STOCHASTIC LOT SIZING PROBLEMS UNDER MONOPOLY

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

İhsan Yanıkoğlu

July, 2009

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Hande Yaman (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Oya Ekin Karaşan

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Banu Yüksel Özkaya

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

STOCHASTIC LOT SIZING PROBLEMS UNDER MONOPOLY

İhsan Yanıkoğlu
M.S. in Industrial Engineering
Supervisor: Assoc. Prof. Dr. Hande Yaman
July, 2009

In this thesis, we study stochastic lot sizing problems under monopoly. We consider production planning of a single item using uncapacitated resources over a multi-period time horizon. The demand uncertainty is modeled via a scenario tree structure. Each node of the tree corresponds to a scenario of demand realization with an associated probability.

We first consider the stochastic lot sizing problem under monopoly (SLS), which addresses the period based production plan of a manufacturer with uncertain demands and a monopolistic supplier. We propose an exact dynamic programming algorithm to solve the SLS problem in polynomial time. The second problem we consider, the stochastic lot sizing problem with extra ordering (SLSE), is based on two-stage stochastic programming. In addition to the period based production decision variables of the SLS model, there exist scenario based extra ordering decision variables in the problem setting of SLSE. We develop two families of valid inequalities for the feasible region of the introduced SLSE model. The required separation algorithms of both valid inequalities are presented along with their implementations with branch-and-cut algorithm in solving SLSE. An extensive computational analysis with branch-and-cut algorithms shows the effectiveness of these inequalities.

Keywords: stochastic lot sizing problem, two-stage stochastic programming, scenario tree, dynamic programming, branch-and-cut algorithm.

ÖZET

TEKEL ALTINDA RASSAL ÖBEK BOYUTLANDIRMA PROBLEMLERİ

İhsan Yanıkoğlu
Endüstri Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Doç. Dr. Hande Yaman
Temmuz, 2009

Bu tezde, tekel altında rassal öbek boyutlandırma problemleri çalışılmıştır. Tek bir birimin sınırsız kaynak ile çok dönemli zaman çevreninde üretim planlaması ele alınmıştır. Talep belirsizliği bir senaryo ağacı yapısıyla modellenmiştir. Senaryo ağacındaki herbir düğüm, belirli bir talep miktarının ilişkili bir olasılıkla gerçekleştiği bir senaryoya karşılık gelmektedir.

İlk olarak, bir üreticinin belirsiz talep ve tekelci sunucunun varlığında üretim planlamasını konu alan, tekel altında rassal öbek boyutlandırma problemi ele alınmıştır. Bu problem için çözüm yolu olarak da polinom zamanda çalışan bir dinamik izlenceleme algoritması önerilmiştir. İkinci sırada ele alınan ek ısmarlamalı rassal öbek boyutlandırma problemi, iki aşamalı rassal izlencelemeye dayanmaktadır. Ek ısmarlamalı rassal öbek boyutlandırma probleminde, rassal öbek boyutlandırma modelindeki dönem bazlı üretim karar değişkenlerine ek olarak, senaryo bazlı ek ısmarlama karar değişkenleri bulunmaktadır. Sunulan ek ısmarlamalı rassal öbek boyutlandırma modelinin uygun çözüm kümesi için iki geçerli eşitsizlik ailesi geliştirilmiştir. Her iki geçerli eşitsizlik ailesi için gerekli olan ayrıştırma algoritmaları, ek ısmarlamalı rassal öbek boyutlandırma probleminin çözümünde kullanılan dal-kesi algoritması içindeki uygulamalarıyla birlikte verilmiştir. Dal-kesi algoritması üzerine yapılan kapsamlı deneysel çözümlemeler bu eşitsizliklerin etkinliğini göstermiştir.

Anahtar sözcükler: rassal öbek boyutlandırma problemi, iki aşamalı rassal izlenceleme, senaryo ağacı, dinamik izlenceleme, dal-kesi algoritması.

Acknowledgement

I would like to thank to my mother and father, Meral and Cengiz Yanıkoğlu, for their invaluable love and support. This study is devoted to you and it is just the beginning.

I would like to express my most sincere thanks to my advisors Assoc. Prof. Hande Yaman and Assoc. Prof. Oya Ekin Karaşan. You were always more than a professor for me. I would like to thank once more for your everlasting patience and helps throughout this study.

I would like to thank to Adnan Tula, Utku Guruşçu, Safa Bingol, Duygu Tutal, Ezel Budak, Onur Özkök, Zeynep Aydın, Könül Bayramova, Haluk Eliş, Didem Batur, Can Öz, Sıtkı Gülten, Burak Paç, Merve Çelen, Burak Ayar, Gökay Erön and my other collegues. The friendly environment you have created at Bilkent University helped me a lot during my studies. From now on we can continue our lives in different places but we should not forget that we can grow separately without growing apart.

I would like to thank to Assist. Prof. Banu Yüksel Özkaya for accepting to read and review this thesis.

I would like to thank to TÜBİTAK for the financial support they have provided to me for this research.

Finally, I would like to thank to the one who leaves footprints in my heart.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Lot can be defined as a group of items produced at one expedition. As it is stated on the title of the article by Harris [7], the problem of lot sizing is to decide on 'How many parts to make at once'. So, demand volumes of several periods are produced in one period. This is logical in cases when there exist setup related costs on production planning of a product.

Setting up a production system to produce a certain product or multiple products consumes time and money, this is why lot sizing decision is an important phenomenon in production planning. Producing a demand different from its order time results in inventory holding or backlogging costs. Inventory holding cost is incured when a product is carried as stock from one period to a following one. Backlogging cost is paid when a product is received by its customer later then its demand period. The lot sizing problem aims to find optimal timing and amount of production, in order to achieve a least cost production plan over a defined time horizon. In other words, the objective of lot sizing problem is to find optimal production decisions to balance the interaction between setup, production, inventory holding and backlogging costs.

Even though lot sizing problems mostly arise when there exist setup and backlogging costs, they can also be used when those costs are negligible. This is because, lot sizing based production planning decisions must still be undertaken

when there exist limited capacities in the resources of the production system. As an example, a production system with low capacity machines may use lot sizing in order to avoid unused capacities and overcome late deliveries of products to customers.

Lot sizing models can be grouped according to several factors. One of them is the characteristic of demand: deterministic or uncertain demand. There can be several random factors that affect lot sizing decisions and the random demand is potentially the most important one. For this reason, stochastic lot sizing models take into account the demand uncertainty. On the other hand, in the deterministic demand there does not exist an uncertainty, so that the relevant demand data is known with certainty.

As it is stated by Quadt [8], production planning problems are too complex to be solved by straightforward solution techniques without considering the time interval of the problem. It is the fact that, the production planning problems can also be grouped according to time scale, such as long-term, medium-term and short-term. The long-term production planning considers seasonality effects and it works with long-term or medium-term demand forecasts. The general long-term time frame is one to several years with periods of one to three months. In addition, there are conceptual production planning problems which include long time frames that are assumed to be infinite.

The medium and short-term production planning problems are more detailed. These problems use short-term forecasts or direct customer orders. Short-term plans are generally conducted for end products. Time frame is one to several months and periods can be organized by weekly or daily basis.

Furthermore, lot sizing problems can be categorized according to the number of items and the type of resource. Most of the lot sizing problems are based on planning the production of a single item. However, though few in number, there are also multi-item lot sizing problems that are studied in the literature. The items can be produced either with capacitated resources or uncapacitated resources. In capacitated problems, the amount of production at each period can not exceed a given capacity, whereas in uncapacitated problems the production

at each period can be an arbitrary amount.

In this thesis, we study lot sizing problems that arise for industrial companies which use raw materials such as glass, wood and stone in their value adding process. These raw materials are generally obtained from natural resources in huge amounts so their processing and transportation require high setup costs. In addition, suppliers of these raw materials are generally monopolistic companies. For example, Sisecam Company is the single supplier of glass used for returnable packaging materials in Turkey. Furthermore, monopolistic suppliers have various customers from different industries. This is why, they generally enforce manufacturers to give their annual or semi-annual production plans at the beginning of the planning horizon. But, for the manufacturer it is hard to forecast its demand for a long planning horizon. So according to the given conditions, the structure of demand for the manufacturer is stochastic by nature. Because of this reason, the production planning problem of the manufacturer is stochastic and uncertainty of demand must be considered in the required lot sizing decision.

In the light of the above motivation, we have decided to study single-item, multi-period, uncapacitated, stochastic lot sizing problems under supplier monopoly.

The stochastic nature of the problems is taken into account with scenarios. At each period there exist multiple scenarios, each of which represents unique realization of demand. We use a scenario tree to represent the mentioned structure. So that, each node of the tree is a scenario or a realization of demand with its assigned probability. In addition, except the root node every node in the scenario tree has a unique parent. We have used the same scenario tree structure throughout our study.

Our first problem is the stochastic uncapacitated lot sizing under monopoly (SLS). In SLS, we need to decide on the amount of production at each period with unlimited resources. Currently, the raw materials are supplied to the manufacturer by a single supplier and this is why there exists monopoly in the market structure and competition for prices are outside the scope of study. The costs related with the problem are inventory holding cost, backlogging, production cost

and setup cost of production. We can summarize the objective of the problem as to find the optimal production planning decisions for the manufacturer at each period to minimize the expected inventory holding, backlogging and production related costs of the system.

In order to solve the SLS problem, we analyzed the structure of optimal solutions. We found that for any instance of the problem at least one optimal solution satisfies a specific property named as "production path". Using this property, we developed an exact polynomial time dynamic programming algorithm for the SLS problem.

The second problem is the stochastic uncapacitated lot sizing problem with extra ordering under monopoly (SLSE). SLSE is based on two stage stochastic mathematical programming. Similar to SLS, first-stage decisions coincide with the amount of production made at each period. The second-stage decisions are the amounts of extra orders variables. The second-stage decisions are given according to first-stage decisions and realizations of demand. Thus they can be thought as scenario based corrective actions. Backlogging is excluded from SLSE so the related costs of the problem include inventory holding cost, production cost, extra ordering cost, setup costs of doing production and extra ordering. The aim of the problem is to find optimal production and extra ordering decision plans with minimum cost. In the solution approach, we have developed two families of valid inequalities. The required separation algorithms of both inequalities are established and then these separation algorithms are used with the branch-and-cut algorithms to solve the SLSE problem.

The organization of the thesis is as follows:

In Chapter 2, we provide a review of the related literature.

In Chapter 3, first, the mathematical formulation of SLS is given. Second, production path property of SLS is developed and then this property is used to establish an exact dynamic programming algorithm for the problem. In the end, the complexity analysis of dynamic programming algorithm of SLS is presented.

In Chapter 4, we present a mathematical formulation of the stochastic uncapacitated lot sizing problem under monopoly with extra ordering (SLSE). Then, we reformulate the problem and develop two families of valid inequalities for SLSE which are called (ℓ, S, L) and (Q, S_Q, L_Q). Finally, we provide separation algorithms for (ℓ, S, L) and (Q, S_Q, L_Q) inequalities and conclude the chapter.

In Chapter 5, computational results with different problem parameters are are provided for SLSE.

In Chapter 6, we summarize our results and our contribution to the literature and suggest future research directions.

Chapter 2

Literature Review

In this chapter we provide the review of the literature that is related with the problems under concern.

The traditional lot sizing problem is to decide on the amount of production at each period over a given finite time horizon in order to minimize total production, holding and setup costs.

We can start with deterministic lot sizing problems which are studied in the literature. The well known economic order quantity model (EOQ) assumes stationary demand with constant demand rate over time, single commodity and infinite time horizon together with infinite capacity resources. It seeks the optimal solution under the given assumptions.

Seminal work of Wagner and Within [11] is on economic lot size model. They assume that costs of buying and selling a product is constant throughout all periods, as result inventory related costs are of concern. Demands and all costs are positive and the aim of the problem is to meet demands with a minimum cost. There exists an optimal solution to Wagner-Within problem that satisfies one specific condition called zero inventory ordering. So that, at each period we either hold inventory or make production, but not both. In other words, at each period either production or inventory becomes zero, so that production decision

is made when inventory depletes to zero. Using this result, the problem can be reformulated as a shortest path problem and it can be solved in polynomial time.

The capacitated lot sizing problem (CLSP) is an extension of Wagner-Within problem. CLSP differs from Wagner-Within problem in the sense that it considers capacitated resources at each period.

Another field of interest in the lot sizing literature is stochastic lot sizing problems. As we have stated before, uncertainty is generally based on stochasticity of demands. Even though, demand is generally assumed to be a known data by using forecasting techniques, there can be numerous events such as seasonality and customer behavior that can affect the structure of demand. So that, forecasts are subject to deviate from real numbers because of changing conditions. This is why the stochastic nature of demand must be taken into account in order to simulate real life structure of problems in a better way. But, there are limited number of studies in the literature suggesting lot sizing models for environments which have demand uncertainty.

For stochastic programming we can refer to the literature survey of Schultz [9]. Modeling with uncertainty leads to a wide range of stochastic programming options but in the survey only two stage models and some of its extensions are taken into account. Two stage stochastic programming model reflects one commonly used example of hierarchical decision making. The decision factors are divided into two parts as first-stage and second-stage variables. There is also the source of uncertainty which is generally defined as a function on some probability space and second-stage decisions are dependent on this function. The first-stage decisions are given before uncertainty has been revealed and the second-stage decisions are made. Once the first-stage decisions are made and the uncertainty has been observed, the second-stage decisions can be solved as deterministic problem.

In summary, the first-stage corresponds to the decisions that have to be made under uncertainty of problem data. On the other hand, second stage allows corrective action for the first-stage decision, when stochastic nature of the problem disappears. The decision problem under uncertainty is to select the first-stage decision variables and then take the corrective actions accordingly. The objective

of the two-stage stochastic problem is to find the minimum total expected cost.

In the paper, the author also presents the multi-stage extension of stochastic mixed integer programming problems. Then the related branch-and-bound and disjunctive cuts algorithms are presented and some decomposition techniques are mentioned for complex multi-stage problems.

Guan and Miller [5] have studied the stochastic uncapacitated lot sizing (SULS) problem by using scenario based demand uncertainty. The main decision factors at SULS problem are scenario based production variables. As demand is uncertain and it could be so high in some scenarios, the problem is also included backlogging to obtain feasibility. So, the costs of the problem are inventory holding, production, production setup and backlogging. The decision problem is to decide the amount of production at each scenario to obtain the minimum total expected sum of setup, backlogging, production and inventory holding costs.

In the paper, the authors develop a polynomial-time algorithm for the SULS problem. They show that complexity of their dynamic program is $O\left(n^2 max\left\{C, log(n)\right\}\right)$, where n represents the total number of nodes in the scenario tree and C is the maximum number of children for each node at tree. In addition, they showed that the SULS problem without setup costs is continuous, piecewise-linear and convex.

In our study, we have used the same scenario tree structure with the SULS problem, but our mathematical formulations are different. In our stochastic lot sizing problem under monopoly, we have used the period based production variables, which is different from the scenario based production variables of the SULS problem. In the second problem which is stochastic lot sizing with extra ordering under monopoly, we generalized the SULS problem by adding an extra decision variable to the mathematical formulation of the problem.

Ahmed et al. [1] formulated a multistage stochastic capacity expansion problem (SCAP) with fixed-charge expansion costs. Their problem can be described as deciding the timing and level of capacity acquisitions for set of facilities, with

a policy of allocating the available capacity to satisfy the demands of multi-products. Objective is to minimize total investment and allocation costs of n period planning horizon. The uncertainty is modeled by scenario tree, at each time period there exists multiple nodes, so that each node layer matches a time period t in the scenario tree. In the paper, they present a decomposition technique of the mathematical formulation into smaller deterministic problems which can be solved easily by an efficient heuristic.

The authors reformulated SCAP to show the lot sizing substructure of the formulation. They presented that there is one to one correspondence between the set of feasible solution of single resource SCAP and the set of feasible solutions of the stochastic lot sizing problem (SLSP).

As result, they use reduction of SCAP into SLSP for finding lower bounds on SCAP, because LP relaxation of reformulation results smaller gap from optimality with respect to LP relaxation of SCAP. Furthermore, they also presented a heuristic that finds feasible solution for SCAP and they used this heuristic together with LP relaxation of reformulated problem in order to apply branch and bound algorithm for the original problem. According to these results, we can also conclude that deriving a polynomial time algorithm for stochastic capacity expansion problem is nontrivial.

In the pioneer work of Barany et al. [2], a class of valid inequalities are developed for lot sizing problems. They introduce (ℓ, S) valid inequalities for the single-item lot sizing problem. The parameter ℓ represents a period of the planning horizon and the set S is the subset of the set L, where L is the set of periods between the initial period and the period ℓ. The authors show that the (ℓ, S) inequalities are facet defining for the single-item uncapacitated lot sizing problem. Then, they formulate the complete separation of the (ℓ, S) inequalities for single-item case.

Using the (ℓ, S) valid inequalities, the authors also present some practical reformulations for the multi-item lot sizing problems. According to their conclusion, for multi-product case, the (ℓ, S) inequalities are valuable computational tools. However it is important to obtain stronger valid inequalities with better

lower bounds, which take into account the capacity constraints of multi-item lot sizing problems.

Guan et al. [4] studied the multi-stage stochastic integer programming formulation of uncapacitated lot sizing problem under uncertainty. They have proved that the classical (ℓ, S) inequality used in deterministic lot sizing problems, are also valid in the stochastic case. Then they extended this inequality to a more general type of inequality, called (Q, S_Q) inequality. (Q, S_Q) inequality is generalization of (ℓ, S) inequality in such a way that (ℓ, S) is only defined on single paths of scenario tree but (Q, S_Q) can be used on any given subtree, which also includes single paths of the scenario tree.

In addition, it is shown that (Q, S_Q) inequalities are facet defining. Then required separation algorithm is developed for (Q, S_Q) inequality and it is implemented in a branch and cut algorithm. Resulting solutions from the implementation are computed and it is seen that use of (Q, S_Q) is efficient in reducing the number of nodes and LP relaxation gaps with respect to the default CPLEX branch-and-bound algorithm.

Halman et al. [6] worked on the special case of SLSP and they developed a K-approximation algorithm, which guarantees the optimal value of the problem will be no more than K times the value found by the algorithm. According to their definition K can be a value bigger than one and there exists a K-approximation function which always has values between the objective value and K times the objective value of the original problem. By using K-approximation functions, the authors developed fully polynomial time approximation scheme for single item lot sizing problem.

Finally, our contribution to the literature is that we have used two different sources of decision variables such as period based production and scenario based extra ordering. In addition, we have used different mathematical formulations with respect to previous stochastic lot sizing models which are studied in the literature. In the solution approach, we have developed an exact algorithm for one of our stochastic models. For the second model, we have developed two different families of valid inequalities. The valid inequalities we have developed

are generalizations of well known (ℓ, S) and (Q, S_Q) inequalities, which have been used for lot sizing problems studied in the literature.

Chapter 3

Stochastic Uncapacitated Lot Sizing Problem under Monopoly (SLS)

In this chapter, we consider a single-item multi-period uncapacitated stochastic lot sizing problem. Stochastic nature of the problem comes from the uncertain demand and we use scenario trees to model uncertainty. Each level of the scenario tree represents a period of the finite horizon. Nodes represent the scenarios and each node has a realization of demand and a probability associated with it.

The cost function of the problem includes production, inventory holding, backlogging and setup costs. There exists a monopolistic supplier so that competition for prices is outside the scope of this study. The objective of the problem is to decide on the amount of production at each period, in order to find the minimum expected total cost over the scenarios of the problem.

In the following sections of this chapter, we will describe the mathematical formulation and the production path property of the problem. Then we will develop a dynamic programming algorithm for the problem and we will finish this chapter with the complexity analysis of the dynamic programming algorithm.

3.1 Formulating the Problem

Let T be the set of periods in the planning horizon and V be the set of nodes in the scenario tree with $|V| = n$.

3.1.1 Parameters

$f_t \equiv$ the unit production cost at period $t \in T$
$\beta_t \equiv$ the setup cost of production at period $t \in T$
$h_i \equiv$ the unit holding cost at node $i \in V$
$b_i \equiv$ the unit backlogging cost at node $i \in V$
$d_i \equiv$ the demand at node $i \in V$

3.1.2 Decision Variables

$o_t \equiv$ the production amount of product at period $t \in T$
$z_t \equiv \begin{cases} 1, & \text{if there exists a production at period } t \in T \\ 0, & \text{o.w} \end{cases}$
$s_i^+ \equiv$ the inventory of product at node $i \in V$
$s_i^- \equiv$ the backlog amount of the product at node $i \in V$

3.1.3 Notation

We shall adopt the following notation throughout this chapter.

$$\begin{aligned} N_t &\equiv \text{ the set of nodes in period } t \in T. \\ N_{t_1,t_2} &\equiv \bigcup_{t=t_1}^{t_2} N_t;\ t_1, t_2 \in T \text{ such that } t_1 < t_2. \\ t(i) &\equiv \text{ the period of node } i \in V \end{aligned}$$

.

$a(i) \equiv$ the immediate predecessor of node $i \in V$.

$d_{ij} \equiv$ the summation of demands on the unique path from node $i \in V$ to node $j \in V$ in the scenario tree such that $t(i) < t(j)$ and i is a predecessor of j.

$V(i) \equiv$ the set of descendants of node $i \in V$ including i.

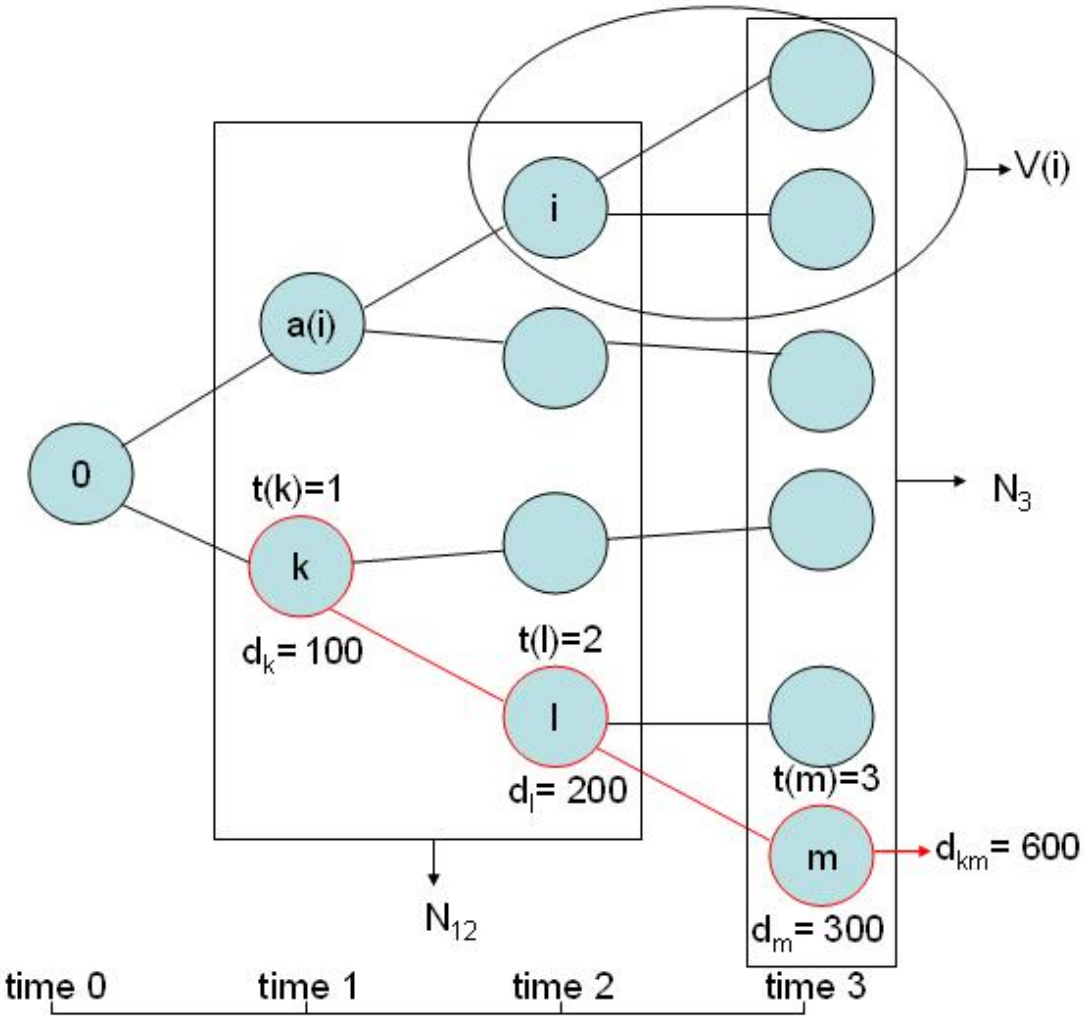

Figure 3.1: Scenario Tree

In figure 3.1, we represent a general scenario tree with the notation used in the SLS problem.

3.1.4 Mathematical Formulation

$$(M1): \quad \min \qquad c(o, s^+, s^-, z) = \sum_{t \in T} (f_t o_t + \beta_t z_t) + \sum_{i \in V} \left(h_i s_i^+ + b_i s_i^-\right)$$

$$s.t.$$

$$s_0^-,\ s_0^+ = 0 \tag{3.1}$$

$$s_{a(i)}^+ + s_i^- + o_{t(i)} = d_i + s_i^+ + s_{a(i)}^- \qquad \forall i \in V \tag{3.2}$$

$$o_t \leq M z_t \qquad \forall t \in T \tag{3.3}$$

$$s_i^+, s_i^- \geq 0, \qquad \forall i \in V \tag{3.4}$$

$$o_t \geq 0, \quad z_t \in \{0, 1\} \qquad \forall t \in T. \tag{3.5}$$

The initial inventory and backlog are assumed to be zero. Constraint (3.2) is the inventory balance constraint of the problem. By constraint (3.2), we ensure that at each node i, the incoming inventory level $(s_{a(i)}^+ - s_{a(i)}^-)$ plus the amount of production at period $t(i)$ is equal to the outgoing inventory level $(s_i^+ - s_i^-)$ plus the demand of node i.

In constraint (3.3), the setup identifier z_t is set to one when there exists a production at period t. Setup identifier is used in the objective function in order to include setup related costs of the problem in production periods. M is a relatively big number with respect to the amount of production that can be done at any period t, $(M \geq max_{\ell \in N_{|T|}} d_{0\ell})$.

Constraints (3.4) and (3.5) ensure that all inventory related variables are non-negative and the setup identifier is a binary variable.

In the mathematical formulation, the probability of each node is included in cost parameters h_i and b_i, which are holding and backlogging costs, respectively. Finally, in the objective function we minimize the total expected production, setup of production, inventory holding and backlogging costs. The feasible set of solutions will be denoted by X_{SLS}.

Theorem 1 *(Production Path Property)*
Let $n(t)$ be the next production period after period $t \in T$ such that

$$n(t) \equiv \begin{cases} min\left\{k \in T \ : \ k > t \ and \ o_k > 0\right\}, & if \ t < max\left\{l \in T : o_l > 0\right\} \\ |T| + 1, & o.w \ . \end{cases}$$

There exists an optimal solution for $M1$ such that for all $t \in T$, for which $o_t > 0$, there exists $i \in N_t$ and $j \in N_{t,n(t)-1} \bigcap V\left(i\right)$ where $o_t = d_{ij} - s^+_{a(i)} + s^-_{a(i)}$.

Proof.
Let (o, s^+, s^-, z) be an optimal solution to $M1$ and let A be the set of nodes where s_i^+'s are positive in the optimal solution. Suppose there exists $t \in T$, for which $o_t > 0$ and $o_t \neq d_{ij} - s^+_{a(i)} + s^-_{a(i)}$ for all $i \in N_t$ and $j \in N_{t,n(t)-1} \cap V(i)$. It implies that either $s_i^+ > 0$ or $s_i^- > 0$ for all $i \in N_{t,n(t)-1}$.
Two alternative solutions $(\hat{o}, \hat{s}^+, \hat{s}^-, \hat{z})$ and $(\bar{o}, \bar{s}^+, \bar{s}^-, \bar{z})$ such that

- $\hat{o}_t = o_t - \epsilon$
- $\hat{s}^+_k = s^+_k - \epsilon, \ \forall k \in A \cap N_{t,n(t)-1}$
- $\hat{s}^-_k = s^-_k + \epsilon, \ \forall k \in N_{t,n(t)-1} \backslash A$
- if $n(t) \neq |T| + 1$ then $\hat{o}_{n(t)} = o_{n(t)} + \epsilon$
- $\hat{s}^+_k = s^+_k, \ \hat{s}^-_k = s^-_k \ \forall k \in V \backslash N_{t,n(t)-1}$
- $\hat{o}_k = o_k \ \forall k \in T \backslash \left\{t, n(t)\right\}$
- $\hat{z} = z$

and

- $\bar{o}_t = o_t + \epsilon$
- $\bar{s}^+_k = s^+_k + \epsilon, \ \forall k \in A \cap N_{t,n(t)-1}$
- $\bar{s}^-_k = s^-_k - \epsilon, \ \forall k \in N_{t,n(t)-1} \backslash A$
- if $n(t) \neq |T| + 1$ then $\bar{o}_{n(t)} = o_{n(t)} - \epsilon$

- $\bar{s}_k^+ = s_k^+,\ \ \bar{s}_k^- = s_k^-\ \ k \in V \backslash N_{t,n(t)-1}$
- $\bar{o}_k = o_k\ \ \forall k \in T \backslash \{t, n(t)\}$
- $\bar{z} = z$

are also feasible for $\epsilon = min\left\{min_{k \in A \cap N_{t,n(t)-1}} \left\{s_k^+ : s_k^+ > 0\right\}, min_{k \in N_{t,n(t)-1} \backslash A} \left\{s_k^- : s_k^- > 0\right\}\right\}$. The difference of two cost functions with respect to initial one are represented below.

$$c(\hat{o}, \hat{s}^+, \hat{s}^-, \hat{z}) - c(o, s^+, s^-, z) \quad = \quad -f_t\epsilon - \sum_{k \in A \cap N_{t,n(t)-1}} h_k\epsilon + \sum_{k \in N_{t,n(t)-1} \backslash A} b_k\epsilon + f_{n(t)}\epsilon.$$

$$c(\bar{o}, \bar{s}^+, \bar{s}^-, \bar{z}) - c(o, s^+, s^-, z) \quad = \quad f_t\epsilon + \sum_{k \in A \cap N_{t,n(t)-1}} h_k\epsilon - \sum_{k \in N_{t,n(t)-1} \backslash A} b_k\epsilon - f_{n(t)}\epsilon.$$

We assume $f_{|T|+1} = 0$. Let $K = f_t + \sum_{k \in A \cap N_{t,n(t)-1}} h_k - \sum_{k \in N_{t,n(t)-1} \backslash A} b_k - f_{n(t)}$. The two cost differences given above are smaller and equal to $K\epsilon$ or $-K\epsilon$.

If the value of K is non-zero then our initial solution is not optimal and it is a contradiction. So K is equal to zero and solutions $(\bar{o}, \bar{s}^+, \bar{s}^-, \bar{z})$, $(\hat{o}, \hat{s}^+, \hat{s}^-, \hat{z})$ are both optimal. Also there exists $i \in N_t$ and $j \in N_{t,n(t)-1} \bigcap V(i)$ such that $\hat{o}_t + \hat{s}_{a(i)}^+ - \hat{s}_{a(i)}^- = d_{ij}$ or $\bar{o}_t + \bar{s}_{a(i)}^+ - \bar{s}_{a(i)}^- = d_{ij}$.

□

3.2 A Dynamic Programming Algorithm

In this section, we will develop a dynamic programming algorithm to solve the stochastic uncapacitated lot sizing problem under monopoly (SLS).

The most important factor that reduces complexity of the algorithm is the production path property of SLS. What production path property says is that, if there exists a production at period $t \in T$ then it is equal to the summation of demand from a node $i \in N_t$ to some descendant of node i in the scenario tree minus incoming inventory level of node i from the previous period.

If incoming inventory level is positive, it can compensate some part of demand in upcoming periods; else it can be thought as some additional demand that comes from previous period's backlogs.

With this property, the production quantity o_t can take at most n different values in each production period $t \in T$, where n is equal to total number of nodes in the scenario tree. This allows us to define a recursion formula.

3.2.1 Cost Function

Let $H_t(w)$ be the optimal cost function for periods $t, \ldots, |T|$ and the total amount of production until period t is w, which is $w = \sum_{i=1}^{t-1} o_i$.

In period $t \in T$, if production occurs, then the production quantity is $o_t = d_{0j} - w$ for some $i \in N_t$ and $j \in V(i)$ such that $d_{0j} > w$, where $i = 0$ represents the root node. The equality $o_t = d_{0j} - w$ is the direct result of production path property of SLS. The resulting production cost function is,

$$
\begin{aligned}
H_t(w) \quad = \quad min \Bigg\{ & \sum_{i \in N_t} \left(h_i (w - d_{0i})^+ + b_i (d_{0i} - w)^+ \right) + H_{t+1}(w) \ , \qquad (3.6) \\
& \beta_t \ + min_{\hat{t} \in T \,:\, \hat{t} > t} \; min_{j \in N_{t,\hat{t}} \,:\, d_{0j} > w} \Big\{ f_t (d_{0j} - w) + \\
& \sum_{k \in N_{t,\hat{t}}} \left(h_k (d_{0j} - d_{0k})^+ + b_k (d_{0k} - d_{0j})^+ \right) + H_{\hat{t}+1} (d_{0j}) \Big\} \Bigg\}.
\end{aligned}
$$

To begin with, in the $H_t(w)$ function we take the minimum of two possible scenarios for period $t \in T$. First one is the no-production case and we calculate backlogging and holding costs of nodes at period t in this scenario. The latter one is the production case, in which we calculate the cost of doing production in addition to the holding and backlogging costs. There exists a fixed setup cost of doing production at each period $t \in T$ independent of the status of the previous period. This is why we have the additional cost β_t at the very beginning of the cost function of the production case.

In the first line of (3.6), we calculate the cost of doing no production. If $(w - d_{0i})$ is positive, we take into account the related holding cost, otherwise the

backlogging cost is calculated. In the end, we refer to the upcoming period $t+1$ in a recursive manner.

In the second line of (3.6), we calculate the production cost of period t. Then in the third line we decide on the next production period $\hat{t}+1$ and calculate the inventory holding or backlogging costs of nodes between the current and the next production period. Finally, we refer to the upcoming period $\hat{t}+1$ and we update our total production amount as d_{0j}.

It should be noted that $H_{|T|+1}(w) = 0$ for all $w \geq 0$ and it is the base case of the recursive cost function. If initial inventory is zero, the function that gives optimal value for SLS is $H_1(0)$. Otherwise, $H_1(s_0^+ - s_0^-)$ becomes the optimal value.

3.2.2 Complexity Analysis

In order to analyze the complexity of the dynamic programming algorithm, we need to focus on the structure of the recursive cost function $H_t(w)$. To begin with, t can take $|T|$ different values, where $|T|$ is the total number of periods in the given scenario tree. In addition, for any given $t \in T$, w of $H_t(w)$ is restricted to at most n different values by production path property, where n is equal to total number of nodes at the scenario tree.

Proposition 1 *The dynamic programming algorithm of the SLS problem can be solved in $O(|T|^2 n^3)$ steps.*

Proof.
At the beginning of the algorithm, the demand parameters (d_{0j}) that are used in the cost function $H_t(w)$ must be initialized. For a single node $j \in V$, the demand parameters can be calculated in $O(|T|)$ steps. Since there are n nodes in the system, initialization takes $O(|T|n)$ steps. In the following parts we will show that it will be dominated by the number of steps required in evaluating the function $H_t(w)$.

The cost function $H_t(w)$ includes two possible cases, namely, production and no-production.

For a given $t \in T$ and w of $H_t(w)$, let us consider the no-production case of the cost function $H_t(w)$, which coincides with the first term of the minimization function in the expression (3.6). In non-production case, we calculate the inventory holding and backordering cost of nodes in the set N_t, since there can be at most $|N_{|T|}|$ number of nodes in this set, the cost corresponding to no-production case of the algorithm can be calculated in $O(|N_{|T|}|)$ steps.

In the production case we can refer to the second and third line of the expression (3.6). For given values of $t \in T$, w of $H_t(w)$ and $\hat{t} \in T$, the production amount o_t can have at most $|N_{t,\hat{t}}|$ different values by the production path property of SLS, where $\hat{t}$ represents the next decision period after t. When o_t is decided, the cost calculations of the nodes in the set $N_{t,\hat{t}}$ can be evaluated in $O(|N_{t,\hat{t}}|)$ steps. So, when $\hat{t}$ is given, selecting the value of the decision variable o_t together with necessary calculations of the cost function $H_t(w)$ can be completed in $O(|N_{t,\hat{t}}|^2)$ steps for a node set $N_{t,\hat{t}}$. There can be at most n number of nodes in the set $N_{t,\hat{t}}$ and this is why $O(|N_{t,\hat{t}}|^2) < O(n^2)$. Eventually, $\hat{t}$ can take values between t and $|T|$ and the final complexity of calculations in production case can be simplified as $O(|T|n^2)$ steps, since

$$\sum_{\hat{t}=t+1}^{|T|} |N_{t,\hat{t}}|^2 < (|T| - t)n^2 < |T|n^2$$

is satisfied.

When t and w are fixed the complexity of calculations in the cost function $H_t(w)$ is $O(|N_{|T|}|) + O(|T|n^2)$ steps. For changing values of t and w, the final complexity of the dynamic programming algorithm becomes $O(|T|n)\left(O(N_{|T|}) + O(|T|n^2)\right)$ which can be simplified as $O(|T|^2n^3)$ steps.

□

From Proposition 1, we can conclude that the dynamic programming algorithm runs in polynomial time in n.

Chapter 4

Stochastic Uncapacitated Lot Sizing Problem with Extra Ordering under Monopoly (SLSE)

In this chapter, we consider an extension of the stochastic uncapacitated lot sizing problem. The stochastic characteristic of the problem comes from the uncertainty of demands. Random behaviour of the demand is modeled via scenarios and information of scenarios are taken from a scenario tree. The structure of the scenario tree is similar to the one that we have described before. In summary, each node represents a demand realization with its associated probability.

The problem has two main decisions: period based production and scenario based extra ordering. The problem is defined on multi-period time horizon and backlogging is not allowed. So, the two main differences of (SLSE) from (SLS) are these: backlogging is not allowed and we can give scenario dependent extra ordering decisions.

Our cost function includes production, extra ordering and inventory holding expenses. Furthermore, there exits a fixed setup cost of doing production at

each period and a penalty of extra ordering at each scenario of the system. So, the objective of the problem is to decide on the amount of production at each period and extra ordering decisions at scenarios of the system with minimum total expected cost.

In the following sections of this chapter, we will present two different mathematical formulations of the SLSE problem. Then we will develop two families of valid inequalities for the SLSE problem, called (L, S, L) and (Q, S_Q, L_Q) inequalities. Finally, we will present the separation algorithms of the valid inequalities and conclude the chapter.

4.1 Formulating the Problem

Let T be the set of periods and V be the set of nodes in the scenario tree with $|V| = n$. Let $\ddot{T}$ be the scenario tree with periods T and nodes V, $\ddot{T} := \{V, T\}$.

4.1.1 Parameters

$f_t \equiv$ the unit production cost at period $t \in T$
$\beta_t \equiv$ the setup cost of production at period $t \in T$
$\gamma_i \equiv$ the penalty cost of giving extra order at node $i \in V$
$\alpha_i \equiv$ the unit production cost for each extra order at node $i \in V$
$h_i \equiv$ the unit holding cost at node $i \in V$
$d_i \equiv$ the demand at node $i \in V$

4.1.2 Decision Variables

$o_t \equiv$ the production amount of product at period $t \in T$
$z_t \equiv \begin{cases} 1, & \text{if there exists a production at period } t \in T \\ 0, & \text{o.w} \end{cases}$
$s_i \equiv$ the inventory of product at node $i \in V$

$x_i \equiv$ the amount of extra orders at node $i \in V$

$$y_i \equiv \begin{cases} 1, & \text{if there exists an extra order at node } i \in V \\ 0, & \text{o.w} \end{cases}$$

4.1.3 Notation

We shall adopt the following notation throughout this chapter.

$N_t \equiv$ the set of nodes on period $t \in T$.

$N_{t_i t_j} \equiv \bigcup_{t=t_i}^{t_j} N_t$; $t_i, t_j \in T$ such that $t_i < t_j$.

$t(i) \equiv$ the period of node $i \in V$.

$a(i) \equiv$ the immediate predecessor of node $i \in V$.

$P(i,j) \equiv$ the set of nodes on the path from node $i \in V$ to node $j \in V$ such that $t(i) < t(j)$.

$P(i) \equiv P(0,i)$, the set of nodes on the path from root node to node $i \in V$.

$d_{ij} \equiv$ the summation of demands on the unique path from node $i \in V$ to node $j \in V$ in the scenario tree such that $t(i) < t(j)$ and node i is predecessor of node j .

$C(i) \equiv$ the set of immediate descendants of node $i \in V$.

$V(i) \equiv$ the set of descendants of node $i \in V$ including i.

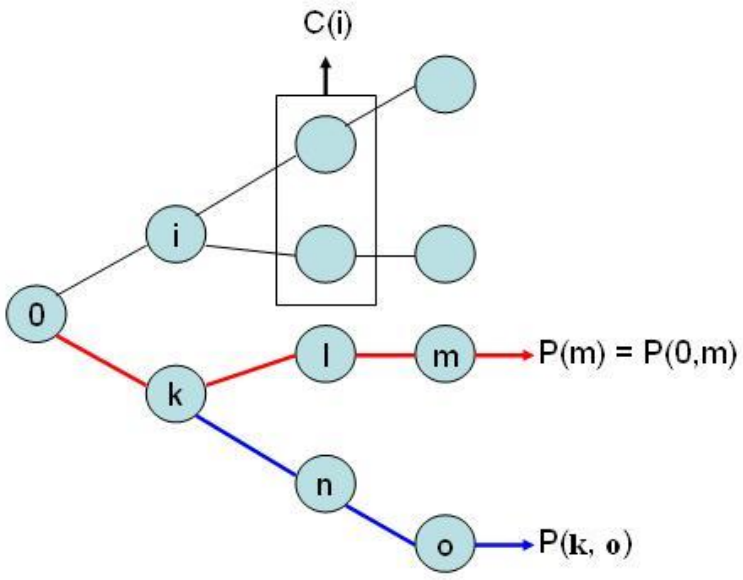

Figure 4.1: Paths on the Scenario Tree

In figure 4.1, we represent the paths on the scenario tree with the related notation.

4.1.4 Mathematical Formulation

$$(M2): \min\ c(o,x,s,y,z) = \sum_{t\in T}(f_t o_t + \beta_t z_t) + \sum_{i\in V}(\alpha_i x_i + \gamma_i y_i + h_i s_i)$$

$$s.t.$$

$$s_0 = 0$$

$$s_{a(i)} + o_{t(i)} + x_i = d_i + s_i \qquad \forall i \in V \qquad (4.1)$$

$$0 \le x_i \le M y_i \qquad \forall i \in V \qquad (4.2)$$

$$0 \le o_t \le M z_t \qquad \forall t \in T \qquad (4.3)$$

$$s_i \ge 0 \qquad \forall i \in V \qquad (4.4)$$

$$y_i \in \{0,1\} \qquad \forall i \in V \qquad (4.5)$$

$$z_t \in \{0,1\} \qquad \forall t \in T. \qquad (4.6)$$

To begin with, the initial inventory is assumed to be zero in SLSE. Then in constraint (4.1), we satisfy the inventory balance; so that for any node $i \in V$, the

incoming inventory of node plus the amount of production at period $t(i)$ plus the extra order amount of node i is equal to the demand of node i plus the outgoing inventory of node i. Constraint (4.2) sets the extra ordering identifier to one when there exists an extra order at node i. Similarly, constraint (4.3) sets production identifier to one if there exists a production at period t.

All decision variables are non-negative. In constraints (4.5) and (4.6), we set variables y_i and z_t as the binary variables of the formulation. Probabilities of each node i at the scenario tree is included in parameters γ_i, α_i and h_i, which are penalty, extra ordering and inventory holding costs of the problem, respectively. Finally, in the objective function of the problem, we minimize the sum of the expected production, extra ordering, inventory holding, setup and penalty costs.

4.1.5 Reformulation

In what follows, as also traditionally done in the literature, we shall eliminate inventory variable s from the mathematical formulation. By doing so, the inventory balance constraint of $M2$ can be replaced by an inequality constraint in the reformulation. This constraint will be used in the validity proofs of the valid inequalities that will be developed in the next section.

For $i \in V$, $s_i = \sum_{j \in P(i)} \left(o_{t(j)} + x_j\right) - d_{0i}$. If we substitute this into formulation M2, the objective function becomes

$$\sum_{t \in T} \left(f_t o_t + \beta_t z_t\right) + \sum_{i \in V} \left(\alpha_i x_i + \gamma_i y_i + h_i \left(\sum_{j \in P(i)} \left(o_{t(j)} + x_j\right) - d_{0i}\right)\right)$$

$$= -\sum_{i \in V} h_i d_{0i} + \sum_{t \in T} \left[\left(f_t + \sum_{j \in N_{t,|T|}} h_j\right) o_t + \beta_t z_t\right] + \sum_{i \in V} \left[\left(\alpha_i + \sum_{j \in V(i)} h_j\right) x_i + \gamma_i y_i\right].$$

Now, we obtain the following equivalent formulation for SLSE:

$$(RM2): \quad \min \quad c(o,x,y,z) = \sum_{t\in T}\left(\bar{f}_t o_t + \beta_t z_t\right) + \sum_{i\in V} \left(\bar{\alpha}_i x_i + \gamma_i y_i\right)$$

$$s.t.$$

$$\sum_{j\in P(i)} \left(o_{t(j)} + x_j\right) \geq d_{0i} \qquad \forall i \in V \qquad (4.7)$$

$$0 \leq x_i \leq M y_i \qquad \forall i \in V \qquad (4.8)$$

$$0 \leq o_t \leq M z_t \qquad \forall t \in T \qquad (4.9)$$

$$y_i \in \{0,1\} \qquad \forall i \in V \qquad (4.10)$$

$$z_t \in \{0,1\} \qquad \forall t \in T \qquad (4.11)$$

where $\bar{f}_t = f_t + \sum_{j\in N_{t|T|}} h_j$ for $t \in T$ and $\bar{\alpha}_i = \alpha_i + \sum_{j\in V(i)} h_j$ for $i \in V$. Constraint (4.7) ensures that the summation of production and extra ordering amounts on a unique path from the root node to a node $i \in V$ must be greater than or equal to the summation of demands on the same path. Constraints from (4.8) to (4.11) are same with $M2$. In the objective function, we minimize the sum of the expected production, extra ordering, setup and penalty costs with updated parameters.

Throughout this chapter we will use the $RM2$ formulation for the $SLSE$ problem. The feasible solutions of the problem will be denoted by the set X_{SLSE}.

4.2 Valid Inequalities

In this section, we develop two families of valid inequalities for the SLSE problem. First one is the (ℓ, S, L) inequality which resembles the (ℓ, S) inequality of Guan et al. [4] in the sense that both inequalities are defined on a unique path between the root node and a node $\ell \in V$ of the given scenario tree. In addition, both inequalities are extensions of general (ℓ, S) inequalities that are used for deterministic lot sizing problems [2].

Our second valid inequality is called (Q, S_Q, L_Q) inequality. The (Q, S_Q, L_Q) inequalities are general cases of (ℓ, S, L) inequalities so they can be used not

only for a unique path between the root node and a node in V but also for a subtree of the given scenario tree $\ddot{T}$. The Q parameter represents the leaf nodes of the selected subtree $\ddot{T}_Q$ and we can reduce (Q, S_Q, L_Q) inequality to (ℓ, S, L) inequality when $|Q| = 1$.

4.2.1 The (ℓ, S, L) inequalities for SLSE

Theorem 2 *Given $\ell \in V$, $S \subseteq P(\ell)$, $L \subseteq P(\ell)$ and $\bar{S} = P(\ell)\backslash S$, $\bar{L} = P(\ell)\backslash L$, the (ℓ, S, L) inequality*

$$\sum_{i \in S} x_i + \sum_{i \in \bar{S}} d_{i\ell} y_i + \sum_{j \in L} o_{t(j)} + \sum_{j \in \bar{L}} d_{j\ell} z_{t(j)} \geq d_{0\ell}$$

is valid for X_{SLSE}.

Proof. Similar to the proof of Theorem 1 in Guan et al. [4].
Let $k = argmin\left\{t(i) : i \in \bar{S}, y_i = 1\right\}$ and $n = argmin\left\{t(j) : j \in \bar{L}, z_{t(j)} = 1\right\}$. For a feasible solution $(o, x, z, y) \in X_{SLSE}$, there exist four possible cases:

1) there exits $i \in \bar{S}$ such that $y_i = 1$ and there exits $j \in \bar{L}$ such that $z_{t(j)} = 1$,
2) $y_i = 0$ for all $i \in \bar{S}$ and there exits $j \in \bar{L}$ such that $z_{t(j)} = 1$,
3) there exits $i \in \bar{S}$ such that $y_i = 1$ and $z_{t(j)} = 0$ for all $j \in \bar{L}$,
4) $y_i = 0$ for all $i \in \bar{S}$ and $z_{t(j)} = 0$ for all $j \in \bar{L}$.

Case 1): Let $m = min\{k, n\}$. Then $y_i = 0$ for all $i \in \bar{S} \cap P(a(m))$ and $z_{t(j)} = 0$ for all $j \in \bar{L} \cap P(a(m))$. This implies $x_i = 0$ for all $i \in \bar{S} \cap P(a(m))$ and $o_{t(j)} = 0$ for all $j \in \bar{L} \cap P(a(m))$. Thus

$$\sum_{i \in S} x_i + \sum_{i \in \bar{S}} d_{i\ell} y_i + \sum_{j \in L} o_{t(j)} + \sum_{j \in \bar{L}} d_{j\ell} z_{t(j)} \ \geq \ \sum_{l \in P(a(m))} (x_l + o_{t(l)}) + d_{m\ell}$$

is satisfied. From constraint (4.7), we have $\sum_{i \in P(a(m))}(x_i + o_{t(i)}) \geq d_{0a(m)}$ and it implies

$$\sum_{i \in P(a(m))} (x_i + o_{t(i)}) + d_{m\ell} \ \geq \ d_{0\ell}.$$

Case 2): As $y_i = 0$ for all $i \in \bar{S}$, we have $x_i = 0$ for all $i \in \bar{S}$. Also $z_{t(j)} = 0$ and $o_{t(j)} = 0$ for all $j \in \bar{L} \cap P(a(n))$. Thus

$$\sum_{i \in S} x_i + \sum_{i \in \bar{S}} d_{i\ell} y_i + \sum_{j \in L} o_{t(j)} + \sum_{j \in \bar{L}} d_{j\ell} z_{t(j)} \;\geq\; \sum_{l \in P(a(n))} (x_l + o_{t(l)}) + d_{n\ell}.$$

is satisfied. From constraint (4.7), we have $\sum_{i \in P(a(n))}(x_i + o_{t(i)}) \geq d_{0a(n)}$ and it implies

$$\sum_{i \in P(a(n))} (x_i + o_{t(i)}) + d_{n\ell} \;\geq\; d_{0\ell}.$$

Case 3): As $z_{t(j)} = 0$ for all $j \in \bar{L}$, it implies $o_{t(j)} = 0$ for all $j \in \bar{L}$. In addition, $y_i = 0$ and $x_i = 0$ for all $i \in \bar{S} \cap P(a(k))$. Thus

$$\sum_{i \in S} x_i + \sum_{i \in \bar{S}} d_{i\ell} y_i + \sum_{j \in L} o_{t(j)} + \sum_{j \in \bar{L}} d_{j\ell} z_{t(j)} \;\geq\; \sum_{l \in P(a(k))} (x_l + o_{t(l)}) + d_{k\ell}$$

is satisfied. From constraint (4.7), we have $\sum_{i \in P(a(k))}(x_i + o_{t(i)}) \geq d_{0a(k)}$ and it implies

$$\sum_{i \in P(a(k))} (x_i + o_{t(i)}) + d_{k\ell} \;\geq\; d_{0\ell}.$$

Case 4): $y_i = 0$ for all $i \in \bar{S}$ and $z_{t(j)} = 0$ for all $j \in \bar{L}$. This implies $x_i = 0$ for all $i \in \bar{S}$ and $o_{t(j)} = 0$ for all $j \in \bar{L}$, which also implies $\sum_{i \in S} x_i = \sum_{i \in P(\ell)} x_i$ and $\sum_{j \in L} o_{t(j)} = \sum_{j \in P(\ell)} o_{t(j)}$, since $S \cup \bar{S} = P(\ell)$ and $L \cup \bar{L} = P(\ell)$ are also satisfied by definition. Thus

$$\sum_{i \in S} x_i + \sum_{i \in \bar{S}} d_{i\ell} y_i + \sum_{j \in L} o_{t(j)} + \sum_{j \in \bar{L}} d_{j\ell} z_{t(j)} \;\geq\; \sum_{l \in P(\ell)} (x_l + o_{t(l)}).$$

is satisfied. From constraint (4.7),

$$\sum_{i \in P(\ell)} (x_i + o_{t(i)}) \;\geq\; d_{0\ell}$$

is also satisfied.

Therefore, (ℓ, S, L) inequality is valid for X_{SLSE}.

□

4.2.2 The (Q, S_Q, L_Q) inequalities for SLSE

In this section, we derive the (Q, S_Q, L_Q) inequalities, which are generalizations of (ℓ, L, S) inequalities. Let $Q \subset (V \backslash \{0\})$ and $V_Q = \cup_{i \in Q} P(i)$. Q is the set of leaf nodes, V_Q corresponds to the nodes and T_Q represents the periods of the subtree $\ddot{T}_Q$. So that we can represent the subtree with leaf nodes Q as $\ddot{T}_Q = \{V_Q, T_Q\}$. In addition, $V_Q(i) = V(i) \cap V_Q$ where $V(i)$ is the set of descendants of node i and $Q(i) = V_Q(i) \cap Q$. On the other hand, N_t represents the set of nodes on period t at our initial scenario tree $\ddot{T} = \{V, T\}$.

Guan et al. [4] define the following functions that will be used in (Q, S_Q, L_Q) inequalities. For $i \in V_Q$,

$$M_Q(i) = \max \{d_{ij} : j \in Q(i)\} \tag{4.12}$$

$$\overline{D}_Q(i) = \max \{d_{0j} : j \in Q(i)\} \tag{4.13}$$

$$\tilde{D}_Q(i) = \begin{cases} 0, \text{ if } \left\{j : j \in Q \backslash Q(i) \text{ such that } d_{0j} \leq \overline{D}_Q(i)\right\} = \emptyset \\ \max \left\{d_{0j} : j \in Q \backslash Q(i) \text{ such that } d_{0j} \leq \overline{D}_Q(i)\right\}, \text{ otherwise} \end{cases} \tag{4.14}$$

$$\Delta_Q(i) = \min \left\{\overline{D}_Q(i) - \tilde{D}_Q(i), M_Q(i)\right\}. \tag{4.15}$$

In addition, we define:

$$K_Q(t) = \max \{M_Q(j) : j \in N_t \cap V_Q\} \tag{4.16}$$

for $t \in T_Q$.

Similar to the study of Guan et al. [4], the subset $Q \subset (V \setminus \{0\})$ satisfies the following properties:

P1) If $i, j \in Q$ then $d_{0i} \neq d_{0j}$.

P2) If $i, j \in Q$ then $i \notin P(j)$ and $j \notin P(i)$.

(P1) and (P2) allow us to index nodes in Q in such a way that $d_{01} < d_{02} < ... < d_{0|Q|}$.

P3) Given any node $k \in V_Q$ and nodes $i, j \in Q$ such that $i < j$ and $i, j \in Q(k)$, then $\{i, i+1, ..., j-1, j\} \subseteq Q(k)$.

As it is stated by Guan et al. [4], given $k \in Q$, let $Q_k = \{1, 2, ..., k-1, k\}$ and $\ddot{T}_{Q_k}$ be the subtree of $\ddot{T}_Q$ with leaf nodes Q_k. Since Q satisfies (P1)-(P3) then Q_k satisfies these properties for $k = 1, ..., |Q|$. Now, suppose there exists a node

$j^* \in V_{Q_k}$ such that $j^* \in P(k)$ and $\tilde{D}_{Q_k} > 0$. Then there exists $r^* \in Q_k$ such that $\tilde{D}_{Q_k} = d_{0r^*}$ and clearly $1 \leq r^* \leq k$.

Proposition 2 *$K_{Q_k}(t) \geq K_{Q_{r^*}}(t)$ for all $t \in T_{Q_k}$ such that $k \in Q$, $r^* \in Q_k$ and $1 \leq r^* \leq k$.*

Proof.
By definition $Q_{r^*} \subseteq Q_k$ since $1 \leq r^* \leq k$ and it implies $V_{Q_{r^*}} \subseteq V_{Q_k}$. Then $(N_t \cap V_{Q_{r^*}}) \subseteq (N_t \cap V_{Q_k})$ is also satisfied for $t \in T$. Thus,

$$K_{Q_{r^*}}(t) = \max \{ \max \{d_{ij} : j \in Q_{r^*}(i)\} : i \in (N_t \cap V_{Q_{r^*}})\} \leq$$
$$K_{Q_k}(t) = \max \{ \max \{d_{ij} : j \in Q_k(i)\} : i \in (N_t \cap V_{Q_k})\}$$

is satisfied for all $t \in T$.

□

Theorem 3 *Given any $Q \subseteq V$ and any two subsets $S_Q \subseteq V_Q$, $L_Q \subseteq T_Q$, the inequality*

$$\sum_{i \in S_Q} x_i + \sum_{i \in \bar{S}_Q} \Delta_Q(i) y_i + \sum_{t \in L_Q} o_t + \sum_{t \in \bar{L}_Q} K_Q(t) z_t \geq M_Q(0)$$

where $\bar{S}_Q = V_Q \backslash S_Q$ and $\bar{L}_Q = T_Q \backslash L_Q$ is called a (Q, S_Q, L_Q) inequality and it is valid for X_{SLSE}.

Proof.
We will show by induction that (Q_k, S_{Q_k}, L_{Q_k}) inequality is valid for $k = \{1, ..., |Q|\}$, where the tree $\ddot{T}_{Q_k} = \{V_{Q_k}, T_{Q_k}\}$ with leaf nodes Q_k is the subtree of $\ddot{T}_Q$.
The base case (k=1): Let $\bar{p}(t) = min\{i \in N_t \cap Q_1 : t(i) = t\}$. Note that $\overline{D}_{Q_1}(i) = d_{01}$, $\tilde{D}_{Q_1}(i) = 0$, $M_{Q_1}(i) = d_{i1}$ and $K_{Q_1}(t) = d_{\bar{p}(t)1}$ for all $t \in T_{Q_1}$. Given any solution $(x, y, o, z) \in X_{SLSE}$, the left hand side of the (Q_1, S_{Q_1}, L_{Q_1})

inequality is given by

$$\sum_{i\in S_{Q_1}} x_i + \sum_{i\in \bar{S}_{Q_1}} min\left\{d_{01}, d_{i1}\right\} y_i + \sum_{t\in L_{Q_1}} o_t + \sum_{t\in \bar{L}_{Q_1}} min\left\{d_{01}, d_{\bar{p}(t)1}\right\} z_t =$$
$$\sum_{i\in S_{Q_1}} x_i + \sum_{i\in \bar{S}_{Q_1}} d_{i1} y_i + \sum_{t\in L_{Q_1}} o_t + \sum_{i\in \bar{L}_{Q_1}} d_{\bar{p}(t)1} z_t \geq d_{01} = M_{Q_1}(0)$$

It should be noted that $d_{01} \geq d_{\bar{p}(t)1}$. The validity of the inequality comes from the validity of (ℓ, S, L) inequality with $\ell = 1$, $S = S_{Q_1}$ and $L = L_{Q_1}$. $M_{Q_1}(0) = d_{01}$ follows from definition since $Q_1 = \{1\}$.

The inductive step: We assume that for all $k \in \{1, ..., \kappa - 1\}$ such that $\kappa - 1 < |Q|$ and for any given $S_{Q_k} \subseteq V_{Q_k}$ and $L_{Q_k} \subseteq T_{Q_k}$, the (Q_k, S_{Q_k}, L_{Q_k}) inequality

$$\sum_{i\in S_{Q_\kappa}} x_i + \sum_{i\in \bar{S}_{Q_\kappa}} \Delta_{Q_\kappa}(i) y_i + \sum_{t\in L_{Q_\kappa}} o_t + \sum_{t\in \bar{L}_{Q_\kappa}} K_{Q_\kappa}(t) z_t \geq M_{Q_\kappa}(0)$$

is also valid for X_{SLSE}.

Similar to the proof of Theorem 2 by Guan et al. [4]. Let $F_\kappa = \left\{i \in P(\kappa) \cap \bar{S}_{Q_\kappa} : \overline{D}_{Q_\kappa}(i) - \tilde{D}_{Q_\kappa}(i) < M_{Q_\kappa}(i)\right\}$. Given any solution $(x, y, o, z) \in X_{SLSE}$, we consider two cases:

(a) there exists $j^* \in F_\kappa$ such that $y_{j^*} = 1$

(b) $y_j = 0$ for all $j \in F_\kappa$.

Case (a): $\overline{D}_{Q_\kappa}(i) - \tilde{D}_{Q_\kappa}(i) < M_{Q_\kappa}(i)$ implies $\tilde{D}_{Q_\kappa} > 0$. Since $\overline{D}_{Q_\kappa}(j^*) \geq M_{Q_\kappa}(j^*)$, then there exists $r^* \in Q$ such that $\tilde{D}_{Q_\kappa}(j^*) = d_{0r^*}$. Let $S_{Q_{r^*}} = S_{Q_\kappa} \cap V_{Q_{r^*}}$ and $\overline{S}_{Q_{r^*}} = \overline{S}_{Q_\kappa} \cap V_{Q_{r^*}}$. Then the left hand side of the inequality can be rewritten as

$$\sum_{i\in S_{Q_{r^*}}} x_i + \tag{4.17}$$

$$\sum_{i\in S_{Q_\kappa}\setminus S_{Q_{r^*}}} x_i + \tag{4.18}$$

$$\sum_{i\in \bar{S}_{Q_{r^*}}} \Delta_{Q_\kappa}(i) y_i + \tag{4.19}$$

$$\sum_{i\in \bar{S}_{Q_\kappa}\setminus \bar{S}_{Q_{r^*}}} \Delta_{Q_\kappa}(i) y_i \tag{4.20}$$

$$\sum_{i\in L_{Q_\kappa}} o_i + \tag{4.21}$$

$$\sum_{i\in \bar{L}_{Q_\kappa}} K_{Q_\kappa}(i) z_i. \tag{4.22}$$

As it is stated by Guan et al. [4], let $u^* = argmax\{t(i) : i \in V_{Q_{r^*}} \cap P(\kappa)\}$. Expression (4.19) can be divided into two expressions below:

$$\sum_{i \in \bar{S}_{Q_{r^*}} \cap P(u^*)} \Delta_{Q_\kappa}(i) y_i + \tag{4.23}$$

$$\sum_{i \in \bar{S}_{Q_{r^*}} \setminus P(u^*)} \Delta_{Q_\kappa}(i) y_i. \tag{4.24}$$

From Lemma 1 and Lemma 2 of Guan et al. [4] respectively, it follows that

$$(4.23) \geq \sum_{i \in \bar{S}_{Q_{r^*}} \cap P(u^*)} \Delta_{Q_{r^*}}(i) y_i \text{ (From Lemma 1 of Guan et al. [4]) , and}$$

$$(4.24) = \sum_{i \in \bar{S}_{Q_{r^*}} \setminus P(u^*)} \Delta_{Q_{r^*}}(i) y_i \text{ (From Lemma 2 of Guan et al. [4]).}$$

Now we will show that $\sum_{t \in \bar{L}_{Q_\kappa}} K_{Q_\kappa}(t) z_t \geq \sum_{t \in \bar{L}_{Q_{r^*}}} K_{Q_{r^*}}(t) z_t$ and $\sum_{t \in L_{Q_\kappa}} o_t \geq \sum_{t \in L_{Q_{r^*}}} o_t$ are satisfied for any given set $\bar{L}_{Q_\kappa}$ and L_{Q_κ} such that $\bar{L}_{Q_\kappa} \subseteq T_{Q_\kappa}$ and $L_{Q_\kappa} = T_{Q_\kappa} \setminus \bar{L}_{Q_\kappa}$.

We consider two cases:

(a.1) if $\bar{L}_{Q_\kappa} \subseteq T_{Q_{r*}}$

(a.2) if $\bar{L}_{Q_\kappa} \not\subset T_{Q_{r*}}$.

$T_{Q_{r^*}} \subseteq T_{Q_\kappa}$ is satisfied by definition since $1 \leq r^* \leq \kappa$.

Case (a.1): We let $\bar{L}_{Q_{r^*}} = \bar{L}_{Q_\kappa}$. Then,

$$\sum_{t \in \bar{L}_{Q_\kappa}} K_{Q_\kappa}(t) z_t \geq \sum_{t \in \bar{L}_{Q_{r^*}}} K_{Q_{r^*}}(t) z_t$$

is satisfied by Proposition 2 since all coefficients are non-negative.

$T_{Q_{r^*}} \subseteq T_{Q_\kappa}$ and $\bar{L}_{Q_{r^*}} = \bar{L}_{Q_\kappa}$, which imply $L_{Q_{r^*}} \subseteq L_{Q_\kappa}$. Thus,

$$\sum_{t \in L_{Q_\kappa}} o_t \geq \sum_{t \in L_{Q_{r^*}}} o_t$$

is also satisfied since all variables are non-negative.

Case (a.2): $\bar{L}_{Q_\kappa} \not\subset T_{Q_{r^*}}$ and $T_{Q_{r^*}} \subseteq T_{Q_\kappa}$ imply $T_{Q_{r^*}} \subset T_{Q_\kappa}$. We let $\bar{L}_{Q_{r^*}} = T_{Q_{r^*}} \cap \bar{L}_{Q_\kappa}$. Then,

$$\sum_{t \in \bar{L}_{Q_\kappa}} K_{Q_\kappa}(t) z_t \geq \sum_{t \in \bar{L}_{Q_{r^*}}} K_{Q_{r^*}}(t) z_t$$

is satisfied by Proposition 2, since $\bar{L}_{Q_{r^*}} \subset \bar{L}_{Q_\kappa}$ and coefficients are non-negative.

$L_{Q_{r^*}} \subset L_{Q_\kappa}$ is satisfied since we have $T_{Q_{r^*}} \subset T_{Q_\kappa}$ and $L_{Q_{r^*}} \subset L_{Q_\kappa}$. Thus,

$$\sum_{t \in \bar{L}_{Q_\kappa}} o_t \geq \sum_{t \in \bar{L}_{Q_{r^*}}} o_t$$

is also satisfied since all variables are non-negative.
By (a.1) and (a.2), we can conclude that for any given sets L_{Q_κ} and $\bar{L}_{Q_\kappa}$ we can find $L_{Q_{r^*}}$ and $\bar{L}_{Q_{r^*}}$ such that

$$(4.21) + (4.22) = \sum_{t \in L_{Q_\kappa}} o_t + \sum_{t \in \bar{L}_{Q_\kappa}} K_{Q_\kappa}(t) z_t \geq \sum_{t \in L_{Q_{r^*}}} o_t + \sum_{t \in \bar{L}_{Q_{r^*}}} K_{Q_{r^*}}(t) z_t$$

is satisfied.

From the validity of the $(Q_{r^*}, S_{Q_{r^*}}, L_{Q_{r^*}})$ inequality, we have

$$\sum_{i \in S_{Q_{r^*}}} x_i + \sum_{i \in \bar{S}_{Q_{r^*}}} \Delta_{Q_{r^*}}(i) y_i + \sum_{i \in L_{Q_{r^*}}} o_i + \sum_{i \in \bar{L}_{Q_{r^*}}} K_{Q_{r^*}}(i) z_i \geq M_{Q_{r^*}}(0) = d_{0r^*}.$$

As we have represented in previous results

$$(4.17) + (4.23) + (4.24) + (4.21) + (4.22) \geq$$
$$\sum_{i \in S_{Q_{r^*}}} x_i + \sum_{i \in \bar{S}_{Q_{r^*}}} \Delta_{Q_{r^*}}(i) y_i + \sum_{i \in L_{Q_{r^*}}} o_i + \sum_{i \in \bar{L}_{Q_{r^*}}} K_{Q_{r^*}}(i) z_i$$

is satisfied. Thus,

$$(4.17) + (4.23) + (4.24) + (4.21) + (4.22) \geq M_{Q_{r^*}}(0) = d_{0r^*}$$

is correct.
Now consider the expression (4.20). As it is stated by Guan et al. [4] the following expression is satisfied since all coefficients are non-negative,

$$(4.20) \geq \overline{D}_{Q_\kappa}(j^*) - \tilde{D}_{Q_\kappa}(j^*) = d_{0\kappa} - d_{0r^*}.$$

Thus,

$$(4.17) + (4.23) + (4.24) + (4.21) + (4.22) + (4.20) \geq d_{0\kappa},$$

which implies

$$(4.17) + (4.23) + (4.24) + (4.21) + (4.22) + (4.20) + (4.18) \geq d_{0\kappa}.$$

Therefore, the (Q_k, S_{Q_k}, L_{Q_k}) inequality is valid.

Case (b): The left hand side of the (Q_k, S_{Q_k}, L_{Q_k}) inequality is

$$\sum_{i \in S_{Q_\kappa}} x_i + \sum_{i \in \bar{S}_{Q_\kappa}} \Delta_{Q_\kappa}(i) y_i + \sum_{t \in L_{Q_\kappa}} o_t + \sum_{t \in \bar{L}_{Q_\kappa}} K_{Q_\kappa}(t) z_t. \tag{4.25}$$

Let $\hat{P}(\kappa) = \{t(i) \in T_{Q_k} : i \in P(\kappa)\}$. Then from the validity of (ℓ, S, L) inequality for $\ell = \kappa$, we have the following expressions:

$$(4.25) \geq \sum_{i \in S_{Q_\kappa} \cap P(\kappa)} x_i + \sum_{i \in \bar{S}_{Q_\kappa} \cap P(\kappa)} \Delta_{Q_\kappa}(i) y_i + \sum_{t \in L_{Q_\kappa} \cap \hat{P}(\kappa)} o_t + \sum_{t \in \bar{L}_{Q_\kappa} \cap \hat{P}(\kappa)} K_{Q_\kappa}(t) z_t =$$

$$\sum_{i \in S_{Q_\kappa} \cap P(\kappa)} x_i + \sum_{i \in \bar{S}_{Q_\kappa} \cap P(\kappa)} d_{i\kappa} y_i + \sum_{t \in L_{Q_\kappa} \cap \hat{P}(\kappa)} o_t + \sum_{t \in \bar{L}_{Q_\kappa} \cap \hat{P}(\kappa)} d_{p^1(t)\kappa} z_t \geq$$

$$d_{0\kappa} = M_{Q_\kappa}(0).$$

Therefore, the (Q_k, S_{Q_k}, L_{Q_k}) inequality is valid.

□

For the validity of (Q, S_Q, L_Q) inequalities, the leaf node set Q satisfies properties (P1), (P2) and (P3). For any Q, the properties (P1) and (P2) can be obtained without loss of generality, by doing necessary updates stated by Guan et al [4]. Finally, the structure of scenario tree according to the defined properties give us practical advantages for defining the parameters of (Q, S_Q, L_Q) inequalities. In the following section you can find the separation algorithm of (Q, S_Q, L_Q) inequalities.

4.3 Separation of Valid Inequalities

In this section, we will develop the separation algorithms of our two families of valid inequalities, (ℓ, S, L) and (Q, S_Q, L_Q). In next chapter, these separation

algorithms will be used with branch-and-cut algorithms to solve the stochastic lot sizing problems.

4.3.1 Separation of (ℓ, S, L) inequalities

Given a path $P(\ell)$ for $\ell \in V$ and a fractional solution (x^*, y^*, o^*, z^*) for SLSE, let

$$S^* = \{i \in P(\ell) : x_i^* \leq d_{i\ell} y_i^*\}, \tag{4.26}$$

$$\bar{S}^* = P(\ell) \backslash S^*, \tag{4.27}$$

$$L^* = \left\{i \in P(\ell) : o_{t(i)}^* \leq d_{i\ell} z_{t(i)}^*\right\}, and \tag{4.28}$$

$$\bar{L}^* = P(\ell) \backslash L^*. \tag{4.29}$$

If $\sum_{i \in S^*} x_i^* + \sum_{i \in \bar{S}^*} d_{i\ell} y_i^* + \sum_{i \in L^*} o_{t(i)}^* + \sum_{i \in \bar{L}^*} d_{i\ell} z_{t(i)}^* < d_{0\ell}$ then (ℓ, S^*, L^*) inequality is violated. Otherwise, the fractional solution satisfies the (ℓ, S^*, L^*) inequality and there are no node sets that violate (ℓ, S, L) inequalities on path $P(\ell)$, since

$$\textbf{min}_{\ S \subseteq P(\ell),\, L \subseteq P(\ell)} \left\{ \sum_{i \in S} x_i^* + \sum_{i \in \bar{S}} d_{i\ell} y_i^* + \sum_{i \in L} o_{t(i)}^* + \sum_{i \in \bar{L}} d_{i\ell} z_{t(i)}^* \right\} =$$
$$\sum_{i \in S^*} x_i^* + \sum_{i \in \bar{S}^*} d_{i\ell} y_i^* + \sum_{i \in L^*} o_{t(i)}^* + \sum_{i \in \bar{L}^*} d_{i\ell} z_{t(i)}^* \geq d_{0\ell}.$$

When node ℓ is decided, separation decision of (S^*, L^*) takes $O(|T|)$ steps, since there can be at most $|T|$ number of nodes on each path $P(\ell)$. In addition, $O(n)$ steps come from the all possible values that $\ell \in V$ can take. So the final complexity of (ℓ, S, L) inequalities is $O(|T|n)$ steps. Thus, (ℓ, S, L) inequalities can be separated in polynomial time.

4.3.2 Separation of (Q, S_Q, L_Q) inequalities

Given a leaf node set Q and a fractional solution (x^*, y^*, o^*, z^*) for SLSE, let

$$S_Q^* = \{i \in V_Q : x_i^* \leq \Delta_Q(i) y_i^*\}, \tag{4.30}$$

$$\bar{S}_Q^* = V_Q \backslash S_Q^*, \tag{4.31}$$

$$L_Q^* = \{t \in T_Q : o_t^* \leq K(t) z_t^*\}, and \tag{4.32}$$

$$\bar{L}_Q^* = T_Q \backslash L_Q^*. \tag{4.33}$$

If $\sum_{i \in S_Q^*} x_i^* + \sum_{i \in \bar{S}_Q^*} \Delta_Q(i) y_i^* + \sum_{t \in L_Q^*} o_t^* + \sum_{t \in \bar{L}_Q^*} K(t) z_t^* < M_Q(0)$, then (Q, S_Q^*, L_Q^*) inequality is violated. Otherwise, fractional solution satisfies the (Q, S_Q^*, L_Q^*) inequality, which implies that there are no node subsets of V_Q and T_Q that violate (Q, S_Q, L_Q) inequalities, since

$$\textbf{min}_{\ S_Q \subseteq V_Q} \left\{ \textbf{min}_{\ L_Q \subseteq T_Q} \left\{ \sum_{i \in S_Q} x_i^* + \sum_{i \in \bar{S}_Q} \Delta_Q(i) y_i^* + \sum_{t \in L_Q} o_t^* + \sum_{t \in \bar{L}_Q} K(t) z_t^* \right\} \right\} =$$
$$\sum_{i \in S_Q^*} x_i^* + \sum_{i \in \bar{S}_Q^*} \Delta_Q(i) y_i^* + \sum_{t \in L_Q^*} o_t^* + \sum_{t \in \bar{L}_Q^*} K(t) z_t^* \geq M_Q(0).$$

For given Q, The complexity of the separation of (Q, S_Q, L_Q) inequality is $O(|V_Q|)$ steps. In other words, when Q is given, violations can be found in $O(|V_Q|)$ steps. The difficult part of the separation is the selection process of Q because by enumeration the (Q, S_Q, L_Q) inequalities can not be separated in polynomial time. Since we do not know a polynomial time algorithm for a general Q, we apply a heuristic algorithm to find violated inequalities for $|Q| \geq 2$.

Similar to Guan et al. [4], the idea of the heuristic algorithm is to add nodes to the initial leaf node set Q such that the right hand side of the inequality $M_Q(0)$ remains the same, while the left hand side is decreasing. When the algorithm finds a violated inequality for $|Q| \geq 2$, it stops. By doing so, we aim to find valid inequalities for larger cardinality Q's (see Algorithm 1 below).

Algorithm 1: Algorithm for $|Q| \geq 2$

Input: fractional solution (x^*, y^*, o^*, z^*)
begin
  for $\ell \in V$ **do**
    Step 0. Set $Q = \{\ell\}$ and $\hat{i} = \ell$.
    Step 1. **if** $|Q| \geq 2$ **then**
      go to Step 2.
    else
      go to Step 3.
    Step 2. Compute S_Q^* and L_Q^* as in 4.30 and 4.32.
      if (Q, S_Q^*, L_Q^*) *is violated* **then**
        stop.
      else
        go to Step 3.
    Step 3. Let $k = argmin\Big\{ d_{0j} : j \in V(a(\hat{i}))\backslash V(\hat{i}),\ Q^{'} = Q \cup \{j\},\ d_{0j} < d_{0\hat{i}}$ and
      $\sum_{i \in S^*_{Q'}} x_i + \sum_{i \in \bar{S}^*_{Q'}} M_Q(i) y_i + \sum_{t \in L^*_{Q'}} o_t + \sum_{t \in \bar{L}^*_{Q'}} K(t) z_t$
      $< \sum_{i \in S_Q} x_i + \sum_{i \in \bar{S}_Q} M_Q(i) y_i + \sum_{t \in L_Q} o_t + \sum_{t \in \bar{L}_Q} K(t) z_t \Big\}$.
      if k *exists* **then**
        go to Step 5.
      else
        go to Step 4.
    Step 4. **if** $\hat{i} \neq 0$ **then**
      $\hat{i} \leftarrow a(\hat{i})$ and **go** to Step 3.
    else
      end for loop.
    Step 5. $Q \leftarrow Q \cup \{k\}$ and $\hat{i} \leftarrow k$ and **go** to Step 1.
end

Chapter 5

Computational Analysis

In this chapter, we provide the computational analysis of the branch-and-cut algorithms in which the separations of (ℓ, S, L) and (Q, S_Q, L_Q) inequalities are done.

The branch-and-cut algorithm is coded in C++ environment with ILOG Concert Technology of CPLEX version 11.2. We have used a 2x2.83 Ghz Intel Xeon CPU and 8 GB memory HP workstation with the operating system Ubuntu 8.04. However, only one-eighth (1024 MB) of the memory is used for our runs. In addition, the time limit on each run is defined as one hour.

5.1 Parameters

We use the scenario tree structure in the formulation of SLSE and the size of the problem coincides with the size of the scenario tree. There are two important parameters of the scenario tree. One of them is the number of levels and the other is number of branches for each non-leaf node. As the number of branch K or the level of the scenario tree T increases, the number of nodes in the scenario tree increases exponentially, so it becomes harder to find an optimal solution for the SLSE problem. We use a scenario tree with $T = 7$ and $K = 2$ in our numerical

analysis.

Throughout our analysis, we use four different cost ratios. Similar to the study of Guan et al. [4], we take setup to holding cost ratio $\beta/h \in \{1750, 3500, 7000\}$, production to holding cost ratio $f/h \in \{50, 100, 200\}$, extra ordering to production cost ratio $\alpha/f \in \{1, 1.5, 2\}$ and penalty to setup cost ratio $\gamma/\beta \in \{2, 5\}$, where h_i is uniform in the interval [0,1]. In addition, nodes in a branch are assigned with equal probabilities and demands are uniform in the interval [0, 1000].

5.2 Performance Measures

We use three performance measures in our analysis. These are the CPU time, the gap between the root node value of the branch-and-cut tree and the optimal value and the number of nodes explored by the branch-and-cut algorithm. In addition, if the given instance of the problem can not be solved to optimality in the allowed time limit, we present the optimality gap of the problem at termination. This gap is presented in parentheses under CPU time data.

There are four branch-and-cut algorithms in our numerical experiments. First algorithm is the default branch-and-cut algorithm of CPLEX that uses the default cuts of CPLEX. Second algorithm uses the default cuts of CPLEX together with (ℓ, S, L) cuts. The remaining two algorithms implement (Q, S_Q, L_Q) cuts with the default cuts of CPLEX, but the former one is with $|Q| = 2$ and the latter one is with $|Q| \geq 2$. In our numerical results, we denote the number of nodes at each of these algorithms as LP-BBNode, LSL-Node, QSL-Node(2) and QSL-Node(Gen.), respectively. The column labeled as CPU gives the running time and the column corresponding to GAP represents the LP relaxation gap of the branch-and-cut algorithm. The total number of cuts added by each algorithm is also presented at columns which are labeled as #cuts. Refer to Tables 5.1 and Table 5.2 for the results.

5.3 Implementation

Our branch-and-cut algorithms that use (ℓ, S, L) or (Q, S_Q, L_Q) inequalities work as follows. At each node of the branch-and-cut tree, first the default cuts of CPLEX such as Gomory fractional, mixed integer rounding, flow and flow path are added. Then the violated user defined cuts are added to the problem until no more additional cuts are found. The Dynamic Search Algorithm that comes with version 11.2 is switched off in the default branch-and-cut algorithm of CPLEX, since it can not be used with user defined cuts.

5.4 Results

From the results, we can conclude that when (ℓ, S, L) and (Q, S_Q, L_Q) inequalities are used with default cuts of CPLEX, they are effective in reducing the number of nodes explored by the default branch-and-cut algorithm of CPLEX. According to this performance measure, the (Q, S_Q, L_Q) inequalities perform better than branch-and-cut algorithms with (ℓ, S, L) inequalities. For instances, which are corresponding to 22^{nd}, 23^{rd} and 24^{th} row of Table 5.1 and 2^{nd} and 11^{th} row of Table 5.2, the branch-and-cut algorithm that use (Q, S_Q, L_Q) inequalities with $|Q| \geq 2$ performs better than (Q, S_Q, L_Q) inequalities with $|Q| = 2$. But for all other instances of Table 5.1 and 5.2, the branch-and-cut algorithms that use (Q, S_Q, L_Q) inequalities with $|Q| = 2$ give the least number of nodes among other algorithms. Last but not least, the relation between the number nodes explored by the branch-and-cut algorithms and the cost ratios is that the number of nodes increases as γ/β and β/h ratios increase and it also increases when α/f and f/h ratios decrease, and the opposite.

For instances which can be solved in less than 10 seconds, the default branch-and-cut algorithm of CPLEX gives the best CPU time results in Table 5.1 and 5.2. On the other hand, for instances which are solved more than 10 seconds, branch-and-cut algorithms with the (ℓ, S, L) inequalities generally give better CPU times with respect to other algorithms in Table 5.1. But in the 9^{th} and 18^{th} row of Table

5.1, (Q, S_Q, L_Q) inequalities with $|Q| \geq 2$ outperforms the CPU times of (ℓ, S, L) inequalities. On the other hand, in Table 5.2 the branch-and-cut algorithms that use (Q, S_Q, L_Q) inequalities with $|Q| \geq 2$ generally give better CPU times with respect to the branch-and-cut algorithm of (ℓ, S, L). According to the results of Table 5.1 and 5.2, we can conclude that (ℓ, S, L) and (Q, S_Q, L_Q) valid inequalities are effective in reducing the run times of algorithms for instances, which can not be solved less than 10 second by the default settings.

The CPU time increases for ascending values of γ/β and β/h ratios and it also increases for descending values of α/f and f/h ratios, and the opposite. For example, among 54 combinations only one instance can not be solved to optimality by our branch-and-cut algorithms in the allowed time limit and its parameters are $(\gamma/\beta = 5, \beta/h = 7000, \alpha/f = 1.5, f/h = 50)$, which are the largest γ/β and β/h ratios and the smallest α/f and f/h ratios.

The branch-and-cut algorithms that implement (ℓ, S, L) and (Q, S_Q, L_Q) inequalities always have smaller gaps with respect to the default branch-and-cut algorithm of CPLEX. For any instances of our experiments, the branch-and-cut algorithm that uses (Q, S_Q, L_Q) inequalities with $|Q| = 2$ has smaller LP relaxation gap with respect to the branch-and-cut algorithm that uses (ℓ, S, L) inequalities. Similarly, the branch-and-cut algorithm that use (Q, S_Q, L_Q) inequalities with $|Q| \geq 2$ has smaller LP relaxation gap with respect to the branch-and-cut algorithm that uses (Q, S_Q, L_Q) inequalities with $|Q| = 2$. In addition, in the same direction number of cuts added by the branch-and-cut algorithm increases.

Finally, the effectiveness of our valid inequalities are observed in reducing the number of nodes and LP relaxation gaps. In short runs, CPLEX has beaten our CPU times. However for instances which can not be solved in less than a minute, our algorithms outperform the CPU times of the default branch-and-cut algorithm of CPLEX.

Table 5.1: T=7, $\gamma/\beta = 2$

			$LP-BB$			ℓSL			$QSL_Q(2)$				$QSL_Q(Gen.)$			
β/ h	α/ f	f/h	Node	CPU	GAP	Node	#cuts	CPU	GAP	Node	#cuts	CPU	GAP	Node	#cuts	CPU	GAP
1750	2.5		979	0.1	%0.05	850	805	0.1	%0.03	276	1293	0.1	%0.03	515	3878	1	%0.02
	2	200	869	0.1	%0.11	504	912	0.1	%0.06	211	1304	0.1	%0.04	410	4363	1	%0.04
	1.5		5461	0.1	%1.3	4709	1482	2.79	%1	3703	1992	3.11	%0.76	4030	4802	8.9	%0.75
	2,5		1766	0.5	%0.09	1668	882	1.2	%0.06	612	4168	4.3	%0.07	631	4288	10.1	%0.07
	2	100	1576	0.6	%0.14	1566	1359	1.2	%0.11	1300	4881	5.8	%0.09	1681	5011	9.6	%0.09
	1.5		39143	1.64	%0.25	16380	1588	10.1	%0.16	16201	5047	14.7	%0.11	16259	5332	24.5	%0.11
	2.5		4686	2.37	%0.32	4467	667	2.63	%0.26	2611	4188	5.6	%0.22	2611	4188	8.2	%0.22
	2	50	55230	14.87	%0.47	28825	1232	11.6	%0.37	14326	4762	17.2	%0.28	14389	4962	21.2	%0.28
	1.5		440427	156.6	%0.44	253748	2277	132.7	%0.34	157290	5016	118.3	%0.3	157290	5016	187	%0.3
3500	2,5		1925	0.2	%0.12	1960	964	2.2	%0.08	1300	4222	2.8	%0.08	1300	4222	3.6	%0.08
	2	200	2637	0.83	%0.17	2059	1010	5.2	%0.12	1311	4356	5.6	%0.1	1311	4359	8.2	%0.1
	1.5		13128	3.9	%0.24	13073	1539	7.2	%0.17	12255	5020	8.1	%0.16	12255	5020	13.4	%0.16
	2.5		6966	1.64	%0.23	4300	1024	2.52	%0.15	2203	4197	2.54	%0.12	2203	4197	5.3	%0.12
	2	100	9800	2,87	%0.31	4449	1257	1.99	%0.22	3225	4803	2.3	%0.17	3225	4803	5.3	%0.17
	1.5		181850	58.2	%0.55	89536	1503	46.6	%0.36	76400	5024	47.1	%0.33	76400	5024	51.2	%0.33
	2.5		13451	4.61	%0.48	12336	1344	3.8	%0.36	11700	4947	4.4	%0.36	11700	4974	7.2	%0.36
	2	50	18918	7.91	%0.78	20017	1652	12.9	%0.58	20018	5016	13	%0.58	20018	5016	14.2	%0.58
	1.5		1011710	337.6	%1	900228	1698	271.8	%0.61	623600	5017	253.4	%0.59	623600	5017	289.3	%0.59
7000	2.5		8193	2.49	%0.27	6064	692	5.2	%0.14	3004	1808	10.2	%0.11	3103	4141	13.4	%0.11
	2	200	9297	2.62	%0.31	6215	809	5.1	%0.18	2532	1972	13.7	%0.19	5672	4788	23.2	%0.19
	1.5		72404	22.3	%0.37	53212	1081	10.1	%0.22	52707	2201	37.3	%0.21	57895	5024	48.6	%0.2
	2.5		60700	18.9	%0.77	16945	981	17.6	%0.45	11190	1786	22.5	%0.45	10630	4443	32.2	%0.44
	2	100	62689	22.7	%0.81	31851	959	18.4	%0.62	30663	2131	31.7	%0.46	24294	4678	58.7	%0.44
	1.5		299941	128.7	%0.86	177096	1162	82.1	%0.72	157730	4723	100.3	%0.65	128517	5017	130.2	%0.63
	2.5		152006	54.6	%1.51	27546	841	31.1	%0.83	20800	4806	42.2	%0.58	20800	4806	53.1	%0.58
	2	50	252100	82.4	%1.31	158588	1180	63	%0.87	92471	5016	103.2	%0.63	92471	5016	123.2	%0.63
	1.5		5461084	2639.2	%1.7	1576101	1339	1889.6	%1.16	797231	5406	3505.7	%0.72	797231	5019	3602.9	%0.72

Table 5.2: T=7, $\gamma/\beta = 5$

			$LP-BB$			ℓSL				$QS_QL_Q(2)$				$QS_QL_Q(Gen.)$			
β/ h	α/ f	f/h	BBNode	CPU	GAP	Node	#cuts	CPU	GAP	Node(2)	#cuts	CPU	GAP	Node	#cuts	CPU	GAP
1750	2.5		6020	1.95	%0.12	4404	895	7.58	%0.09	2111	2229	7.3	%0.06	2299	4017	8.1	%0.06
	2	200	7881	2.21	%0.18	4772	1274	8.81	%0.14	2823	3228	7.6	%0.08	2164	4691	10.3	%0.08
	1.5		21096	9.14	%0.24	19403	1603	15.7	%0.19	14200	5008	12.3	%0.16	14028	5008	20.9	%0.15
	2,5		10056	2.9	%0.25	6426	1103	11.1	%0.21	6311	4404	13.2	%0.18	6426	4404	23	%0.18
	2	100	10326	3.9	%0.36	10072	1202	16.8	%0.23	9703	4948	20.1	%0.21	9769	4948	38.1	%0.20
	1.5		230563	88.4	%0.75	95144	1849	74.6	%0.24	93200	5023	144.4	%0.49	93062	5023	150.3	%0.45
	2.5		181486	47.1	%1.1	52672	1242	131.7	%0.53	50213	4428	128,4	%0.44	60242	4915	149.6	%0.43
	2	50	250191	48.3	%0.92	47061	1683	38.6	%0.77	46312	5017	43.9	%0.71	47360	5017	50.1	%0.69
	1.5		1734688	828.9	%1.1	509562	2772	796.2	%0.98	271711	5018	689.6	%0.76	300122	5211	812.3	%0.75
3500	2.5		8739	2.47	%0.25	5817	1180	6.26	%0.18	5302	4636	6.4	%0.13	5302	4636	9.1	%0.13
	2	200	10697	3.19	%0.36	7335	1397	8.06	%0.3	7305	5016	8.2	%0.29	6374	4767	13.3	%0.26
	1.5		70270	23.17	%0.51	58362	1914	28.6	%0.46	55203	5166	27.2	%0.4	59241	5016	36.2	%0.36
	2.5		18016	4.98	%0.5	8343	1138	5.47	%0.22	5601	4515	5.6	%0.21	5601	4515	8.1	%0.21
	2	100	89146	24.2	%0.69	71204	1336	23.9	%0.66	42831	5020	22.3	%0.53	42831	5020	24	%0.53
	1.5		2954531	1033.8	%0.99	1376828	2892	1000.2	%0.71	645129	5029	981.6	%0.64	645129	5029	991.8	%0.64
	2.5		54297	21.7	%0.95	31583	1214	22.1	%0.89	27901	5018	20.8	%0.64	27901	5018	29.4	%0.64
	2	50	229418	100.6	%3.91	113291	1786	74.5	%1.32	106428	5017	72.5	%1.19	106428	5017	82.9	%1.19
	1.5		4087831	2899.6	%2.2	2028749	3000	2695	%1.88	1391805	5023	2587	%1.6	1391805	5023	2796	%1.6
7000	2.5		20183	5.8	%0.78	7121	812	6.7	%0.4	2711	4226	10.3	%0.23	2711	4226	23.2	%0.23
	2	200	163806	45.2	%0.88	44424	1057	23.2	%0.67	33112	4905	32.1	%0.58	33112	4905	45.2	%0.58
	1.5		1949903	699.7	%1.04	869760	1518	297.1	%0.85	216800	5019	288	%0.73	216800	5019	301.3	%0.73
	2.5		53014	15.7	%1.08	33703	763	12,4	%0.87	24803	4389	14.7	%0.66	24803	4389	23.8	%0.66
	2	100	213378	78.9	%1.58	188000	1385	52.1	%0.98	162675	5016	51.9	%0.93	162675	5016	60	%0.93
	1.5		7009567	2165.9	%2.15	1519401	1707	1387	%1.17	911810	5226	1215.4	%1.03	911810	5226	1382.6	%1.03
	2,5		83775	39.3	%2.39	68419	943	29.9	%1.45	67558	4734	28.2	%1.4	67558	4737	31.2	%1.4
	2	50	810040	298.4	%2.93	403960	1232	118.2	%1.88	397806	5024	109	%1.79	397806	5024	154.1	%1.79
	1,5		7577601	3600	%5.8	2661701	1831	3600	%2.36	1119700	5300	3600	%1.54	1119700	5317	3600	%1.54
				(%0.59)				(%0.27)				(%0.25)				(%0.25)	

Last but not least, we have noticed that the branch-and-cut algorithm which uses (Q, S_Q, L_Q) inequality with $|Q| = 2$ generally has better performance results among other alternatives and this is why, for the SLSE problem, we suggest using (Q, S_Q, L_Q) inequalities with $|Q| = 2$.

Chapter 6

Conclusion

In this thesis, we consider a single item, multi period, finite horizon and uncapacitated lot sizing problems of a manufacturer who faces stochastic demand under supplier monopoly. In this respect, we have studied two stochastic lot sizing problems. Both of our studies include demand uncertainty via scenarios. Scenarios are realizations of demands with their associated probabilities. To represent the mentioned scenario structure we have used a scenario tree. Each node of the scenario tree corresponds to a demand scenario and each level is a period of the finite planning horizon.

Our first problem is named as stochastic lot sizing problem under monopoly (SLS). In this problem we aim to find the period based production plan of a manufacturer under demand uncertainty. The related costs of the problem are production, inventory holding, backlogging and setup. The objective of the problem is to find the minimum sum of the expected costs. To solve the problem, we have used the specific property of SLS, which is called the production path property. According to the production path property, at each decision period $t \in T$ we either produce or do not produce. If production decision is made the amount of production must be equal to the summation of demand from node $i \in N_t$ to some other node $j \in N_{\hat{t}}$ such that $t < \hat{t}$ minus the incoming inventory level of node i from the previous period. Using the result of the production path property, we have developed a polynomial time exact dynamic programming algorithm for the

SLS problem.

Our second problem is the stochastic lot sizing problem with extra ordering under monopoly (SLSE). The SLSE problem is based on the two-stage stochastic production planning. We have two decision factors in SLSE, first one is scenario based extra ordering decision and the second one is period based production decision. First-stage decisions coincide with production decisions and second-stage decisions come up to extra-ordering decisions. The first-stage decisions are given under demand uncertainty then the second-stage decisions are given as corrective actions for the first-stage decisions. This is why, when first-stage decisions are known, the remaining problem can be solved easily as stochastic uncapacitated lot sizing problem with a polynomial time algorithm [5]. Backlogging is excluded in SLSE problem and the related costs of the problem are production, extra ordering, inventory holding, setup of doing production and penalty of extra ordering. The objective of the problem is to balance the extra ordering and production decisions in order to find the minimum sum of the expected costs. For the SLSE problem we have developed two families of valid inequalities which are called (ℓ, S, L) and (Q, S_Q, L_Q). Then we have used these valid inequalities in branch-and-cut algorithms and the effectiveness of the valid inequalities is tested via experimental results.

The SLSE problem is the first study that investigates two-stage stochastic decision making with a stochastic lot sizing problem. In addition, the valid inequalities which we have developed for SLSE are generalizations of well known valid inequalities of the lot sizing literature that are studied before.

In this study, we work on single-item, uncapacitated lot sizing problems. These problems can be extended for multi-item case in the future research. In addition, the uncapacitated structures of resources can also be changed. Bitran and Yanesse [3], show that computational complexity of capacitated lot sizing problem (CLSP) is NP-Hard and it implies that capacitated stochastic lot sizing problem (CSLSP) is also NP-Hard. Because, CLSP is general case of CSLSP, when there exists one scenario at each period of the scenario tree. But, for constant capacities we may find polynomial time algorithms for stochastic lot sizing

problems since the deterministic case of the problem can be solved in polynomial time [10]. So that as future research, we can study stochastic lot sizing problems with constant capacities. Furthermore, we can also introduce capacities for the second stage decisions of the the two-stage stochastic lot sizing problems, so that second-stage extra-ordering decisions can not exceed some period based capacities or percentages of the first-stage production decisions. For this problem, we may develop valid inequalities or heuristic algorithms. In addition it may also be possible to solve this problem in polynomial time but this is also an area of future research. Finally, decomposition techniques for two-stage stochastic lot sizing problems may also be studied and some handy formulations may be found.

Bibliography

[1] S. Ahmed, A. J. King, and G. Parija. A multi-stage stochastic integer programming approach for capacity expansion under uncertainty. *Journal of Global Optimization*, 23:324, 2003.

[2] I. Barany, T. J. V. Roy, and L. A. Wolsey. Strong formulations for multi-item capacitated lot sizing. *Management Science*, 30:1255–1261, 1984.

[3] G. R. Bitran and H. H. Yanesse. Computational-complexity of the capacitated lot size problem. *Management Science*, 28:11741186, 1982.

[4] Y. Guan, S. Ahmed, G. L. Nemhauser, and A. J. Miller. A branch-and-cut algorithm for the stochastic uncapacitated lot-sizing problem. *Mathematical Programming*, 105:55–84, 2006.

[5] Y. Guan and A. J. Miller. Polynomial-time algorithms for stochastic uncapacitated lot-sizing problems. *Operations Research*, 56:11721183, 2008.

[6] N. Halman, D. Klab, J. M. Mostagir, J. Orlin, and D. Simchi-Levi. Shortest route formulation of mixed-model assembly line balancing problem. *European Journal of Operational Research*, 116:194–204, 1999.

[7] F. W. Harris. How many parts to make at once. *The Magazine of Management*, 10:135–136, 1913.

[8] D. Quadt and H. Kuhn. Conceptual framework for lot-sizing and scheduling of flexible flow lines. *International Journal of Production Research*, 43:22912308, 2005.

[9] R. Schultz. Two-stage stochastic integer programming: a survey. *Statistica Neerlandica*, 50:404–416, 1993.

[10] C. P. M. van Hoesel and A. P. M. Wagelmans. A polynomial time algorithm for the economic lot sizing problem with constant capacities. *Management Science*, 42:142–150, 1996.

[11] H. Wagner and T. M. Within. Dynamic version of the economic lot size. *Management Science*, 50:1770–1774, 1958.