\tau} = M_{ijt,\tau+1} = \ldots = M_{iRtT} = 0 \]

and

\[ w_{ijt\tau} = w_{ijt,\tau+1} = \ldots = w_{iRtT} = 0 \]

The primal algorithm sets \( y_{it} = 1 \) only when \( M_{ijt\tau} = 0 \); this implies that conditions \((C_1)\) will always be satisfied.

Hence, one can determine whether the solution obtained with the primal and dual algorithms is optimal by checking if conditions \((C_3)\) are satisfied, or if the objective function values from the primal and dual algorithms are equal.

3.2.5 Computational Results

In order to test the performance of our algorithm, as in previous sections, we randomly generated a set of test problems and compared their computation times and solution quality to the general purpose solver CPLEX. We generated feasible solutions to our problem using the primal algorithm and lower bounds using the dual algorithm.

The nominal case problem has the following characteristics:

- Production set-up costs \( s_{it} \sim U[200, 300] \)
- Production variable costs \( p_{it} \sim U[5, 15] \)
- Holding costs \( h_{it} \sim U[5, 15] \)
- Number of retailers \( R = 60 \)

The transportation variable costs are generated the same way as described in Section 2.8. Seasonal demands are randomly generated in the same way as presented in Baker et al. [5] and Chen et al. [22].

\[ b_t = 200 + \sigma z_t + \alpha \sin \left( \frac{2\pi}{d} (t + d/4) \right) \]

In our test problems we take \( \sigma = 67, \alpha = 125 \) and \( d = 12 \).

The characteristics of the problem classes are presented in Table 3–5. For each problem class we generate 20 problem instances and we report on the average error.
bounds and average running times. The error bound for each problem instance is calculated as follows: \( \text{Error (\%)} = \frac{\text{Primal Solution} - \text{Dual Solution}}{\text{Dual Solution}} \times 100. \)

Table 3-6 presents the error bounds from the primal-dual algorithm. We were able to get the optimal solutions only for problem classes 1 and 2. For these two problem classes linear programming relaxation of the extended formulation gave the optimal solution. However, because of the the size of the problems (problem classes 3 to 12), CPLEX ran out of memory without providing either an integer feasible solution or a lower bound. Results on Table 3-6 indicate that as the value of set-up cost increased, the problems became more difficult. This affected the performance of the primal-dual algorithm.

For all problem classes, when set-up costs are uniformly distributed on the intervals \([200, 300]\) and \([200, 900]\), the error gap was less than 0.60\% and the running times less than 141 cpu seconds. The maximum error reported is 4.073\%. This corresponds to problem class 3 with set-up costs uniformly distributed in the interval \([1200, 1500]\). Results in Tables 3–6 and 3–7 indicate that increasing the number of facilities and the length of the time horizon affected the performance of the primal and dual algorithm.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Facilities</th>
<th>Periods</th>
<th>Nodes</th>
<th>Arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>24</td>
<td>1,921</td>
<td>360,480</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>24</td>
<td>2,161</td>
<td>540,720</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>24</td>
<td>2,401</td>
<td>720,960</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>48</td>
<td>3,841</td>
<td>1,412,160</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>48</td>
<td>4,321</td>
<td>2,118,240</td>
</tr>
<tr>
<td>6</td>
<td>40</td>
<td>48</td>
<td>4,801</td>
<td>2,824,320</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>96</td>
<td>7,681</td>
<td>5,589,120</td>
</tr>
<tr>
<td>8</td>
<td>30</td>
<td>96</td>
<td>8,641</td>
<td>8,383,680</td>
</tr>
<tr>
<td>9</td>
<td>40</td>
<td>96</td>
<td>9,601</td>
<td>11,178,240</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>192</td>
<td>15,361</td>
<td>22,237,440</td>
</tr>
<tr>
<td>11</td>
<td>30</td>
<td>192</td>
<td>17,281</td>
<td>33,356,160</td>
</tr>
<tr>
<td>12</td>
<td>40</td>
<td>192</td>
<td>19,201</td>
<td>44,474,880</td>
</tr>
</tbody>
</table>
Table 3–6: Error bounds (in %) of primal-dual heuristic

<table>
<thead>
<tr>
<th>Problem</th>
<th>Set-Up Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>200-300</td>
</tr>
<tr>
<td>1</td>
<td>0.249</td>
</tr>
<tr>
<td>2</td>
<td>0.343</td>
</tr>
<tr>
<td>3</td>
<td>0.423</td>
</tr>
<tr>
<td>4</td>
<td>0.246</td>
</tr>
<tr>
<td>5</td>
<td>0.321</td>
</tr>
<tr>
<td>6</td>
<td>0.395</td>
</tr>
<tr>
<td>7</td>
<td>0.248</td>
</tr>
<tr>
<td>8</td>
<td>0.333</td>
</tr>
<tr>
<td>9</td>
<td>0.419</td>
</tr>
<tr>
<td>10</td>
<td>0.250</td>
</tr>
<tr>
<td>11</td>
<td>0.341</td>
</tr>
<tr>
<td>12</td>
<td>0.413</td>
</tr>
</tbody>
</table>

We next randomly generated a second group of problems for which demand is uniformly distributed in the following intervals: [20, 40], [40, 100], [100, 200], [200, 400] and [400, 1000]. We rerun problem classes 4, 5 and 6 for each demand distribution (problem classes 13 to 27). For example, for problem classes 13, 14 and 15, demand is uniformly distributed in [20, 40] and the problem characteristics are the same as the characteristics of problems 4, 5 and 6. This gives a total of 15 problem classes and 300 problem instances. CPLEX failed to solve problem classes 13 to 27. Table 3–8

Table 3–7: Running times (in seconds) of primal-dual heuristic

<table>
<thead>
<tr>
<th>Problem</th>
<th>Set-Up Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>200-300</td>
</tr>
<tr>
<td>1</td>
<td>1.16</td>
</tr>
<tr>
<td>2</td>
<td>1.58</td>
</tr>
<tr>
<td>3</td>
<td>2.03</td>
</tr>
<tr>
<td>4</td>
<td>4.15</td>
</tr>
<tr>
<td>5</td>
<td>6.15</td>
</tr>
<tr>
<td>6</td>
<td>8.09</td>
</tr>
<tr>
<td>7</td>
<td>16.78</td>
</tr>
<tr>
<td>8</td>
<td>25.15</td>
</tr>
<tr>
<td>9</td>
<td>33.68</td>
</tr>
<tr>
<td>10</td>
<td>74.42</td>
</tr>
<tr>
<td>11</td>
<td>109.83</td>
</tr>
<tr>
<td>12</td>
<td>140.55</td>
</tr>
</tbody>
</table>
presents the error and running times of the primal-dual heuristic for these problems. The average running time of the primal algorithm was less than 5.22 cpu seconds for all problem classes and less than 3.23 for the dual algorithm. Notice that the error decreased as the average value of demand increased. For these problem classes, everything else kept the same, an increase in demand affected the ratio of total variable cost to total fixed cost. Increasing this ratio generally makes the problems easier (Hochbaum and Segev [60]).

Table 3–8: Results of primal-dual heuristic

<table>
<thead>
<tr>
<th>Problem</th>
<th>Error (%)</th>
<th>Time (sec)</th>
<th>Primal</th>
<th>Dual</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>3.94</td>
<td>2.54</td>
<td>1.63</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>5.08</td>
<td>3.88</td>
<td>2.41</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>5.98</td>
<td>5.20</td>
<td>3.18</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>1.34</td>
<td>2.54</td>
<td>1.64</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>1.78</td>
<td>3.87</td>
<td>2.42</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>2.18</td>
<td>5.22</td>
<td>3.21</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>0.43</td>
<td>2.54</td>
<td>1.66</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.59</td>
<td>3.88</td>
<td>2.42</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>0.73</td>
<td>5.20</td>
<td>3.20</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0.14</td>
<td>2.55</td>
<td>1.68</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>0.19</td>
<td>3.87</td>
<td>2.45</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0.25</td>
<td>5.22</td>
<td>3.22</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0.03</td>
<td>2.55</td>
<td>1.67</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>0.04</td>
<td>3.88</td>
<td>2.45</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>0.06</td>
<td>5.22</td>
<td>3.23</td>
<td></td>
</tr>
</tbody>
</table>

3.3 Multi Facility Lot-Sizing Problem with Fixed Charge Transportation Costs

In this section we discuss the uncapacitated multi-facility, multi-retailer problem with fixed charge transportation costs and linear production and inventory costs. Usually, when shipments are sent from a facility to a retailer, a fixed charge is paid (for example, the cost of the paperwork necessary) to initiate the shipment plus a variable cost for every unit transported. Therefore, modelling the transportation cost function as a fixed charge cost function makes sense. The following is the MILP formulation of the problem:
minimize \[ \sum_{i=1}^{F} \sum_{t=1}^{T} (p_{it}q_{it} + \sum_{j=1}^{R} (c_{ijt}x_{ijt} + s_{ijt}y_{ijt}) + h_{it}I_{it}) \]

subject to \[
I_{i,t-1} + q_{it} = \sum_{j=1}^{R} x_{ijt} + I_{i,t} \quad i = 1, \ldots, F; t = 1, \ldots, T
\]

\[
\sum_{i=1}^{F} x_{ijt} = b_{jt} \quad j = 1, \ldots, R; t = 1, \ldots, T
\]

\[
q_{it} \leq \sum_{j=1}^{R} b_{jt}y_{ijt} \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T
\]

\[
q_{it}, I_{i,t}, x_{ijt} \geq 0 \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T
\]

\[
y_{it} \in \{0, 1\} \quad i = 1, \ldots, F; t = 1, \ldots, T
\]

\(s_{ijt}\) is the transportation fixed charge cost from facility \(i\) to retailer \(j\) in period \(t\).

A nice property of this problem is that its linear programming relaxation gives the optimal solution. An optimal solution to (MR-T) has the following properties: (i) in a facility, production will take place if there are no inventories (the zero inventory policy) (ii) the demand at a retailer will be satisfied by production or inventory from exactly one facility (iii) the quantity produced in a facility is equal to the demand of at least one retailer, for at least one time period (these time periods do not need to be successive). These properties imply that in an optimal solution, if a transportation arc is used, the total demand is shipped. The transportation cost function then consists of only two points, \(x_{ijt} = 0\) and \(x_{ijt} = b_{jt}\).

The transportation cost function is separable by arc. For each transportation arc, the linear approximation of the fixed charge cost function passes through \(x_{ijt} = 0\) and \(x_{ijt} = b_{jt}\). This shows that the linear approximation exactly represents the transportation cost function. A well-known result of linear programming is the following: when a linear cost function is minimized over a convex set, the solution corresponds to an extreme point. In this case, the extreme points correspond to
Figure 3-5: Fixed charge transportation cost function

$x_{ijt} = 0$ or $x_{ijt} = b_{jt}$. Therefore, the solution to (MR-T) and its linear programming relaxation will be the same.

The extended formulation of (MR-T) is the following:

$$\text{minimize} \ \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{j=1}^{R} \sum_{\tau=1}^{T} c_{ijt\tau} q_{ijt\tau} + s_{ijt} y_{ijt}$$

subject to

$$\sum_{i=1}^{F} \sum_{t=1}^{T} q_{ijt\tau} = b_{j\tau} \quad j = 1, \ldots, R; \tau = 1, \ldots, T$$

$$\sum_{t=1}^{\tau} q_{ijt\tau} \leq b_{j\tau} y_{ij\tau} \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T; t \leq \tau \leq T$$

$$q_{ijt\tau} \geq 0 \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T; \tau \leq T$$

$$y_{ijt} \in \{0, 1\} \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T;$$

where $c_{ijt\tau} = p_{it} + c_{ij\tau} + \sum_{s=t+1}^{\tau} h_{is}$.

For the same reason, the same result holds true for the linear programming relaxation of extended formulation, its linear programming relaxation gives integer solutions.

3.4 Conclusions

In this chapter we discuss the capacitated multi-commodity, multi-facility lot-sizing problem, the single-commodity, multi-facility, multi-retailer lot-sizing problem as well as a special case of this last problem with fixed charge transportation costs and linear production costs.
The multi-commodity and multi-facility problem is solved using a Lagrangean decomposition based heuristic. The decomposition separates the problem into two subproblems that are computationally easier to solve. The decomposition is performed in such a way that it provides interesting managerial insights.

The multi-facility and multi-retailer problem is solved using a primal-dual algorithm. The performance of the algorithm is tested on two groups of randomly generated problems. In the first group of problems demand shows seasonality, and in the second group demand is uniformly distributed. Results indicate that the performance of the primal-dual algorithm depends on the value of set-up costs, the value of demand, the number of facilities and the number of periods.

In this chapter we also discussed a class of uncapacitated, multi-facility and multi-retailer problems with linear production and inventory costs and fixed charge transportation costs. We show that this class of problems has a nice property, the linear programming relaxation of the MILP formulations gives integer solutions.
CHAPTER 4
PRODUCTION-DISTRIBUTION PROBLEM

4.1 Introduction

The supply chain optimization problem we discuss in this chapter considers production, inventory, and transportation decisions in a dynamic environment. In particular, we model a two-stage supply chain, coordinating production, inventory, and distribution decisions. This supply chain consists of $F$ facilities and $R$ retailers. The facilities produce and store $K$ different commodities for which there is a demand over a planning horizon of length $T$. The problem is to find the production, inventory, and transportation quantities that satisfy demand at minimum cost.

We formulate the problem as a multi-commodity network flow problem with fixed charge cost function. We present a heuristic procedure that can be used to solve any multi-commodity network flow problem with fixed charge cost functions. This method, called the multi-commodity, dynamic slope scaling procedure (MCDSSP) is an extension of a procedure that was proposed by Kim and Pardalos [66] for the single-commodity fixed-charge network flow problem. MCDSSP approximates the fixed charge cost function by a linear cost function, and iteratively updates the coefficients of the linear approximation until no better solution is found.

We also propose a Lagrangean decomposition based algorithm to solve the production-distribution problem. The algorithm decomposes the problem into two subproblems. One of the subproblems can be further decomposed into $K$ single-commodity, multi-facility, multi-retailer lot-sizing problems. These single-commodity subproblems are solved using the primal-dual algorithm proposed in Section 3.2.2. In Section 4.2 we give a mathematical formulation of the production-distribution problem. In Section 4.3 we provide a detailed description of the MCDSSP, in Section
4.4 we discuss the Lagrangean decomposition algorithm and in Section 4.5 we present results from implementing the MCDSSP on a set of randomly generated problems. Section 4.6 concludes this chapter.

4.2 Problem Formulation

In this chapter we finally discuss and propose solution algorithms for the production-distribution problem we introduced in Section 1.3. The production-distribution planning problem is formulated as a multi-commodity network flow problem on a directed, single-source graph consisting of $T$ layers (Figure 1–1). Recall that each layer of the graph represents a time period. In each layer, a bipartite graph represents the transportation network between facilities and retailers. Facilities in successive time periods are connected through inventory arcs. There is a dummy source node with supply for each commodity equal to the corresponding total demand. Production arcs connect the dummy source node to each facility in every time period. The total number of nodes in the network is $|N| = (R + F)T + 1$, and the total number of arcs is $|A| = FT + FRT + F(T − 1)$. There are unit costs associated with each arc and each commodity. In addition, there are setup costs associated with each production and transportation arc.

Recall that we have made the following assumptions in our model:

- Inventory costs are linear.
- Transportation and production costs are nonlinear, in particular, fixed-charge cost functions.
- Backorders are not allowed.
- Products are stored at their production location until being transported to a retailer.
- The only capacity constraints are bundle capacity constrains on the production and transportation arcs, corresponding to a limited production capacity or truck capacity.

The production-distribution model we propose can easily be extended to count for backorders and inventories at the retailers by adding extra arcs in the network. Most of the decision variables and cost data are already defined in the previous chapters. The new variables that we introduce are
• $x_{ijtk}$ represents the amount of commodity $k$ transported from facility $i$ to retailer $j$ in period $t$

• $y_{ijt}$ represents the binary setup decision for transport from facility $i$ to retailer $j$ in period $t$

Since we have a decision variable for each commodity on every arc in the network, the total number of decision variables in the model is $K|A| = FKT + FRKT + FK(T - 1)$. In addition, the following new cost data is used:

• $c_{ijtk}$ denotes the unit transportation cost for commodity $k$ between facility $i$ and retailer $j$ in period $t$

• $s_{ijt}$ denotes the setup costs for transportation between facility $i$ and retailer $j$ in period $t$

Finally, we have

• $v_{it}$ is the bundle capacity for production at facility $i$ in period $t$

• $v_{ijt}$ is the bundle (e.g., truck) capacity for transportation between facility $i$ and retailer $j$ in period $t$

• $b_{jtk}$ is the demand for commodity $k$ at retailer $j$ in period $t$

The problem can now be formulated as a MILP as follows:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} \left( s_{itk} y_{itk} + p_{itk} q_{itk} + h_{itk} I_{itk} \right) + \sum_{i=1}^{F} \sum_{j=1}^{R} \sum_{t=1}^{T} \left( s_{ijt} y_{ijt} + \sum_{k=1}^{K} c_{ijtk} x_{ijtk} \right) \\
\text{subject to} & \quad \text{(PD)}
\end{align*}
\]

\[I_{i,t-1,k} + q_{itk} = \sum_{j=1}^{R} x_{ijtk} + I_{itk} \quad i = 1, \ldots, F; \ t = 1, \ldots, T; \ k = 1, \ldots, K\]

\[\sum_{i=1}^{F} x_{ijtk} = b_{jtk} \quad j = 1, \ldots, R; \ t = 1, \ldots, T; \ k = 1, \ldots, K\]

\[\sum_{k=1}^{K} q_{itk} \leq v_{it} \quad i = 1, \ldots, F; \ j = 1, \ldots, R; \ t = 1, \ldots, T\]

\[\sum_{k=1}^{K} x_{ijtk} \leq v_{ijt} y_{ijt} \quad i = 1, \ldots, F; \ j = 1, \ldots, R; \ t = 1, \ldots, T\]

\[q_{itk}, I_{itk}, x_{ijtk} \geq 0 \quad i = 1, \ldots, F; \ j = 1, \ldots, R; \ t = 1, \ldots, T; \ k = 1, \ldots, K\]

\[y_{ijt} \in \{0, 1\} \quad i = 1, \ldots, F; \ j = 1, \ldots, R; \ t = 1, \ldots, T.\]
This problem is NP-hard even in the simpler case when there is only one time period and a single commodity flowing in the network (Garey and Johnson [45]).

### 4.3 Dynamic Slope Scaling Procedure

Kim and Pardalos [66] developed a heuristic called the Dynamic Slope Scaling Procedure (DSSP) for the fixed-charge network flow problem with a single commodity, which was refined and tested in Ekşioğlu et al. [34]. In this section, we extend this heuristic to the multi-commodity case.

Since MCDSSP is a general heuristic that can be used to solve any multi-commodity problem with fixed charge cost functions, we give at first a description of the MCDSSP in this context.

#### 4.3.1 Multi-Commodity Network Flow Problem with Fixed Charge Cost Function

The multi-commodity network flow problem with fixed-charge arc cost functions has a broad area of applications, such as production and distribution of goods in a supply chain, or the distribution of messages in a communication network (see for instance, Magnanti and Wong [80], Gavish [47], Balakrishnan et al. [7], and Ahuja et al. [2, 3]). This problem is a generalization of the classical single-commodity network flow problem with fixed-charge costs. Several special cases of this single-commodity network flow problem were shown to be $NP$-hard such as bipartite networks (Johnson et al. [63]), single-source networks and constant fixed-to-variable cost ratio (Hochbaum and Segev [60]), and the case of zero variable costs (Lozovanu [76]). In addition, the single commodity network flow problem with general concave arc costs was studied by Garey and Johnson [45] and Guisewite and Pardalos [53].

Since the multi-commodity network flow problem with fixed-charge arc costs is a concave minimization problem, any exact general-purpose solution method for solving such problems can be used to solve the multi-commodity network flow problem with fixed-charge costs. Examples of such methods are branch-and-bound (Hirsch and Dantzig [59], Gray [51], Kennington and Unger [65], Barr et al. [11], Cabot and
Erengüç [20], Palekar et al. [87], and Lamar and Wallace [70]), vertex enumeration (Murty [84]), and dynamic programming (Erickson et al. [37]). However, such general purpose algorithms are often not adequate tools for solving large-scale instances of our problem efficiently. For instance, the linear programming relaxation of the traditional MILP formulation of the problem does not provide a tight lower bound. Therefore, standard simplex-based branch-and-bound methods that do not include cutting plane or column generation procedures are not likely to solve large instances of the problem in reasonable time. Therefore, more efficient special purpose algorithms have been proposed as well.

Bienstock and Günlük [14] used a simplex-based cutting plane approach. This approach offers an opportunity for continuous improvement of lower bounds through valid inequalities. Gavish [46] proposed Lagrangean relaxation, which not only exploits the structure of the problem, but facilitates the design of heuristics as well.

Cranic, Frangioni and Gendron [26] compared lower bounds generated using different Lagrangean relaxations of the capacitated multi-commodity network flow problem with fixed-charge costs. They showed that bundle methods used to optimize the Lagrangean duals are superior to subgradient methods, because they converge faster and are more robust with respect to problem characteristics. Gendron and Crainic [48] used a bounding procedure to solve the problem. Their procedure was based on generating lower bounds using Lagrangean relaxation (relaxing the bundling constraints) and generating upper bounds using a Resource Decomposition approach. Crainic et al. [27] solved the capacitated multi-commodity network flow problem with fixed-charge costs using a cutting plane algorithm combined with a Lagrangean relaxation.

The uncapacitated multi-commodity network flow problem with fixed-charge costs is not as difficult as the capacitated problem. Magnanti et al. [79] proposed a methodology to improve the performance of Benders decomposition when used to solve the uncapacitated problem. Holmberg and Hellstrand [61] presented a
Lagrangean heuristic within a branch-and-bound framework as a method for finding the exact optimal solution of the uncapacitated problem with single origins and destinations for each commodity.

**Problem description and formulation.** Consider a connected graph $G(\mathcal{N}, A)$, where $\mathcal{N} = \{1, \ldots, N\}$ is the set of nodes and $A \subseteq \mathcal{N} \times \mathcal{N}$ is the set of arcs. The number of commodities that need to be routed through this network is given by $K$, and the demand for commodity $k$ at node $i$ is denoted by $b_{ik}$. Each arc $(i, j) \in A$ has a capacity for each individual commodity denoted by $u_{ijk}$. In addition, each arc $(i, j) \in A$ has a bundling capacity jointly for all commodities, denoted by $v_{ij}$.

The decision variables are the quantities of flow of each commodity along each arc, and are denoted by $x_{ijk}$ ($(i, j) \in A; k = 1, \ldots, K$). We assume that the total cost of flow is separable in the arcs, and the cost of flow along arc $(i, j)$ is given by $f_{ij}(x_{ij1}, \ldots, x_{ijK})$.

The general minimum cost multi-commodity network flow problem can then be formulated as follows:

$$\begin{align*}
\text{minimize} \quad & \sum_{(i,j) \in A} f_{ij}(x_{ij1}, \ldots, x_{ijK}) \\
\text{subject to} \quad & \sum_{j: (j,i) \in A} x_{ijk} - \sum_{j: (i,j) \in A} x_{ijk} = b_{ik} \quad i = 1, \ldots, N; \ k = 1, \ldots, K \quad (4.1) \\
& \sum_{k=1}^{K} x_{ijk} \leq v_{ij} \quad (i, j) \in A \quad (4.2) \\
& x_{ijk} \leq u_{ijk} \quad (i, j) \in A; \ k = 1, \ldots, K \quad (4.3) \\
& x_{ijk} \geq 0 \quad (i, j) \in A; \ k = 1, \ldots, K.
\end{align*}$$

In this formulation, constraints (4.1) are the flow conservation constraints, (4.2) are the bundle capacity constraints, and (4.3) are the individual arc capacity constraints. It will be convenient to denote the feasible region of (P) by $X \subset \mathbb{R}^{K|A}$. 
Although not essential for the applicability of the heuristic we will propose in Section 4.3, we will assume that the arc cost functions have a fixed-charge structure. In particular, let \( s_{ij} \) represent the fixed-charge cost that is incurred whenever arc \((i, j) \in A\) is used. In addition, let \( c_{ijk} \) represent the variable per unit cost of moving commodity \( k \) along arc \((i, j)\). More formally, this yields

\[
f_{ij}(x_{ij1}, \ldots, x_{ijk}) = \begin{cases} 
0 & \text{if } \sum_{k=1}^{K} x_{ijk} = 0 \\
s_{ij} + \sum_{k=1}^{K} c_{ijk} x_{ijk} & \text{if } \sum_{k=1}^{K} x_{ijk} > 0.
\end{cases}
\]

The standard MILP reformulation of this problem can be obtained by introducing a binary setup variable \( y_{ij} \) corresponding to each arc \((i, j) \in A\). The arc cost functions can then be replaced by

\[
f_{ij}(x_{ij1}, \ldots, x_{ijk}) = s_{ij} y_{ij} + \sum_{k=1}^{K} c_{ijk} x_{ijk}
\]

where

\[
y_{ij} = \begin{cases} 
0 & \text{if } \sum_{k=1}^{K} x_{ijk} = 0 \\
1 & \text{if } \sum_{k=1}^{K} x_{ijk} > 0.
\end{cases}
\]

The MILP formulation of the fixed-charge multi-commodity network flow problem then reads

\[
\begin{array}{ll}
\text{minimize} & \sum_{(i,j) \in A} s_{ij} y_{ij} + \sum_{(i,j) \in A} \sum_{k=1}^{K} c_{ijk} x_{ijk} \\
\text{subject to} & \sum_{j: (j,i) \in A} x_{jik} - \sum_{j: (i,j) \in A} x_{ij} = b_{ik} \quad i = 1, \ldots, N; \quad k = 1, \ldots, K \\
& \sum_{k=1}^{K} x_{ijk} \leq v_{ij} y_{ij} \quad (i, j) \in A \\
& x_{ijk} \leq u_{ijk} \quad (i, j) \in A; \quad k = 1, \ldots, K \\
& x_{ijk} \geq 0 \quad (i, j) \in A; \quad k = 1, \ldots, K. \\
& y_{ij} \in \{0, 1\} \quad (i, j) \in A.
\end{array}
\]
In principle, standard solvers such as CPLEX can be used to solve formulation (Q) of the fixed-charge multi-commodity network flow problem.

4.3.2 Single-Commodity Case

The DSSP is a procedure that iteratively approximates the fixed-charge cost function by a linear function, and solves the corresponding linear programming problem. Note that each of the approximating linear programs has exactly the same set of constraints, and differs only with respect to the objective function coefficients. The motivation behind the DSSP is the fact that a concave function (such as the fixed-charge cost function), when minimized over a set of linear constraints, will have an extreme point solution. Therefore, there exists a linear cost function that yields the same optimal solution as the concave cost function. The procedure does not guarantee that the optimal solution to the fixed-charge network flow problem is indeed found. However, substantial experimental analysis indicates that the procedure yields high quality solutions that are close to the optimal solution.

4.3.3 Multi-Commodity Case

As for the single commodity case, the multi-commodity variant of the DSSP, which we will call MCDSSP, consists of an initialization phase and an update phase. In the former, we need to initialize the linear approximation of the fixed-charge cost function, and in the latter we need to update the linear approximation.

Initialization scheme for MCDSSP. Consider the MILP formulation (Q) of the fixed-charge multi-commodity network flow problem. The linear programming relaxation of this formulation relaxes the binary constraints on the variables $y_{ij}$, allowing them to assume any variable in $[0, 1]$. From equation (4.4), we can see that

$$y_{ij} \geq \frac{\sum_{k=1}^{K} x_{ijk}}{v_{ij}}$$

so that, without loss of optimality, we can let $y_{ij} = \frac{\sum_{k=1}^{K} x_{ijk}}{v_{ij}}$. 
Making this substitution yields the following formulation of the linear programming relaxation of (P):

\[
\text{minimize} \quad \sum_{(i,j) \in A} \sum_{k=1}^{K} \left( \frac{s_{ij}}{v_{ij}} + c_{ijk} \right) x_{ijk} \\
\text{subject to} \quad x \in X
\]

This motivates the initial linear approximation of (P) (Figure 4–1) as

\[
\bar{c}_{ijk}^{(0)} = \frac{s_{ij}}{v_{ij}} + c_{ijk} \quad \text{for } (i, j) \in A.
\]

Note that the initial linear cost function is a linear underestimator of the true concave cost function.

![Figure 4–1: Initial linear approximation of the fixed-charge cost function](image)

**Update scheme for the MCDSSP.** In the single commodity variant of the DSSP, the solution of a linear approximation is used as follows to construct a new linear objective function. For all arcs that are used in the solution, the new linear cost coefficient is chosen to be the average cost per unit shipped on that arc, measured using the true, fixed-charge arc cost function. If, in the multi-commodity case, the variable unit costs happen to be commodity-independent (i.e., \( c_{ijk} = c_{ij} \) for all \( k = 1, \ldots, K \) and for all \((i, j) \in A\)), we could use the same approach as in the single commodity case. Suppose that, in the \( \ell \)th iteration of the procedure, we have used the coefficients \( \bar{c}_{ij}^{(\ell)} \) to obtain the solution \( (x_{ijk}^{(\ell)}) \). The slopes can then be updated to
\[
\bar{c}^{(\ell+1)}_{ij} = \begin{cases} 
\frac{s_{ij}}{\sum_{k=1}^{K} x_{ijk}^{(\ell)}} + c_{ij} & \text{if } \sum_{k=1}^{K} x_{ijk}^{(\ell)} > 0 \\
c_{ij} & \text{otherwise.}
\end{cases}
\]

However, in general we need to somehow distribute the arc costs among the \( K \) commodities. In other words, we would like to find slopes \( c_{ij}^{(\ell+1)} \) that satisfy

\[
s_{ij} + \sum_{k=1}^{K} c_{ijk} x_{ijk}^{(\ell)} = \sum_{k=1}^{K} c_{ijk}^{(\ell+1)} x_{ijk}^{(\ell)}
\]

whenever \( \sum_{k=1}^{K} x_{ijk}^{(\ell)} > 0 \). Thus, when arc \((i, j)\) is used, we need to find a way to distribute the fixed-charge cost \( s_{ij} \) among the commodities flowing on arc \((i, j)\). It is clear that there is no unique way of accomplishing this. To characterize the possible ways of distributing the costs, we introduce a set of weights \( w_{ijk}^{(\ell)} \), and let

\[
\bar{c}_{ijk}^{(\ell+1)} = w_{ijk}^{(\ell)} s_{ij} + c_{ijk}.
\]

In general, these weights will actually be weight functions, with their values depending on the most recent approximate solution \( x^{(\ell)} \). To ensure that equation (4.4) holds, the weights should satisfy the following conditions:

\[
\sum_{k=1}^{K} w_{ijk}^{(\ell)} = 1 \quad \text{for all } (i, j) \in A
\]

\[
w_{ijk}^{(\ell)} \geq 0 \quad \text{for all } (i, j) \in A, k = 1, \ldots, K.
\]

We propose three different ways of choosing the weights (where we assume that \((i, j)\) is an arc that is used in the solution \( x^{(\ell)} \), i.e., \( \sum_{k=1}^{K} x_{ijk}^{(\ell)} > 0 \)).

**MCDSSP1:** Distribute the fixed-charge costs \( s_{ij} \) equally over the commodities that use arc \((i, j)\), independent of the flow quantities

\[
w_{ijk}^{(\ell)} = \frac{1}{\sum_{k=1}^{K} |x_{ijk}^{(\ell)}|}.
\]
where, $|x|^+ = \begin{cases} 
1 & \text{ if } x > 0 \\
0 & \text{ otherwise.}
\end{cases}$

This scheme implicitly assumes that the commodities flowing on arc $(i, j)$ have similar impacts on the total costs.

**MCDSSP2:** Distribute the fixed-charge costs $s_{ij}$ equally over all units flowing on arc $(i, j)$ as

$$w^{(\ell)}_{ijk} = \frac{x_{ijk}^{(\ell)}}{\sum_{k'=1}^{K} x_{ijk'}^{(\ell)}}.$$

This scheme allocates the setup costs among the commodities using arc $(i, j)$ based on their contribution to the total flow on that arc.

**MCDSSP3:** Distribute the fixed-charge costs $s_{ij}$ as follows:

$$w^{(\ell)}_{ijk} = \frac{c_{ijk} x_{ijk}^{(\ell)}}{\sum_{k'=1}^{K} c_{ijk'} x_{ijk'}^{(\ell)}}.$$

This scheme allocates the setup costs among the commodities using arc $(i, j)$ based on their contribution to the total costs incurred on that arc.

In the $\ell$th iteration of the MCDSSP, the following linear programming problem is solved:

$$\begin{aligned}
\text{minimize} & \quad \sum_{(i,j) \in A} \sum_{k=1}^{K} c_{ijk}^{(\ell)} x_{ijk}^{(\ell)} \\
\text{subject to} & \quad x \in X.
\end{aligned}$$

**Stopping criterion.** The heuristic will stop if one of the following conditions is met:

- the solutions found in two consecutive iterations are the same, i.e.,

$$x_{ijk}^{(\ell+1)} = x_{ijk}^{(\ell)} \text{ for all } (i, j) \in A \text{ and } k = 1, \ldots, K.$$

- no improvement has been found in the last $L$ iterations.
As in the single-commodity case, the MCDSSP does not guarantee convergence, or even monotonicity. Therefore, after each iteration of the updating procedure, we save the best solution found so far.

### 4.3.4 Production-Distribution Problem

Application of MCDSSP to solve the production-distribution problem (PD) is straightforward. The only slight difference with respect to the general case discussed in the previous sections is that some arcs have linear costs. Since inventory arcs have linear costs, the only cost coefficients that need to be initialized and updated are production and transportation cost coefficients.

The updating schemes for production and transportation costs are different. We use DSSP to update the cost coefficients for the production arcs and MCDSSP for the transportation arcs. The reason is that in the production cost function, each commodity has its own setup and variable unit costs, different from the transportation arcs where the setup cost is shared among all the commodities using that arc.

The total cost on a production arc is equal to

\[
\sum_{k=1}^{K} (s_{itk} + p_{itk}q_{itk}) \quad \text{for } i = 1, \ldots, F; t = 1, \ldots, T.
\]

The above production cost function can be further separated by commodity into

\[
s_{itk} + p_{itk}q_{itk} \quad \text{for } k = 1, \ldots, K.
\]

Then, for each production arc and for each commodity the slope of the linear approximation in iteration \( \ell + 1 \) \( (p_{itk}^{(\ell+1)}) \) is such that

\[
s_{itk} + p_{itk}q_{itk}^{(\ell)} = p_{itk}^{(\ell+1)}q_{itk}^{(\ell)}.
\]

Since the initial linear approximation is an underestimator of the cost function, the following should hold:

\[
s_{itk} + p_{itk}v_{it} = p_{itk}^{(0)}v_{it}.
\]
The initial cost coefficients for the production arcs are
\[ \bar{p}_{itk}^{(0)} = \frac{s_{itk}}{v_{it}} + p_{itk}. \]

The cost coefficients are updated as follows:
\[
\bar{p}_{itk}^{(\ell+1)} = \begin{cases} 
\frac{s_{itk}}{q_{itk}} + p_{itk} & \text{if } q_{itk}^{(\ell)} > 0 \\
\bar{p}_{itk}^{(\ell)} & \text{otherwise.}
\end{cases}
\]

For transportation arcs, the linear approximation of the fixed charge cost function is such that
\[ s_{ijt} + \sum_{k=1}^{K} c_{ijtk} x_{ijtk}^{(\ell)} = \sum_{k=1}^{K} p_{itk}^{(\ell+1)} q_{ijtk}^{(\ell)} \quad \text{for } i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T. \]

The fixed charge cost \( s_{ijt} \) is shared among all the commodities using a transportation arc. In this case we use MCDSSP since it allows to distribute the fixed setup cost among all commodities using weights \( w_{ijtk}^{(\ell)} \). The initial cost coefficients for transportation arcs are
\[ \bar{c}_{ijtk}^{(0)} = \frac{s_{ijt}}{v_{ijt}} + c_{ijtk}, \]
and the updating scheme is
\[
\bar{c}_{ijtk}^{(\ell+1)} = \begin{cases} 
\frac{w_{ijtk}^{(\ell)} s_{ijt}}{v_{ijt}} + c_{ijtk} & \text{if } \sum_{k=1}^{K} x_{ijtk}^{(\ell)} > 0 \\
\bar{c}_{ijtk}^{(\ell)} & \text{otherwise.}
\end{cases}
\]

weights \( w_{ijtk}^{(\ell)} \) are calculated as described in Section 4.3.3.

### 4.3.5 Extended Problem Formulation

In this section we provide an extended formulation for the production-distribution problem (PD). MCDSSP provides feasible solutions to the (PD) problem. In order to evaluate the quality of the solutions from MCDSSP, we generate lower bounds using the linear programming relaxation of the extended formulation. Experimental results from previous chapters has shown that linear programming relaxation of the extended
problem formulation gives tighter bounds than the linear programming relaxation of the “original” MILP formulation.

Let us denote by \( q_{ijtrk} \) the production at facility \( i \) in period \( t \) for commodity \( k \) at retailer \( j \) in period \( \tau \). Splitting the production variables \( q_{itk} \) by destination into \( q_{ijtrk} \) allows for the following substitutions:

\[
q_{itk} = \sum_{j=1}^{R} \sum_{\tau=t}^{T} q_{ijtrk} \quad (4.5)
\]

\[
x_{ijtk} = \sum_{s=1}^{t} q_{ijstk} \quad (4.6)
\]

\[
I_{itk} = \sum_{j=1}^{R} \sum_{s=1}^{t} \sum_{\tau=t}^{T} q_{ijstk} - \sum_{s=1}^{t} q_{ijstk} \quad (4.7)
\]

The extended formulation is the following:

\[
\text{minimize} \quad \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} \left[ \sum_{j=1}^{R} \sum_{\tau=t}^{T} c_{ijtrk} q_{ijtrk} + s_{itk} y_{itk} \right] + \sum_{i=1}^{F} \sum_{j=1}^{R} \sum_{t=1}^{T} s_{ijt} y_{ijt}
\]

subject to

\[
\sum_{i=1}^{F} \sum_{t=1}^{T} q_{ijtrk} = b_{jrk} \quad j = 1, \ldots, R; \tau = 1, \ldots, T; k = 1, \ldots, K \quad (4.8)
\]

\[
\sum_{j=1}^{R} \sum_{k=1}^{K} \sum_{\tau=t}^{T} q_{ijtrk} = v_{it} \quad i = 1, \ldots, F; t = 1, \ldots, T \quad (4.9)
\]

\[
\sum_{k=1}^{K} \sum_{t=1}^{T} q_{ijtrk} \leq v_{ijr} y_{ijr} \quad i = 1, \ldots, F; j = 1, \ldots, R; \tau = 1, \ldots, T \quad (4.10)
\]

\[
q_{ijtrk} \leq d_{jrk} y_{itk} \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T; t \leq \tau \leq T; \quad k = 1, \ldots, K \quad (4.11)
\]

\[
q_{ijtrk} \geq 0 \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T; t \leq \tau \leq T; \quad k = 1, \ldots, K \quad (4.12)
\]

\[
y_{itk}, y_{ijt} \in \{0, 1\} \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T; \quad k = 1, \ldots, K \quad (4.13)
\]
The variable unit cost in the arcs of the extended network is equal to
\[ \bar{c}_{ij\tau k} = p_{itk} + c_{ij\tau k} + \sum_{s=t+1}^{\tau} h_{isk}. \]
\( \bar{c}_{ij\tau k} \) consists of unit production cost, unit transportation cost and total unit inventory cost from the time production occurs to the moment the product is shipped to the retailer.

### 4.4 A Lagrangean Decomposition Procedure

In this section we discuss a Lagrangean decomposition procedure to solve the production, inventory and transportation problem introduced in Section 4.2. We apply the Lagrangean decomposition procedure on the extended problem formulation (Ex-PD).

In order to apply Lagrangean decomposition, we first duplicate variables \( q_{ij\tau k} \). We do this by introducing the “copy” variables \( z_{ij\tau k} \) such that

\[ q_{ij\tau k} = z_{ij\tau k} \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T; t \leq \tau \leq T; k = 1, \ldots, K \quad (4.14) \]

The following is an equivalent formulation of (Ex-PD).

\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} \left[ \sum_{j=1}^{R} \sum_{\tau=t}^{T} \bar{c}_{ij\tau k} q_{ij\tau k} + s_{itk} y_{itk} \right] + \sum_{i=1}^{F} \sum_{j=1}^{R} \sum_{t=1}^{T} s_{ijt} y_{ijt} \\
\text{subject to} & \quad (4.8),(4.11),(4.12),(4.13) \text{ and } (4.14) \\
\sum_{i=1}^{F} \sum_{\tau=t}^{T} z_{ij\tau k} &= d_{j\tau k} \quad \tau = 1, \ldots, T; j = 1, \ldots, R; k = 1, \ldots, K \quad (4.15) \\
\sum_{j=1}^{R} \sum_{k=1}^{K} \sum_{\tau=t}^{T} z_{ij\tau k} &\leq v_{it} \quad i = 1, \ldots, F; t = 1, \ldots, T \quad (4.16) \\
\sum_{k=1}^{K} \sum_{t=1}^{T} z_{ij\tau k} &\leq v_{ij\tau} y_{ij\tau} \quad i = 1, \ldots, F; j = 1, \ldots, R; \tau = 1, \ldots, T \quad (4.17) \\
z_{ij\tau k} &\geq 0 \quad i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T; t \leq \tau \leq T; k = 1, \ldots, K \quad (4.18)
\end{align*}
Let $z^{**}$ be the optimal solution to subproblem (SP₂) in iteration $s$

Initialize $UB^s = 0$; $\text{sum}_{itk} = 0$ for $i = 1, \ldots, F; t = 1, \ldots, T; k = 1, \ldots, K$

for $i = 1, \ldots, F; t = 1, \ldots, T; k = 1, \ldots, K$

for $\tau = t, \ldots, T; j = 1, \ldots, R$

if $z^{**}_{ijt\tau k} \geq 0$ then $UB^s = UB^s + \bar{c}_{ijt\tau k}z^{**}_{ijt\tau k}$

$\text{sum}_{itk} = \text{sum}_{itk} + z^{**}_{ijt\tau k}$

if $\text{sum}_{itk} > 0$ then $UB^s = UB^s + s_{itk}$

Initialize $\text{sum}_{ijt} = 0$ for $i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T$

for $i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T$

for $k = 1, \ldots, K; \tau = 1, \ldots, t$

$\text{sum}_{ijt} = \text{sum}_{ijt} + z^{**}_{ijt\tau k}$

if $\text{sum}_{ijt} > 0$ then $UB^s = UB^s + s_{ijt}$

---

Figure 4–2: Upper bound procedure

Relaxing constraints (4.14) decomposes the problem into the following two

subproblems:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{F} \sum_{t=1}^{T} \sum_{k=1}^{K} \sum_{j=1}^{R} \sum_{\tau=t}^{T} \left( \bar{c}_{ijt\tau k} - \lambda_{ijt\tau k} \right) q_{ijt\tau k} + s_{itk} y_{itk} \\
\text{subject to} & \quad (4.8), (4.11), (4.12) \text{ and } (4.13)
\end{align*}
\]

and

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{F} \sum_{j=1}^{R} \sum_{t=1}^{T} \left( \sum_{k=1}^{K} \sum_{\tau=t}^{T} \lambda_{ijt\tau k} z_{ijt\tau k} + s_{ijt} y_{ijt} \right) \\
\text{subject to} & \quad (4.15), (4.16), (4.17), (4.18) \text{ and } (4.13)
\end{align*}
\]

Subproblem (SP₁) can further be decomposed by commodity. This gives $K$
single-commodity, multi-facility, multi-retailer problems with only fixed charge
production costs (SP₁ₖ). The production set-up cost does not allow the problem to
decompose further by retailer. Subproblem (SP₂) is an integer programming problem.
Step 1: Initialize $\lambda$, $\min_{UB}$, $\max_{LB}$, $s$, $u$, $\gamma$, $\epsilon$, $count$.

Step 2: Solve the subproblems $(SP_1)$ and $(SP_2)$. Compute the lower bound:

$$LB^s = \sum_{k=1}^{K} \omega(SP_{1k}(\lambda^s)) + \omega(SP_{2}(\lambda^s))$$

for the current iteration $s$

$count = count + 1$

If $LB^s > \max_{LB}$, then $\max_{LB} = LB^s$

Step 3: Compute an upper bound $UB^s$

If $UB^s < \min_{UB}$, then $\min_{UB} = UB^s$

If $\min_{UB} = \max_{LB}$, then STOP

If $\gamma \leq \epsilon$, then STOP

Step 4: Update the multipliers using equation (4.19)

Step 5: Stop if the number of iterations reach the prespecified limit ($count$)

Otherwise go to Step 2

Figure 4–3: Lagrangean decomposition algorithm

A solution to this subproblem is a feasible solution to (Ex-PD). We use these solutions to obtain upper bounds. Figure 4–2 presents the upper bound procedure.

Both subproblems $(SP_1)$ and $(SP_2)$ do not have the integrality property, therefore the value of the Lagrangean dual (in the case that the subproblems are solved optimally) dominates the optimal value of the Lagrangean dual obtained by relaxing one set of constraints (see Guignard and Kim [52]).

We solve the sub-subproblem $(SP_{1k})$ using the dual algorithm discussed in Section 3.2.2. This algorithm gives high quality lower bounds in reasonable amount of time. The dual algorithm does not guarantee the optimal solution to the subproblem. However, using the dual algorithm to obtain a lower bound, rather than solving the problem to optimality, saves in the computational time. Subproblem $(SP_2)$ is not solved to optimality. We provide a lower bound to these subproblems by solving the corresponding linear programming relaxation.

Subgradient optimization is used to maximize the Lagrangean dual function $z(\lambda)$ ($z(\lambda) = v(LD(x, z, \lambda))$). At each iteration $s$, we calculate the Lagrangean multipliers $\lambda_{ijtrk}$ using the following equation:
\[ \lambda_{ijtrk}^{s+1} = \lambda_{ijtrk}^s + u^s(q_{ijtrk} - z_{ijtrk}) \] (4.19)

for \( i = 1, \ldots, F; j = 1, \ldots, R; t = 1, \ldots, T; t \leq \tau \leq T; k = 1, \ldots, K, \) and

\[ u^s = \frac{\gamma^s (\min_{UB} - \max_{LB})}{\sum_{i=1}^{F} \sum_{j=1}^{R} \sum_{t=1}^{T} \sum_{\tau=t}^{T} \sum_{k=1}^{K} (q_{ijtrk} - z_{ijtrk})^2}. \]

\( \gamma^s \) is a number greater than 0 and less or equal to 2. \( \gamma^s \) is reduced if the lower bound fails to improve after a fixed number of iterations; \( \max_{LB} \) is the best lower bound found up to iteration \( s \) and \( \min_{UB} \) is the best upper bound.

In order to find a subgradient direction at each step of the Lagrangean decomposition procedure using equations 4.19, we need to find a feasible solution to subproblem (SP₁). The dual algorithm provides only a lower bound to (Ex-PD), but does not provide a feasible primal solution. Therefore, we use the primal algorithm (Section 3.2.2) to find a feasible solution to these subproblems.

In our computational experiments, we terminate the algorithm if one of the following happens: (i) the best lower bound is equal to the best upper bound (the optimal solution is found), (ii) the number of iterations reaches a prespecified bound, (iii) the scalar \( \gamma^s \) is less than or equal to \( \epsilon \) (a small number close to zero). Figure 4–3 presents the steps of the Lagrangean decomposition algorithm. In the next section, we test through computational experiments the performance of the Lagrangean decomposition scheme.

### 4.5 Computational Results

In this section we illustrate the performance of the MCDSSP and Lagrangean decomposition algorithm on large-scale instances of the production-distribution problem introduced in Section 4.2. We randomly generated test problems, and compared the running times and solution quality to the general purpose solver CPLEX. The CPLEX runs were stopped whenever a guaranteed error bound of 1%
or less was achieved, allowing for a maximum CPU time of 1,000 seconds (or 5,000 seconds depending on the size of the problem).

For various problem classes characterized by the number of facilities, retailers, periods, and commodities, we randomly generated the retailer’s demands; unit production, inventory and transportation costs; fixed production and transportation setup costs; and production and transportation capacities. The variable transportation costs depend on the length of the route from the facility to the retailer (Section 2.8).

The transportation arcs are subject to bundle capacity constraints. The bundle capacities should be chosen large enough to make the problem feasible, but not so large that they are effectively absent. We have used the following approach to generate "good" bundle capacities: for every time period, a necessary condition for feasibility of the proposed model is

$$\sum_{i=1}^{F} v_{ijt} \geq \sum_{k=1}^{K} b_{jtk}$$ for $j = 1, \ldots, R; \ t = 1, \ldots, T$. (4.20)

Now if all trucks that may serve retailer $j$ in period $t$ have the same capacity $v_{jt}$, (4.20) is equivalent to

$$v_{jt} \geq \frac{1}{F} \sum_{k=1}^{K} b_{jtk}.$$ At the other extreme, if

$$v_{jt} \geq \sum_{k=1}^{K} b_{jtk}$$

the bundle capacities are redundant. We therefore choose

$$v_{jt} = \frac{\delta^*}{F} \sum_{k=1}^{K} b_{jtk}$$

for different values of $\delta^*$. The fact that (4.20) is a necessary condition for feasibility, says that we should have $\delta^* \geq 1$. In the extreme case that $\delta^* = 1$, the total bundle capacities for all transportation arcs to a given retailer are equal to the retailer’s demand, and thus the transportation arcs to retailer $j$ are used at full capacity. In
other words, the retailer does not have a choice. In order to satisfy the demand, full load shipments should be received from all the facilities. Since this implies that all transportation setup variables need to be equal to one, the problem reduces to a linear programming problem. Similarly, for values of $\delta^*$ only slightly larger than 1, few or none of the setup variables will be allowed to have a value of zero, still making the problem relatively easy to solve to optimality. On the other hand, if $\delta^* = F$, the bundle capacities are redundant.

In our test problems we consider not only transportation arcs to be subject to bundle capacities, but production arcs as well. In order to generate bundle capacities for production arcs we use the same procedure as described in Section 3.1.7. The production bundle capacities are calculated as follows:

$$\bar{v} = \frac{\delta}{F} \max_t \sum_{j=1}^R \sum_{\tau=1}^T \sum_{k=1}^K b_{j\tau k}.$$

The scope of our experiments is to see how factors such as the value of set-up costs, number of facilities and retailers, tightness of arc capacities, etc. affected the performance of MCDSSP and Lagrangean decomposition algorithm. We first generated a nominal case problem as follows:

- Production & transportation set-up costs $\sim U[200, 300]$
- Production variable costs $p_{itk} \sim U[5, 15]$
- Holding costs $h_{itk} \sim U[5, 15]$
- Tightness of production arc capacities $\delta = 1.3$
- Tightness of transportation arc capacities $\delta^* = 1.3$
- Demand $b_{jtk} \sim U[50, 200]$
- Number of facilities $F = 5$
- Number of retailers $R = 10$
- Number of periods $T = 5$
- Number of commodities $K = 10$

Varying one or more factors from the nominal case, we generated seven groups of problems. In the first group of problems we increased the number of commodities from 10 to 15, 20, 25, 30 and 35 (problem classes 1 to 6; problem class 1 corresponds
Table 4–1: Problem characteristics

<table>
<thead>
<tr>
<th>Problem</th>
<th>Nodes</th>
<th>Arcs*</th>
<th>Arcs**</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>760</td>
<td>2,950</td>
<td>7,750</td>
</tr>
<tr>
<td>2</td>
<td>1,140</td>
<td>4,425</td>
<td>11,625</td>
</tr>
<tr>
<td>3</td>
<td>1,520</td>
<td>5,900</td>
<td>15,500</td>
</tr>
<tr>
<td>4</td>
<td>1,900</td>
<td>7,375</td>
<td>19,375</td>
</tr>
<tr>
<td>5</td>
<td>2,280</td>
<td>8,850</td>
<td>23,250</td>
</tr>
<tr>
<td>6</td>
<td>2,660</td>
<td>10,325</td>
<td>27,125</td>
</tr>
<tr>
<td>7</td>
<td>1,010</td>
<td>4,200</td>
<td>11,500</td>
</tr>
<tr>
<td>8</td>
<td>1,260</td>
<td>5,450</td>
<td>15,250</td>
</tr>
<tr>
<td>9</td>
<td>1,510</td>
<td>6,700</td>
<td>19,000</td>
</tr>
<tr>
<td>10</td>
<td>1,760</td>
<td>7,950</td>
<td>22,750</td>
</tr>
<tr>
<td>11</td>
<td>2,010</td>
<td>9,200</td>
<td>26,500</td>
</tr>
<tr>
<td>12</td>
<td>2,260</td>
<td>10,450</td>
<td>30,250</td>
</tr>
<tr>
<td>13</td>
<td>1,510</td>
<td>5,950</td>
<td>28,000</td>
</tr>
<tr>
<td>14</td>
<td>2,260</td>
<td>8,950</td>
<td>60,750</td>
</tr>
<tr>
<td>15</td>
<td>3,010</td>
<td>11,950</td>
<td>106,000</td>
</tr>
<tr>
<td>16</td>
<td>3,760</td>
<td>14,950</td>
<td>163,750</td>
</tr>
<tr>
<td>17</td>
<td>4,510</td>
<td>17,950</td>
<td>234,000</td>
</tr>
<tr>
<td>18</td>
<td>5,260</td>
<td>20,950</td>
<td>316,750</td>
</tr>
<tr>
<td>19</td>
<td>810</td>
<td>3,540</td>
<td>9,300</td>
</tr>
<tr>
<td>20</td>
<td>860</td>
<td>4,130</td>
<td>10,850</td>
</tr>
<tr>
<td>21</td>
<td>910</td>
<td>4,720</td>
<td>12,400</td>
</tr>
<tr>
<td>22</td>
<td>960</td>
<td>5,310</td>
<td>13,950</td>
</tr>
<tr>
<td>23</td>
<td>1,010</td>
<td>5,900</td>
<td>15,500</td>
</tr>
<tr>
<td>24–43</td>
<td>760</td>
<td>2,950</td>
<td>7,750</td>
</tr>
</tbody>
</table>

* Arcs of (MC)
** Arcs of (Ex-MC)
to the nominal case). In the second group we increased the number of retailers from 10 (nominal case) to 15, 20, 25, 30, 35 and 40 (problem classes 7 to 12). In the third group of problems we increased the length of time horizon to 10, 15, 20, 25, 30 and 35 (problem classes 13 to 18). In the fourth group we increased the number of facilities to 6, 7, 8, 9 and 10 (problem classes 19 to 23). In the fifth group we changed the level of production set-up costs to $s_{itk} \sim U[200, 900]$, $s_{itk} \sim U[600, 900]$, $s_{itk} \sim U[900, 1500]$, and $s_{itk} \sim U[1200, 1500]$ (problem classes 24 to 27) and in the sixth group of problems, the level of transportation set-up costs is changed to $s_{ijt} \sim U[200, 900]$, $s_{ijt} \sim U[600, 900]$, $s_{ijt} \sim U[900, 1500]$ and $s_{ijt} \sim U[1200, 1500]$ (problem classes 28 to 31).

With the seventh group of problems we want to test the effect of the capacity tightness on the performance of the algorithm. Since we consider capacities and fixed charges, we cannot claim that the tighter the arc capacity, the more difficult the problem becomes. The two extreme cases, arc capacities being too tight or too loose, make the problem easier. This is the reason that we want to analyze the effect of arc capacities on the difficulty of the problem. We changed the value of $\delta$ to 1, 1.1, 1.2, 1.4, 1.5, 1.6 (problem classes 32 to 37) and $\delta^*$ to 1, 1.1, 1.2, 1.4, 1.5, 1.6 (problem classes 38 to 43). In implementing the Lagrangean decomposition algorithm, for all problem instances we set $\gamma = 1.8$. $\gamma$ is reduced by 20% if there is no improvement in the last 5 iterations. We set a limit of 300 iterations for the Lagrangean decomposition algorithm.

Table 4–1 presents the characteristics of the problems generated in this section. Tables 4–2 and 4–3 present the errors and running times of the three variants of MCDSSP. Tables 4–5 and 4–6 present the errors and running times of the Lagrangean decomposition algorithm. The error presented for the heuristics is with respect to the best lower bound found from CPLEX.
Table 4-2: Results of MCDSSP on problem Groups 1, 2, 3 and 4

<table>
<thead>
<tr>
<th>Problem</th>
<th>CPLEX</th>
<th>MCDSSP1</th>
<th>MCDSSP2</th>
<th>MCDSSP3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error (%)</td>
<td>Time (sec)</td>
<td>Error (%)</td>
<td>Time (sec)</td>
</tr>
<tr>
<td>1</td>
<td>0.01</td>
<td>540.01</td>
<td>0.87</td>
<td>1.05</td>
</tr>
<tr>
<td>2</td>
<td>0.01</td>
<td>137.93</td>
<td>0.77</td>
<td>4.03</td>
</tr>
<tr>
<td>3</td>
<td>0.01</td>
<td>85.12</td>
<td>0.70</td>
<td>4.05</td>
</tr>
<tr>
<td>4</td>
<td>0.01</td>
<td>36.98</td>
<td>0.68</td>
<td>8.08</td>
</tr>
<tr>
<td>5</td>
<td>0.01</td>
<td>35.45</td>
<td>0.65</td>
<td>5.88</td>
</tr>
<tr>
<td>6</td>
<td>0.01</td>
<td>41.27</td>
<td>0.61</td>
<td>22.63</td>
</tr>
<tr>
<td>7</td>
<td>0.08</td>
<td>997.66</td>
<td>0.99</td>
<td>1.05</td>
</tr>
<tr>
<td>8</td>
<td>0.10</td>
<td>1,000.00</td>
<td>0.82</td>
<td>3.58</td>
</tr>
<tr>
<td>9</td>
<td>0.11</td>
<td>1,000.00</td>
<td>0.83</td>
<td>3.02</td>
</tr>
<tr>
<td>10</td>
<td>0.11</td>
<td>1,000.00</td>
<td>0.75</td>
<td>3.85</td>
</tr>
<tr>
<td>11</td>
<td>0.10</td>
<td>1,000.00</td>
<td>0.75</td>
<td>5.26</td>
</tr>
<tr>
<td>12</td>
<td>0.12</td>
<td>1,000.00</td>
<td>0.71</td>
<td>5.15</td>
</tr>
<tr>
<td>13</td>
<td>0.29</td>
<td>1,000.00</td>
<td>1.37</td>
<td>2.09</td>
</tr>
<tr>
<td>14</td>
<td>N/A</td>
<td>5,000.00</td>
<td>1.37</td>
<td>4.05</td>
</tr>
<tr>
<td>15</td>
<td>N/A</td>
<td>5,000.00</td>
<td>1.48</td>
<td>6.09</td>
</tr>
<tr>
<td>16</td>
<td>N/A</td>
<td>5,000.00</td>
<td>1.47</td>
<td>7.48</td>
</tr>
<tr>
<td>17</td>
<td>N/A</td>
<td>5,000.00</td>
<td>1.65</td>
<td>19.30</td>
</tr>
<tr>
<td>18</td>
<td>N/A</td>
<td>5,000.00</td>
<td>2.54</td>
<td>12.59</td>
</tr>
<tr>
<td>19</td>
<td>0.23</td>
<td>5,000.00</td>
<td>1.82</td>
<td>0.86</td>
</tr>
<tr>
<td>20</td>
<td>0.67</td>
<td>5,000.00</td>
<td>2.36</td>
<td>1.30</td>
</tr>
<tr>
<td>21</td>
<td>1.39</td>
<td>5,000.00</td>
<td>2.83</td>
<td>3.30</td>
</tr>
<tr>
<td>22</td>
<td>0.72</td>
<td>5,000.00</td>
<td>3.45</td>
<td>2.78</td>
</tr>
<tr>
<td>23</td>
<td>0.74</td>
<td>5,000.00</td>
<td>3.81</td>
<td>4.39</td>
</tr>
</tbody>
</table>
\[
\text{Error}(\%) = \frac{\text{MCDSSP} - \text{CPLEX Lower Bound}}{\text{CPLEX Lower Bound}} \times 100.
\]

In all cases, the results are averaged over 20 generated instances.

Tables 4–2 and 4–3 indicate that MCDSSP3 consistently gave the best results in terms of time and solution quality. This is due to the fact that the updating scheme of MCDSSP3 distributes the fixed charge cost among the commodities using the same arc, by considering more information than MCDSSP1 and MCDSSP2. MCDSSP3 assigns to every commodity that uses the arc a part of the setup costs, considering the unit cost as well as the amount shipped. Different from MCDSSP3, MCDSSP1 equally distributes the setup costs among all the commodities flowing on the arc, and MCDSSP2 assigns the setup costs to the commodities considering only the amount shipped.

<table>
<thead>
<tr>
<th>Problem</th>
<th>CPLEX</th>
<th>MCDSSP1</th>
<th>MCDSSP2</th>
<th>MCDSSP3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error (%)</td>
<td>Time (sec)</td>
<td>Error (%)</td>
<td>Time (sec)</td>
</tr>
<tr>
<td>24</td>
<td>0.10</td>
<td>958.95</td>
<td>1.28</td>
<td>0.58</td>
</tr>
<tr>
<td>25</td>
<td>0.14</td>
<td>1,000.00</td>
<td>3.12</td>
<td>0.57</td>
</tr>
<tr>
<td>26</td>
<td>0.30</td>
<td>1,000.00</td>
<td>4.38</td>
<td>0.60</td>
</tr>
<tr>
<td>27</td>
<td>0.49</td>
<td>1,000.00</td>
<td>7.31</td>
<td>0.61</td>
</tr>
<tr>
<td>28</td>
<td>0.10</td>
<td>959.18</td>
<td>1.28</td>
<td>0.57</td>
</tr>
<tr>
<td>29</td>
<td>0.26</td>
<td>1,000.00</td>
<td>2.80</td>
<td>0.62</td>
</tr>
<tr>
<td>30</td>
<td>0.36</td>
<td>1,000.00</td>
<td>3.36</td>
<td>0.65</td>
</tr>
<tr>
<td>31</td>
<td>0.57</td>
<td>1,000.00</td>
<td>5.51</td>
<td>0.72</td>
</tr>
<tr>
<td>32</td>
<td>0.26</td>
<td>1,000.00</td>
<td>1.33</td>
<td>1.40</td>
</tr>
<tr>
<td>33</td>
<td>0.18</td>
<td>1,000.00</td>
<td>1.27</td>
<td>0.78</td>
</tr>
<tr>
<td>34</td>
<td>0.12</td>
<td>962.10</td>
<td>1.30</td>
<td>0.66</td>
</tr>
<tr>
<td>35</td>
<td>0.09</td>
<td>944.15</td>
<td>1.30</td>
<td>0.54</td>
</tr>
<tr>
<td>36</td>
<td>0.10</td>
<td>946.46</td>
<td>1.30</td>
<td>0.52</td>
</tr>
<tr>
<td>37</td>
<td>0.10</td>
<td>949.81</td>
<td>1.29</td>
<td>0.51</td>
</tr>
<tr>
<td>38</td>
<td>0.01</td>
<td>506.53</td>
<td>1.17</td>
<td>0.66</td>
</tr>
<tr>
<td>39</td>
<td>0.03</td>
<td>812.30</td>
<td>1.10</td>
<td>0.56</td>
</tr>
<tr>
<td>40</td>
<td>0.12</td>
<td>965.23</td>
<td>1.08</td>
<td>0.58</td>
</tr>
<tr>
<td>41</td>
<td>0.05</td>
<td>859.45</td>
<td>1.25</td>
<td>0.54</td>
</tr>
<tr>
<td>42</td>
<td>0.03</td>
<td>693.57</td>
<td>1.17</td>
<td>0.54</td>
</tr>
<tr>
<td>43</td>
<td>0.02</td>
<td>537.20</td>
<td>1.09</td>
<td>0.52</td>
</tr>
</tbody>
</table>
Table 4–4 presents the quality of the lower bounds as compared to CPLEX solutions. The error presented is with respect to the best upper bound found by CPLEX.

\[
\text{Error(\%)} = \frac{\text{CPLEX Upper Bound} - \text{LP Lower Bound}}{\text{CPLEX Upper Bound}} \times 100.
\]

Several things may be noted from Tables 4–2 to 4–6. Increasing the number of commodities (problem group one) made the problems easier. The quality of the solutions from MCDSSP and Lagrangean decomposition got better as the number of commodities increased. The running times of MCDSSP for these problems,

<table>
<thead>
<tr>
<th>Problem</th>
<th>Error (%)</th>
<th>Time (sec)</th>
<th>Problem</th>
<th>Error (%)</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.82</td>
<td>1.53</td>
<td>24</td>
<td>1.98</td>
<td>1.62</td>
</tr>
<tr>
<td>2</td>
<td>1.40</td>
<td>2.69</td>
<td>25</td>
<td>2.16</td>
<td>1.83</td>
</tr>
<tr>
<td>3</td>
<td>1.11</td>
<td>3.56</td>
<td>26</td>
<td>2.43</td>
<td>2.19</td>
</tr>
<tr>
<td>4</td>
<td>0.90</td>
<td>4.53</td>
<td>27</td>
<td>2.57</td>
<td>2.23</td>
</tr>
<tr>
<td>5</td>
<td>0.75</td>
<td>5.67</td>
<td>28</td>
<td>2.60</td>
<td>1.61</td>
</tr>
<tr>
<td>6</td>
<td>0.65</td>
<td>6.86</td>
<td>29</td>
<td>2.85</td>
<td>1.56</td>
</tr>
<tr>
<td>7</td>
<td>1.70</td>
<td>2.41</td>
<td>30</td>
<td>2.95</td>
<td>1.62</td>
</tr>
<tr>
<td>8</td>
<td>1.70</td>
<td>3.48</td>
<td>31</td>
<td>3.04</td>
<td>1.57</td>
</tr>
<tr>
<td>9</td>
<td>1.63</td>
<td>5.30</td>
<td>32</td>
<td>2.02</td>
<td>2.92</td>
</tr>
<tr>
<td>10</td>
<td>1.70</td>
<td>5.95</td>
<td>33</td>
<td>1.97</td>
<td>1.97</td>
</tr>
<tr>
<td>11</td>
<td>1.56</td>
<td>7.62</td>
<td>34</td>
<td>1.85</td>
<td>1.68</td>
</tr>
<tr>
<td>12</td>
<td>1.58</td>
<td>9.52</td>
<td>35</td>
<td>1.81</td>
<td>1.52</td>
</tr>
<tr>
<td>13</td>
<td>1.86</td>
<td>10.11</td>
<td>36</td>
<td>1.81</td>
<td>1.50</td>
</tr>
<tr>
<td>14</td>
<td>2.84</td>
<td>32.1</td>
<td>37</td>
<td>1.82</td>
<td>1.40</td>
</tr>
<tr>
<td>15</td>
<td>2.85</td>
<td>70.77</td>
<td>38</td>
<td>0.49</td>
<td>2.69</td>
</tr>
<tr>
<td>16</td>
<td>2.84</td>
<td>144.41</td>
<td>39</td>
<td>1.14</td>
<td>1.81</td>
</tr>
<tr>
<td>17</td>
<td>2.91</td>
<td>301.96</td>
<td>40</td>
<td>1.77</td>
<td>1.60</td>
</tr>
<tr>
<td>18</td>
<td>2.97</td>
<td>608.08</td>
<td>41</td>
<td>2.09</td>
<td>1.56</td>
</tr>
<tr>
<td>19</td>
<td>2.00</td>
<td>2.23</td>
<td>42</td>
<td>2.37</td>
<td>1.53</td>
</tr>
<tr>
<td>20</td>
<td>2.68</td>
<td>3.13</td>
<td>43</td>
<td>2.65</td>
<td>1.52</td>
</tr>
<tr>
<td>21</td>
<td>3.72</td>
<td>4.29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>3.02</td>
<td>5.89</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>3.04</td>
<td>7.99</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

however, increased, since as the number of commodities increases, the size of the linear programs to be solved in every iteration of MCDSSP increases. The running
times of Lagrangean decomposition increased as well, since by increasing the number of commodities we are increasing the number of subproblems \((SP_{1k})\) that are solved in every iteration. These results are consistent with our findings in Section 3.1.7 and other studies on fixed charge multi-commodity network flow problems (Wu and Golbasi [106], Ekşioğlu et al. [35]).

The size of the problems increases with the number of retailers (the second group of problems). This explains the increase in the running times of CPLEX, MCDSSP and Lagrangean decomposition. Everything else kept the same, more retailers imply higher demand, higher production quantities, small ratios of total fixed charge to total variable costs. These ratios affect the performance of MCDSSP and Lagrangean decomposition. In problems with small ratios of total fixed charge to total variable costs, the linear programming relaxation of the fixed charge cost functions gives better approximations. Recall that, in the Lagrangean decomposition algorithm, subproblem \((SP_2)\) is not solved to optimality. We provide lower bounds to \((SP_2)\) by solving its linear programming relaxation.

For problem classes 14 to 18, CPLEX wasn’t able to find a feasible solution within 5,000 CPU seconds. The solutions from the three MCDSSP schemes were within 2.54% of optimality, and the solutions from the Lagrangean decomposition were within 1.64% of optimality.

Increasing the number of facilities while keeping everything else the same (problem classes 19 to 23) made the problems difficult. Recall that we assign to each facility equal production capacity. The assignment is such that the total production capacity up to period \(t\) is equal to \(\delta\) times total demand up to period \(t\). Increasing the number of facilities (while keeping total demand fixed), decreases the bundle capacity assigned to each facility. This is equivalent to decreasing \(\delta\). So far, we have analyzed the effect of a single factor at a time on the performance of the heuristic. However, it seems that when increasing the number of facilities, we are changing two parameters, since we are also decreasing the bundle capacities in each facility.
<table>
<thead>
<tr>
<th>Problem</th>
<th>CPLEX</th>
<th>Lag.Decomp.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error (%), Time (sec)</td>
<td>Error (%), Time (sec)</td>
</tr>
<tr>
<td>1</td>
<td>0.01, 540.01</td>
<td>0.73, 47.47</td>
</tr>
<tr>
<td>2</td>
<td>0.01, 137.93</td>
<td>0.62, 65.37</td>
</tr>
<tr>
<td>3</td>
<td>0.01, 85.12</td>
<td>0.57, 95.39</td>
</tr>
<tr>
<td>4</td>
<td>0.01, 36.98</td>
<td>0.54, 102.66</td>
</tr>
<tr>
<td>5</td>
<td>0.01, 35.45</td>
<td>0.52, 130.33</td>
</tr>
<tr>
<td>6</td>
<td>0.01, 41.27</td>
<td>0.49, 155.26</td>
</tr>
<tr>
<td>7</td>
<td>0.08, 997.66</td>
<td>0.95, 53.37</td>
</tr>
<tr>
<td>8</td>
<td>0.10, 1,000.00</td>
<td>0.85, 87.23</td>
</tr>
<tr>
<td>9</td>
<td>0.11, 1,000.00</td>
<td>0.92, 109.29</td>
</tr>
<tr>
<td>10</td>
<td>0.11, 1,000.00</td>
<td>0.84, 142.89</td>
</tr>
<tr>
<td>11</td>
<td>0.10, 1,000.00</td>
<td>0.88, 180.61</td>
</tr>
<tr>
<td>12</td>
<td>0.12, 1,000.00</td>
<td>0.83, 224.17</td>
</tr>
<tr>
<td>13</td>
<td>0.29, 1,000.00</td>
<td>1.17, 467.06</td>
</tr>
<tr>
<td>14</td>
<td>N/A, 5,000.00</td>
<td>1.18, 1,093.59</td>
</tr>
<tr>
<td>15</td>
<td>N/A, 5,000.00</td>
<td>1.28, 2,318.55</td>
</tr>
<tr>
<td>16</td>
<td>N/A, 5,000.00</td>
<td>1.24, 4,943.88</td>
</tr>
<tr>
<td>17</td>
<td>N/A, 5,000.00</td>
<td>1.32, 7,676.56</td>
</tr>
<tr>
<td>18</td>
<td>N/A, 5,000.00</td>
<td>1.64, 10,963.00</td>
</tr>
<tr>
<td>19</td>
<td>0.23, 5,000.00</td>
<td>1.76, 57.31</td>
</tr>
<tr>
<td>20</td>
<td>0.67, 5,000.00</td>
<td>2.13, 316.61</td>
</tr>
<tr>
<td>21</td>
<td>1.39, 5,000.00</td>
<td>2.56, 88.50</td>
</tr>
<tr>
<td>22</td>
<td>0.72, 5,000.00</td>
<td>3.25, 114.22</td>
</tr>
<tr>
<td>23</td>
<td>0.74, 5,000.00</td>
<td>4.10, 413.89</td>
</tr>
</tbody>
</table>
An alternative approach is to add new facilities that have the same production capacity as the ones that already exist. If we do so, the total available production capacity will increase. This is equivalent to increasing δ. Experimental results (problem group seven) show that increasing δ makes problems easier. In our test problems we decided to go with the first approach, of adding new facilities, and then update the capacities of all facilities such that total available capacity up to time period \( t \) is equal to \( \delta \) multiplied by total demand up to time period \( t \).

Furthermore, MCDSSP, Lagrangean decomposition and CPLEX benefited from small fixed costs (problem groups five and six). Finally, the tightness of the bundle capacities for production arcs (δ) and transportation arcs (δ*) affected the performance of CPLEX and of the heuristics (problem group seven). When δ (δ*) increased, and thus the capacity constraints left more room for choice, the problem became more difficult quite rapidly. Bundle capacities for the production arcs showed to affect more the quality of the solutions from CPLEX and the heuristics, than the transportation bundle capacities. Results showed higher error bounds for problem classes 32 to 37 as compared to problem classes 38 to 43. The results of Table 4–4 indicate that the quality of the lower bounds from the linear programming relaxation of extended formulation were at most 3.72% from optimal and the running times were quite small.

The major observation from the tables is that CPLEX was generally able to find a better solution than MCDSSP and Lagrangean decomposition, but at the expense of a large amount of CPU time. Given the operational nature of our problem, it may not be feasible to spend the time taken by CPLEX to solve the problem. For the larger problems (problem classes 14 to 18), CPLEX was not able to find a feasible solution within 5,000 CPU seconds, whereas MCDSSP and Lagrangean decomposition, by their nature, were able to find a feasible solution for all problem instances.

We conclude that MCDSSP can be a very useful tool, given the speed at which reasonable feasible solutions are found. The error bounds of the solution obtained
Table 4–6: Results from Lagrangean decomposition (Problem Classes 24 to 43)

<table>
<thead>
<tr>
<th>Problem</th>
<th>CPLEX</th>
<th>Lag.Decomp.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error (%)</td>
<td>Time (sec)</td>
</tr>
<tr>
<td>24</td>
<td>0.14</td>
<td>1,000.00</td>
</tr>
<tr>
<td>25</td>
<td>0.30</td>
<td>1,000.00</td>
</tr>
<tr>
<td>26</td>
<td>0.49</td>
<td>1,000.00</td>
</tr>
<tr>
<td>27</td>
<td>0.63</td>
<td>1,000.00</td>
</tr>
<tr>
<td>28</td>
<td>0.26</td>
<td>1,000.00</td>
</tr>
<tr>
<td>29</td>
<td>0.36</td>
<td>1,000.00</td>
</tr>
<tr>
<td>30</td>
<td>0.57</td>
<td>1,000.00</td>
</tr>
<tr>
<td>31</td>
<td>0.66</td>
<td>1,000.00</td>
</tr>
<tr>
<td>32</td>
<td>0.26</td>
<td>1,000.00</td>
</tr>
<tr>
<td>33</td>
<td>0.18</td>
<td>1,000.00</td>
</tr>
<tr>
<td>34</td>
<td>0.12</td>
<td>962.10</td>
</tr>
<tr>
<td>35</td>
<td>0.09</td>
<td>944.15</td>
</tr>
<tr>
<td>36</td>
<td>0.10</td>
<td>946.46</td>
</tr>
<tr>
<td>37</td>
<td>0.10</td>
<td>949.81</td>
</tr>
<tr>
<td>38</td>
<td>0.01</td>
<td>506.53</td>
</tr>
<tr>
<td>39</td>
<td>0.03</td>
<td>812.30</td>
</tr>
<tr>
<td>40</td>
<td>0.12</td>
<td>965.23</td>
</tr>
<tr>
<td>41</td>
<td>0.05</td>
<td>859.45</td>
</tr>
<tr>
<td>42</td>
<td>0.03</td>
<td>693.57</td>
</tr>
<tr>
<td>43</td>
<td>0.02</td>
<td>537.20</td>
</tr>
</tbody>
</table>
by MCDSSP were at most as high as 5% for the hardest problems. The errors from MCDSSP and Lagrangean decomposition were comparable.

4.6 Conclusions

In this chapter we studied a class of production-distribution problems arising in supply chains. Our problems consider fixed-charge production and transportation costs and joint production and transportation capacities. We first proposed a heuristic called Dynamic Slope Scaling Procedure (MCDSSP) to solve these problems. This procedure can be used to solve any multi-commodity network flow problem with fixed charge cost functions. We provided three alternative implementations of the MCDSSP. We identified one of the three alternative implementations of our heuristic that seems to consistently provide the best solution quality and computation time.

We also discussed a Lagrangean decomposition based algorithm. The Lagrangean decomposition algorithm decomposed the problem into two subproblems \((SP_1)\) and \((SP_2)\). Subproblem \((SP_1)\) was further decomposed by commodity into \(K\) single-commodity problems \((SP_{1k})\). The problems \((SP_{1k})\) are similar to the multi-facility, multi-retailer lot-sizing problem we discussed in Section 3.2, therefore we use the primal-dual algorithm to solve them. We provide lower bounds to subproblem \((SP_2)\) by solving the corresponding linear programming relaxation.

Comparing our heuristics to CPLEX, we conclude that CPLEX was usually able to find a better solution than the heuristics, at the expense of much more computation time. Given the operational nature of our problem, and the limited availability of time available to solve the problem in these situations, our heuristics may be an attractive alternative.

It is well-known that finding a good lower bound for network flow problems with fixed-charge costs is difficult, due to the fact that the linear programming relaxation of the traditional MILP formulation is not very tight. However, the lower bounds generated using the linear programming relaxation of the extended formulation gave lower bounds that are within 3.72% of optimality.
In this dissertation we study a class of supply chain optimization problems. Several factors motivated this research. First, increased competition and market responsiveness have made supply chains subject to more inter-related dynamics than in the past. As a result companies frequently need to reconsider their distribution networks. Secondly, because of the wide variety of issues involved in managing a supply chain, most of the optimization models have concentrated on specific areas of research such as transportation or inventory, without being able to take a global view of all the processes involved. Finally, the recent developments in computing and solution algorithms have made it possible to investigate richer models than in the past.

In particular, the supply chain optimization problem we study considers a set of facilities and a set of retailers. Retailers face positive demands for a number of commodities. Facilities produce and store the products until retailers demands occur. Production at a facility is constrained and truck capacity limits the total amount of products that can be shipped to a retailer in a period. We do not allow for transportation between facilities or between retailers. The goal is to find the most efficient way (e.g., the cheapest way) to satisfy this demand. Production and transportation costs are modelled using fixed charge cost functions. The decisions to be made are (i) the selection of a production facility and quantities, (ii) the assignment of retailers to facilities and (iii) the location and size of inventories.

For a better understanding, we have followed a contractive approach, where the supply chain models we study have been enriched gradually. First, we analyze the single commodity and multi-facility lot sizing problem. This is an extension of the
economic lot-sizing problem. We add a new dimension to the classical problem, the
certainty selection decision. We consider transportation costs together with production
and inventory cost, and their impacts on lot-sizing decisions. Second, we study a
class of multi-commodity and multi-retailer lot-sizing problems. We evaluate the
performance of the supply chain when multiple products have to be produced and
distributed. In this model only the production capacity has been restricted. The
third model we propose considers a dynamic supply chain consisting of a number of
facilities and retailers. Finally, we discuss a class of production-distribution problems
that considers multiple commodities, fixed charge transportation costs, as well as
production and transportation capacities.

The main focus of this dissertation is to develop solution procedures to solve
these optimization models. Conclusions about their performance are drawn by testing
the algorithms on a wide collection of problem instances. For the multi-retailer lot-
sizing problem we propose a different formulation that we refer to as the extended
problem formulation. The linear programming relaxation of the extended formulation
gives tighter lower bounds as compared to the linear programming relaxation of
the “original” problem formulation. We develop a primal-dual algorithm, a cutting
plane algorithm, and a dynamic programming based algorithm for this problem. We
solve the capacitated, multi-commodity version of the problem using a Lagrangean
decomposition algorithm. The subproblems from the decomposition are solved using
a primal-dual algorithm. A primal-dual algorithm is used to solve the multi-retailer,
multi-facility lot-sizing problem as well.

In Chapter 4 we discuss a slope scaling procedure to solve the production-
distribution problem. This procedure that we call the multi-commodity dynamic slope
scaling procedure (MCDSSP) can be used to solve any multi-commodity network flow
problem with fixed charge cost functions. The motivation behind this heuristics is the
fact that a concave cost function (such as a fixed charge function), when minimized
over a convex set, gives an extreme point solution. The same holds true for linear
programs. Thus, MCDSSP iteratively approximates the fixed charge cost function by a linear function and solves the corresponding linear program in search of a linear program that would be minimized on the same vertex of the feasible region as the fixed charge function.
APPENDIX

Proof of Theorem 2.6.2: In order to prove the theorem we have to show that there exist precisely $\text{dim}(\text{co}(\Phi))$ affinely independent solutions from the convex hull of the feasible region that satisfy the $(l,S)$ inequality (2.25) as equality.

Given a $(l,S)$ such that $l < T$, $1 \in S$ and $L \setminus S \neq \emptyset$. Let $k = \arg\min_j \{j \in L \setminus S\} > 1$. The first set of feasible solutions that we discuss is the following: Consider all feasible solutions $(q, y, x) \in \Phi$ such that $q_{it} = y_{it} = 0$, for $t = k, \ldots, l$ and $i = 1, \ldots, F$. These solutions satisfy the valid inequality $(l, S)$ at equality.

In this particular case, our problem can be decomposed into two multi-facility lot sizing problems, one consisting of periods $1, \ldots, k - 1$, with $b'_t = b_t$, for $t = 1, \ldots, k - 2$, and $b'_{k-1} = b_{k-1,t}$, and the other consisting of periods $l + 1, \ldots, T$. The first sub-problem has $3(k - 1)F - F - (k - 1) - 1$ linearly independent solutions $(q^A_p, y^A_p, x^A_p) \in R^{3(k-1)F}$. The second sub-problem has $3(T - l)F - (T - l) - 1$ linearly independent solutions $(q^B_q, y^B_q, x^B_q) \in R^{3(T-l)F}$.

Combining these vectors and inserting $q_{it} = y_{it} = 0$, for $t = k, \ldots, l$, and $i = 1, \ldots, F$, gives

$$(3(k - 1)F - F - (k - 1)) + (3(T - l)F - F - (T - l)) - 1$$

$$= (3FT - F - T - 1) - (3F - 1)(l - k + 1) - F$$

affinely independent solutions of the form $(q^1_A, y^1_A, x^1_A, 0, 0, 0, q^q_B, y^q_B, x^q_B)$ and $(q^p_A, y^p_A, x^p_A, 0, 0, q^1_B, y^1_B, x^1_A)$. These solutions have $I_{il} = 0$ and satisfy the valid inequality $(l, S)$ as equality.

Now we will present a few more affinely independent solutions. By definition, $n+1$ solutions $\chi_0, \ldots, \chi_n$ are affine independent if directions $\chi_1 - \chi_0, \chi_2 - \chi_0, \ldots, \chi_n - \chi_0$
are linearly independent. In the rest of our discussion we first present a set of solutions that satisfy the \((l, S)\) inequality at equality, then we will generate \((3F - 1)(k - l + 1) + F\) directions and show that they are affinely independent. In generating the directions we refer to the solution \((\chi_0)\) that is subtracted from a given solution \((\chi_i\) for \(i = 1, \ldots, n)\) with \(d\).

Fix \(t' \in \{k, \ldots, l\}\), and \(i \in \{1, \ldots, F\}\).

**Case 1:** Let \(t' \in L \setminus S\).

Solutions 2

\[
q_{it} = b_t; y_{it} = 1 \quad \text{for } t = 1, \ldots, k-2
\]
\[
q_{ik-1} = b_{k-1}; y_{ik-1} = 1
\]
\[
q_{it'} = b_{t'}; y_{it'} = 1
\]
\[
x_{it} = b_t
\]

Solutions 3

\[
q_{it} = b_t; y_{it} = 1 \quad \text{for } t = 1, \ldots, k-2;
\]
\[
q_{ik-1} = b_{k-1}; y_{ik-1} = 1
\]
\[
q_{it'} = b_{t'}; y_{it'} = 1
\]
\[
I_{it} = b_{t+1}
\]
\[
x_{it} = b_t
\]

**Case 2:** Let \(t' = \{k, \ldots, l\} \cap S\). Take solutions 2 the same way as in Case 1.

Solutions 3

\[
q_{it} = b_t; y_{it} = 1 \quad \text{for } t = 1, \ldots, k-2
\]
\[
q_{ik-1} = b_{k-1}; y_{ik-1} = 1
\]
\[
y_{it'} = 1
\]
\[
q_{it} = b_t; y_{it} = 1 \quad \text{for } t = l+1, \ldots, T
\]
\[
x_{it} = b_t \quad \text{for } t = 1, \ldots, T
\]

The number of solutions found in this step is: \(2F(l - k + 1)\).

The fourth set of solutions that satisfies the \((l, S)\) at equality is the following:

For each \(t' = k, \ldots, l\), and \(i = 1, \ldots, F - 1\):
Solutions 4
\[ q_{it} = b_t; y_{it} = 1; x_{it} = b_t \quad \text{for } t = 1, \ldots, k-2 \]
\[ q_{i,k-1} = b_{k-1}; y_{i,k-1} = 1 \]
\[ x_{it} = b_t \quad \text{for } t = k-1, \ldots, t'-1 \]
\[ q_{F'it} = b_{F't}; y_{F'it} = 1 \]
\[ x_{F't} = b_t \quad \text{for } t = t', \ldots, l \]
\[ q_{it} = b_t; y_{it} = 1; x_{it} = b_t \quad \text{for } t = l+1, \ldots, T \]

In this step we found \((F - 1)(l - k + 1)\) solutions that satisfy \((l, S)\) at equality.

A final set of solutions is the following: let \(t' = k\) and \(i = 1, \ldots, F\), and choose the following solutions:

Solutions 5
\[ q_{F1} = b_{F,k-1}; y_{F1} = 1; x_{Ft} = b_t \quad \text{for } t = 1, \ldots, k-1 \]
\[ q_{ik} = b_{kt}; y_{ik} = 1 \]
\[ x_{it} = b_t \quad \text{for } t = k, \ldots, l \]
\[ q_{Ft} = b_t; y_{Ft} = 1; x_{Ft} = b_t \quad \text{for } t = l+1, \ldots, T \]

This step will give \(F\) new solutions that satisfy the valid inequality at equality.

Now, we will construct directions (that we will show to be linearly independent) by subtracting \(d\) from the set of solutions presented above. For Solutions 2 and 3, let \(d_{it}\) be the following solutions for \(i = 1, \ldots, F\).

Directions \(d_{1i}\):
\[ q_{i1} = b_{1i}; y_{i1} = 1 \]
\[ q_{it} = b_t; y_{it} = 1 \quad \text{for } t = l+1, \ldots, T \]
\[ x_{it} = b_t \quad \text{for } t = 1, \ldots, T \]

Then, for \(t' \in \{k, \ldots, l\}\), and \(i \in \{1, \ldots, F\}\):

Case 1: Let \(t' \in L \setminus S\).

Directions 2
\[ q_{i1} = -b_{2i}; q_{it} = b_t; y_{it} = 1 \quad \text{for } t = 2, \ldots, k-2 \]
\[ q_{i,k-1} = b_{k-1}; y_{i,k-1} = 1 \]
\[ q_{i,t'} = b_{t'}; y_{i,t'} = 1 \]

Directions 3
\[ q_{i1} = -b_{2i}; q_{it} = b_t; y_{it} = 1 \quad \text{for } t = 2, \ldots, k-2 \]
\[ q_{i,k-1} = b_{k-1,t-1}; y_{i,k-1} = 1 \]
\[ q_{i,t'} = b_{t',t+1}; y_{i,t'} = 1 \]

**Case 2:** Let \( t' = \{k, \ldots, l\} \cap S \).

**Directions 2**
\[ q_{i1} = -b_{2i}; q_{it} = b_{t}; y_{it} = 1 \quad \text{for } t = 2, \ldots, k-2 \]
\[ q_{i,k-1} = b_{k-1,t-1}; y_{i,k-1} = 1 \]
\[ q_{i,t'} = b_{t,t'}; y_{i,t'} = 1 \]

**Directions 3**
\[ q_{i1} = -b_{2i}; q_{it} = b_{t}; y_{it} = 1 \quad \text{for } t = 2, \ldots, k-2 \]
\[ q_{i,k-1} = b_{k-1,t}; y_{i,k-1} = 1 \]
\[ y_{i,t'} = 1 \]

We generate the fourth set of directions subtracting \( d_{t'} F \) for \( t' = k, \ldots, l \) from the fourth set of solutions.

**Directions \( d_{t'} F \):**
\[ q_{Ft} = b_{t}; y_{Ft} = 1; x_{Ft} = b_{t} \quad \text{for } t = 1, \ldots, k-2 \]
\[ q_{FK-1} = b_{k-1,t-1}; y_{FK-1} = 1 \]
\[ x_{Ft} = b_{t} \quad \text{for } t = k-1, \ldots, l \]
\[ q_{Ft'} = b_{t,t'}; y_{Ft'} = 1; \]
\[ q_{Ft} = b_{t}; y_{Ft} = 1; x_{Ft} = b_{t} \quad \text{for } t = l+1, \ldots, T \]

**Directions 4**
\[ q_{Ft} = -b_{t}; y_{Ft} = -1; x_{Ft} = -b_{t} \quad \text{for } t = 1, \ldots, k-2 \]
\[ q_{it} = b_{t}; y_{it} = 1; x_{it} = b_{t} \quad \text{for } t = 1, \ldots, k-2 \]
\[ q_{FK-1} = -b_{k-1,t-1}; y_{FK-1} = -1; x_{FK-1} = -b_{k-1} \]
\[ q_{ik-1} = b_{k-1,t-1}; y_{ik-1} = 1; x_{ik-1} = b_{k-1} \]
\[ x_{Ft} = -b_{t} \quad \text{for } t = k-1, \ldots, t'-1 \]
\[ x_{it} = b_{t} \quad \text{for } t = k-1, \ldots, t'-1 \]
\[ q_{Ft} = -b_{t}; y_{Ft} = -1; x_{Ft} = -b_{t} \quad \text{for } t = l+1, \ldots, T \]
\[ q_{it} = b_{t}; y_{it} = 1; x_{it} = b_{t} \quad \text{for } t = l+1, \ldots, T \]

The final set of directions is the following: for \( t' = k \) and \( i = 1, \ldots, F - 1 \) (consider \( d \) to be defined as the direction we gave for the fourth group of solutions and fix \( t' = k \) and \( i = F \)):

**Directions 5**
\[ q_{FK} = -b_{k,t}; y_{FK} = -1 \]
$x_{Ft} = -b_t$ for $t = k, \ldots, l$
$q_{ik} = b_{kl}; y_{ik} = 1$
$x_{it} = b_t$ for $t = k, \ldots, l$

Next we need to show that all the directions presented so far are linearly independent. First, within each group, the directions are linearly independent.

Consider Directions 2 and 3, $y_{it'}$ is different from zero in exactly one direction.

Considering Directions 4, for $t' = k, \ldots, l$ we set $x_{it'} = b_t$ for $t = k - 1, \ldots, t' - 1$. These variables form a lower triangular matrix (all elements above the diagonal are zero). Finally, in Directions 5, $y_{ik}$ for $i = 1, \ldots, F - 1$ is different than zero in exactly one direction.

The last step is to show that the individual directions from the five sets of directions are in fact linearly independent. The first set of solutions we have presented are linearly independent by construction. The first set of solutions is such that:

$q_{it'} = y_{it'} = x_{it'} = 0$, for $t' = k, \ldots, l$ and $i = 1, \ldots, F$.

Directions 2 and 3 give a total of $2F(l - k + 1)$ directions. Consider variables $q_{it'}$ and $y_{it'}$ for $t' = k + 1, \ldots, l$ and $i = 1, \ldots, F$. These variables are different than zero on Directions 2 and 3 only.

Consider Case 1 for Directions 2 and 3:

$q_{it'} = b_{it}; y_{it'} = 1$
$q_{it'} = b_{it'+1}; y_{it'} = 1$

Consider Case 2 for Directions 2 and 3:

$q_{it'} = b_{it}; y_{it'} = 1$
$y_{it'} = 1$

One can see that for any $i = 1, \ldots, F$ and $t' = k + 1, \ldots, l$, directions 2 and 3 cannot be expressed as linear combinations of each other (i. e., $b_{it}$ and 1 is not a multiple of $b_{it'+1} and 1$). This shows that in the set of directions we present, there are $2F(l - k)$ linearly independent directions.

For $t = k$, $q_{ik}$ and $y_{ik}$ is different than zero in directions 2, 3 and 5.

Directions 2: $q_{ik} = b_{kl}; y_{ik} = 1$ for $i = 1, \ldots, F$. 

Directions 3: \( q_{ik} = b_{k,l+1}; y_{ik} = 1 \) for \( i = 1, \ldots, F \).

Directions 5: \( q_{ik} = b_{kl}; y_{ik} = 1; q_{Fk} = -b_{kl}; y_{Fk} = -1 \) for \( i = 1, \ldots, F-1 \).

It is obvious that the above directions cannot be expressed as linear combination of each-other. This gives 2F linearly independent directions.

The fourth set has \((F - 1)(l - k + 1)\) directions. For \( t' = k, \ldots, l \) and \( i = 1, \ldots, F - 1 \), we have \( x_{it} = b_t \) for each \( t = k - 1, \ldots, t' - 1 \), that creates a lower triangular matrix. These directions add \((F - 1)(l - k + 1)\) linearly independent directions.

In the fifth set of solutions \( x_{it} \) is different than zero for \( i = 1, \ldots, F - 1 \) in only one direction. \( x_{it} \) is equal to zero in all the other directions. This gives \( F - 1 \) extra linearly independent directions.

In total we have presented (considering Directions 2 to 5):

\[
2F(k - l) + 2F + (F - 1)(k - l + 1) + (F - 1) = (3F - 1)(k - l + 1) + F - 1
\]

linearly independent directions, or:

\[
(3F - 1)(k - l + 1) + F
\]

affinely independent directions. The above directions together with the affinely independent directions from the first set of solutions give:

\[
3FT - F - T - 1
\]

affinely independent directions.

We presented a set of solutions that satisfy the valid inequality as equality and showed that these solutions gave \( \text{dim}(\text{co}(\Phi)) \) affinely independent directions. We have proved that the \((l, S)\) inequalities are facet defining.

\[\Box\]

**Proof of Theorem 2.7.2**: A sequential extreme flow is an extreme flow with the following property: if facility \( i \) produces in time period \( t \), then production satisfies demands in a sequence of time periods \( \tau, \ldots, \tau' \), where \( \tau \geq t \) and \( \tau' \leq T \). By
definition, a path consists of a sequence of nodes and arcs such that there is no
text repetition of the nodes in the path and for any two consecutive nodes an arc exists
connecting them. We first want to show that every path in $G'$ that connects node 1
to $T + 1$, corresponds to an extreme point solution in the extended formulation of the
multi-retailer lot-sizing problem.

Let $P$ be the set of all paths in $G'$ that connect node 1 to $T + 1$, and $p \in P$. Since
$p$ is a path in $G'$, there will be exactly one arc in $p$ starting at node 1 and exactly
one arc ending at node $T + 1$. This implies that demand in periods 1 and $T$ will be
satisfied. Let $(\tau, \tau') \in p$. By construction we know that an arc $(\tau, \tau') \in p$ implies that
demand in periods $\tau, \tau + 1, \ldots, \tau' - 1$ is satisfied from production in a single facility in a
single time period $t$ ($t = 1, \ldots, \tau$). Since $(\tau, \tau') \in p$, than there will be no other arc in
the path that starts at one of the nodes $\tau, \tau + 1, \ldots, \tau' - 1$ and no other arc that ends
at nodes $\tau + 1, \ldots, \tau'$. This implies that demand in periods $\tau, \tau + 1, \ldots, \tau'$ is satisfied
from a single facility in a single time period prior to or beginning at $\tau$.

By definition, there is exactly one arc entering and exactly one arc leaving every
node in a path. Therefore, there will be exactly one arc entering node $\tau$ and exactly
one arc leaving $\tau'$ (since $(\tau, \tau') \in p$), which implies that demands in time periods
before $\tau$ and after $\tau'$ are satisfied by production in (only) one period at exactly one
facility. We conclude that every path in $G'$ that connects nodes 1 to $T + 1$ represents
a production, inventory and transportation schedule that satisfies demand in every
period. This schedule is such that demands are satisfied from production in a single
facility in a single time period. Such a schedule fits the requirements of an extreme
point solution to the multi-facility lot-sizing problem. Therefore, for every path in
$G'$ that connects nodes 1 to $T + 1$ corresponds an extreme point solution of the
multi-facility lot-sizing problem.

The next step is to show that to every sequential extreme point solution of the
feasible region in the extended formulation of the multi-facility lot-sizing problem
corresponds a path in $G'$ that connects node 1 with $T + 1$. Let $R$ represent the
set of all sequential extreme point solutions and \( r \in R \). As such, \( r \) represents a production schedule that satisfies demands during periods \( 1, \ldots, T \). Two important characteristics of \( r \) are the following: (i) demand in every time period is satisfied from production in a single facility in a single time period (ii) a facility at time period \( t \) either does not produce, or produces the demand for one or more periods (these periods should be sequential). Demands in periods \( t = 1, \ldots, T \) are satisfied from production at at most \( F \) different facilities in at most \( T \) periods. Consider the following cases:

**Case 1:** Demands in periods 1 to \( T \) are satisfied from production in different time periods. This case can be generalized to all the tree solutions that are such that: demand in every period is satisfied from a different combinations of facilities and production periods. To these solutions correspond a path \( p \in G' \) such that all the nodes are part of the path (\( p \) consists of nodes \( 1, \ldots, T + 1 \) and arcs \((1, 2), (2, 3), \ldots, (T, T + 1)\)).

**Case 2:** Demands in periods \( \tau, \ldots, \tau' \) are satisfied from production in the same facility in the same time period. All the extreme flow solutions that satisfy this condition have a single arc \((\tau, \tau') \in G'\) and no other arc that that enters nodes \( \tau + 1, \ldots, \tau' \) or leaves nodes \( \tau, \ldots, \tau' - 1 \).

**Case 3:** Demands in periods \( \tau, \ldots, \tau' \) are satisfied from production in different facilities in the same time period. All the extreme flow solutions that satisfy this condition consist of a sequence of arcs such that there will be an arc entering node \( \tau \in G' \) followed by a sequence of arcs \((\tau, \tau + 1), (\tau + 1, \tau + 2), \ldots, (\tau' - 1, \tau')\).

**Case 4:** Demands in periods \( \tau, \ldots, \tau' \) are satisfied from production in the same facility in different time periods. All the extreme flow solutions that satisfy this condition consist of a sequence of arcs in \( G' \) such that: there will be an arc entering node \( \tau \) followed by a sequence of arcs \((\tau, \tau + 1), (\tau + 1, \tau + 2), \ldots, (\tau' - 1, \tau')\).

This shows that to every sequential extreme flow solution to (MF) corresponds a path in \( G' \). This concludes our proof. \( \square \)
REFERENCES


Research Report CRT-96-07, Centre de recherche sur le transports. Université de Montréal, 1996.


BIOGRAPHICAL SKETCH

Sandra Duni Ekşioğlu was born on September 10, 1972 in Tirana, Albania. In 1994, she was awarded a bachelor’s degree in Business Administration from the School of Business, University of Tirana in Tirana, Albania. She earned her master’s degree in Management Sciences from the Mediterranean Agronomic Institute of Chania, Greece in 1996. Then she returned to Albania and worked as instructor in the Business school at the University of Tirana. She began her doctoral studies in the Industrial and Systems Engineering Department at the University of Florida in August 1997 and received her Ph.D. in December 2002.