A METAHEURISTIC APPROACH FOR MULTIPLE-ITEM ECONOMIC LOT SIZING PROBLEM WITH INVENTORY DEPENDENT DEMAND

A Thesis

by

Duru Balpınarlı

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Industrial Engineering

Özyeğin University
December 2022

A METAHEURISTIC APPROACH FOR MULTIPLE-ITEM ECONOMIC LOT SIZING PROBLEM WITH INVENTORY DEPENDENT DEMAND

Approved by:

Asst. Prof. Mehmet Önal,Advisor
Dept. of Industrial Engineering
Özyeğin University

Asst. Prof. Erinç Albey
Dept. of Industrial Engineering
Özyeğin University

Prof. Dr. Sabri Tankut Atan
Dept. of Industrial Engineering
Bahçeşehir University

Date Approved: 28 December 2022

To my beloved family and friends...

ABSTRACT

In this study, we consider a multiple-item Economic Lot Sizing problem where the demands for items depend on their stock quantities. The objective is to find a production plan such that the resulting stock levels (and hence demands) maximize total profit over a finite planning horizon. The single-item version of this problem has been studied in the literature, and a polynomial time algorithm has been proposed when there are no bounds on production. It has also been proven that the single item version is $\mathcal{NP}$-hard even when there are constant (i.e, time-invariant) finite capacities on production. We extend this capacitated single-item model by considering multiple-item. Since the single-item capacitated version is $\mathcal{NP}$-hard, the multiple-item capacitated version is $\mathcal{NP}$-hard as well. In the context of this research, we propose a Lagrangian Relaxation method to find an initial solution to the problem, and a Tabu Search algorithm to find better solutions. The performance of the proposed metaheuristic model is compared with the performance of a standard commercial software that works on a mixed integer programming formulation of the problem. We show that our metaheuristic algorithm finds better solutions within a predetermined time limit.

ÖZETÇE

Bu projede talebin stoğa bağlı olduğu çok ürünlü ekonomik öbek büyüklüğü belirleme problemini ele alıyoruz. Bu problemde amacımız stok seviyelerinin ve dolayısıyla taleplerin sınırlı bir planlama ufku boyunca toplam geliri maksimize edeceği bir üretim planı oluşturmaktır. Bu problemin tek ürünlü versiyonu literatürde yer almaktır ve üretimde kapasite kısıtı olmadığı durumlarda, polinom zamanlı çözülebilmektedir. Eğer modele kapasite dahil edilirse ve bu kapasitelerin zamanla sabit kaldığı durumlarda, tek ürünlü problemin $\mathcal{NP}$-zor olduğu gösterilmiştir. Biz bu modeli ürün sayısını arttıracak şekilde genişletiyoruz. Eğer tek ürünlü model $\mathcal{NP}$-zor ise çok ürünlü model de $\mathcal{NP}$-zor'dur. Bu araştırma içinde, başlangıç çözümü bulmak için Lagranj gevşetme yöntemini, ve daha iyi sonuçlar bulmak için Tabu Arama algoritmasını öneriyoruz. Önerilen metasezgisel modelin performansı, problemin karma tamsayılı programlama formülasyonu üzerinde çalışan ticari yazılımın performansı ile karşılaştırılır. Metasezgisel algoritmamızın belirlenmiş bir zaman sınırı içinde daha iyi çözümler bulduğunu gösteriyoruz.

ACKNOWLEDGEMENTS

I would like to express my great appreciation to my advisor Asst. Prof. Mehmet Önal for his consistent support, guidance, and patience. To study under his direction is a great honor and privilege. He provided me with the unique opportunity of working in this research area.

Furthermore, I would like to express my thanks to my parents for their endless constant, unwavering, and support. They never stopped believing me and they were always by my side. Also, I would like to give my gratitude to Barış who is my biggest support and my other half.

This research is supported in part by TÜBITAK grant 119M278.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In the retail sector, the main purpose is to increase sales and revenues. There are several ways to achieve this purpose. One of them is increasing sales by increasing visible stock levels. It is observed in many cases that increasing the visible stock levels also increases customer demand for some product categories. [1] proposes that increasing the stock level of a product can increase sales for a variety of reasons such as indicating freshness (for food products), stimulating hidden demand, pointing out a popular product, offering customers reassurance about high service standards, and availability in the future. There is a lot of research on the hypothesis that if the visibility of stock increases, then demand also increases. These studies recommend that retailers should concentrate on visible stock levels while making inventory plans. However, retailers cannot increase stock levels indefinitely due to limited storage capacities. Furthermore, companies in the retail sector, are trying to increase the number of alternatives that they offer to their customers. Along with the increasing number of alternatives to offer, even a bigger struggle begins among products for limited shelf space. Therefore, it is never easy for retail companies to arrange large numbers of products in locations with limited storage space. Even if there is an unlimited shelf space, due to perishability, stock levels of some items can not be increased indefinitely.

In this paper, we consider a multiple-item economic lot sizing problem where the demands of items depend on their stock quantities. Our main aim is to find a production plan to maximize total profit over a finite planning horizon. We assume that

demand is a non-decreasing and convex function of inventory available after production. We assume that there are time-variant production which means that capacities are not constant over time. Obviously, this multiple-item problem is $\mathcal{NP}$-hard. Therefore, we propose a metaheuristic solution method to the problem. In particular, we utilize Lagrangian Relaxation to find an initial solution and the Tabu Search algorithm to reach better solutions. We generate several test instances and compare the performance of our proposed Tabu Search algorithm with the performance of commercial software which works on a mixed integer programming formulation of the problem.

1.1 Previous Work

The economic lot-sizing problems have been studied in the literature since the 1950s when they were first time introduced by [2]. In the single-item ELS model of [2], there are demands for an item over the discrete and finite planning horizon. There are production, set-up, and inventory carrying costs. The aim is to find a minimum-cost production plan to satisfy demands in each discrete time period. This model is the simplest version created on this subject. To this day, it has been extended in several ways. Some of these studies assume bounded production or inventory storage, some incorporate shortages, some incorporate back ordering, some consider re-manufacturing options, and some assume a multi-level structure where the items should follow a series of procedures or should go through several locations before satisfying the demand. For instance, [3] and [4] analyze ELS problem with finite production capacities. They propose a dynamic programming algorithm when capacities are time-invariant, i.e., when capacities are constant over time. [5] and [6] prove that single item economic lot sizing problems are $\mathcal{NP}$-hard when capacities are time-varying. [6] list several $\mathcal{NP}$-hard cases of the ELS with production capacities.

[7] works on an ELS problem where backlogging is allowed in a multi-echelon structure. [8] works on an ELS model with finite production and inventory bounds in each period. In all the ELS models listed above, there is a single item, and demands are predetermined.

The multiple-item ELS problems are widely covered in the literature, as are the single-item ELS problems. [6] work on multiple-item capacitated problems and prove that problem is $\mathcal{NP}$-hard when capacities are non-increasing over time. In accordance with the literature, we say a problem is *capacitated* if there are finite production capacities, and *uncapacitated*, if there are no production capacity restrictions or production capacities are infinite. [9] develop a part-period balancing heuristic method to solve multiple-item multi-level capacitated lot-sizing problems to prevent shortages and inventory overages. [10] works on uncapacitated multiple-item economic lot-sizing problems with re-manufacturing and utilizes an approach that decomposes the multiple-item problem into several single-item problems. [11] works with Lagrangian Relaxation and dynamic programming to solve time-varying demands with setup times. [12] prove that the multiple-item capacitated lot-sizing problem is $\mathcal{NP}$-hard and they suggest applying the Lagrangian algorithm to the capacity constraint.

The majority of ELS problems in the literature assume that demand is given as a parameter and known in advance. However, demand might change with some decisions in a limited planning horizon and this change should be integrated into the production plan. There is limited research on models that assume demand is a function of other decision variables. [13] assumes that demand can change as a function of the price for an item. [14] and [15] work on single-item lot sizing problems where demand is a function of price under time-invariant production capacities. In [13], [14] and [15], the objective function is to maximize profit, not to minimize costs. [16] develop a dynamic programming algorithm to solve a single-item uncapacitated lot sizing problem in a two-echelon structure, where demand is a function of price. In

addition to these studies, [17] develop a branch-and-price algorithm for a multiple-item capacitated ELS problem where demand is affected by the pricing decisions. [13] and [18] also work on single-item uncapacitated ELS problems where demand is a function of price. They propose algorithms that can be implemented in polynomial time.

The first paper that incorporates stock-dependent demand in an ELS framework is [19]. They analyze a single-item economic lot sizing problem where demands are stock-dependent. They refer to their problem as ELSIDD, which is short for Economic Lot Sizing with Inventory Dependent Demand. In [19], the demand at discrete times varies with the amount of inventory under capacity restriction. They propose dynamic programming for the uncapacitated version of the problem. They also prove that the problem becomes $\mathcal{NP}$-hard even under time-invariant production capacities. Their work makes the foundation for the work presented in this thesis, where we analyze the multiple-item capacitated ELSIDD problem. Needless to say, since the single-item capacitated ELSIDD problem is $\mathcal{NP}$-hard, the multiple-item capacitated ELSIDD problem is also $\mathcal{NP}$-hard. This justifies the application of heuristic algorithms to solve this problem. In this thesis, we propose a Lagrangian Relaxation method to find an initial solution, and a Tabu Search algorithm to find better solutions for the multiple-item capacitated version of this problem.

Lagrangian Relaxation method has been utilized for decomposing problems since the 1970s. It is a well-known technique, used to find good lower and upper bounds in some problem classes. [20] has a comprehensive review of the work done about the Lagrangian Relaxation method. Briefly, in the Lagrangian Relaxation method, the constraints that make the problem complicated are removed from the constraint set and added to the objective function with a vector of penalty multipliers, which are also called *Langrangian multipliers*. This process of replacing a constraint is sometimes called the dualization of the constraint. The problem obtained this way is called the

Lagrangian Relaxation of the original problem. For any set of non-negative Lagrange multipliers, the solution to the Lagrangian Relaxation of the problem provides a bound for the original problem. Then, the aim is to find those values of the Lagrange multipliers that provide the tightest bound. The problem of finding optimal dual multipliers is called the *Lagrangian Dual* problem. Lagrangian Dual problem is a convex optimization problem, and hence a *sub-gradient method* can be utilized to determine the optimal values of the Lagrange multipliers. The performance of the sub-gradient method heavily depends on the step lengths and iteration number. [21] discusses some methods to determine the step length. [22] shows graphically that even if the formulation of step length changes, after some iteration number the objective function value will not improve.

Some $\mathcal{NP}$-hard problems consist of polynomially solvable sub-problems that are combined (or tied) with some set of constraints, which make them complicated to solve. When these tying constraints are dualized, the resulting problem is decomposed into several easy-to-solve independent sub-problems. For instance, in multiple-item ELS problems with production capacities, capacity constraints are usually the only tying constraints. If these capacity constraints are dualized, the resulting problem is just a group of independent single-item ELS problems. This special structure of multiple-item ELS problems inspires many researchers to apply the Lagrangian Relaxation method to such problems. [11], [23], [24], [25], and [26] utilize Lagrangian Relaxation to solve ELS problems. The common point between them is that they all try to minimize cost and work with capacitated ELS. Since the capacity constraint is the complicating constraint, in all, they dualize the capacity constraints.

For handling complex combinatorial problems, metaheuristics like Tabu Search (TSA), Simulated Annealing (SA), and Genetic Algorithms (GA) have gained popularity in recent years. In the literature, researchers utilize mostly Genetic, Tabu

Search, Greedy, Simulated Annealing algorithms or hybrid versions of these algorithms in ELS problems. [27] work on multiple-item capacitated ELS problems considering back orders and develop two hybrid heuristics: the GA/SA algorithm and Lagrangian Relaxation/SA algorithm. Since Lagrangian Relaxation provides the tightest upper bound and promising initial solutions, a hybrid of Lagrangian Relaxation/SA algorithms provides better results. Also, [28] has comprehensive research about the Genetic algorithm in the different varieties of economic lot-sizing problems. [29] utilize the Tabu Search algorithm that includes penalty costs on infeasible solutions in capacitated economic lot sizing problems. [30] develop Tabu Search algorithm for economic lot sizing problem re-manufacturing. [31] develop general variable neighborhood search for multiple-item lot-sizing problems with product returns and recovery. In [31], uses classical neighborhood search with shifting the production periods for finding an initial solution then a constructive heuristic is developed. [30] select [31]'s work as a benchmark and [30]'s algorithms give better solution in a less time.

The remainder of this thesis is organized as follows. Since it makes the foundation of our study, we start by summarizing the single-item ELSIDD model of [19] in Section (II). Then, in the same section, we present our multiple-item ELSIDD model. In Section (III), we present the details of the Lagrangian Relaxation method and Tabu Search algorithm for the multiple-item ELSIDD problem. Lastly, in Section (IV), we test the effectiveness of our solution method by comparing Tabu Search algorithm performance with the performance of a commercial solver.

CHAPTER II

DETAILED PROBLEM DESCRIPTION

2.1 ELSIDD with multiple-item

The multiple-item ELSIDD problem is an extension of the single-item Economic Lot Sizing (ELS) problem where the demand for each item is a function of the amount of available stock of that item. The multiple-item ELSIDD is defined over a discrete and finite planning horizon of T periods. At the beginning of each period, a decision is made on which items to produce (or to produce at all). If there is production for an item, the produced amount in that period is added to the stocks. The demand for each item in each period is assumed to be a piecewise linear concave function of the total stock of that item available after production. After the demands for items are realized, which depends on the total amount of items in stock, leftover products are carried over to the following period. There are item-dependent unit production and holding costs, and fixed set-up costs. The total amount that can be produced in each period is limited by the production capacity. The problem is to plan production for multiple-item so that satisfying the resulting demands creates maximum profit. The parameters and decision variables along with the formulation of the multiple-item ELSIDD model is given as follows:

Parameters:

C_t = production capacity in period t

S_{it} = fixed setup cost of production in period t of item i

p_{it} = unit selling price in period t of item i

c_{it} = unit production cost in period t of item i

h_{it} = unit inventory holding cost in period t of item i.

Decision variables:

D_{it} = demand in period t of item i

x_{it} = quantity produced in period t of item i

I_{it} = amount of inventory carried from t to $t+1$ of item i

U_{it} = amount of inventory after production in period t of item i.

Mathematical Model:

$$max \sum_{i=1}^{N} \sum_{t=1}^{T} p_{it} D_{it} + \sum_{i=1}^{N} \sum_{t=1}^{T} (S_{it}\delta(x_{it}) + c_{it}x_{it} + h_{it}I_{it}) \tag{1}$$

subject to (ELSIDD)

$$I_{i,t\text{-}1} + x_{it} = D_{it} + I_{it}, \qquad i = 1, \ldots, N \qquad t = 1, \ldots, T \tag{2}$$

$$U_{i,t} = I_{i,t\text{-}1} + x_{it}, \qquad i = 1, \ldots, N \qquad t = 1, \ldots, T \tag{3}$$

$$D_{it} = g_{it}(U_{it}), \qquad i = 1, \ldots, N \qquad t = 1, \ldots, T \tag{4}$$

$$\sum_{i=1}^{N} x_{it} \leq C_t, \qquad i = 1, \ldots, N \qquad t = 1, \ldots, T \tag{5}$$

$$I_{it}, U_{it}, I_{it}, D_{it} \geq 0, \qquad i = 1, \ldots, N \qquad t = 1, \ldots, T \tag{6}$$

$$I_{i0} = 0, \qquad i = 1, \ldots, N \tag{7}$$

The model above is nothing but the multiple-item version of the model studied in [19], who work on the single-item ELSIDD problem. Although we will restate them here, some further details of the model can also be found in [19]. The initial term in the summation of the objective function stands for the total revenues and the second summation stands for total setup, production, and holding costs. In this formulation, if $x = 0$ then $\delta(x)$ is accepted as 0, else $\delta(x)$ is accepted as 1. Constraints(2) guarantee the balance between inventory, production, and demand. Constraints(3) indicate the quantity of available inventory after production in period t for each item with U_{it}.

Constraints(5) are capacity restrictions for production. Constraint(6) states that non-negativity constraint. Constraints(7) define that there are no initial inventories.

Constraints (4) imply that the demand is a function of U_{it}, the total amount of product available after production in period t of item i. [19] assume that functions $g_{it}()$, $i = 1, \ldots, N$, $t = 1, \ldots, T$, which are also called as *Demand Functions*, are piecewise linear and concave in U_{it}. They assume that after the available stock reaches a certain point, the function flattens out (i.e., its slope becomes zero). This implies that after some point, increasing the stocks do not cause any further increase in demand. In particular, below is how functions $g_{it}()$ are described in [19]:

$$g_{it}(U) = \begin{cases} d_{it}^0, & U = u_{it}^0 = 0 \\ d_{it}^{j\text{-}1} + a_{it}^j (U - u_{it}^{j\text{-}1}), & u_{it}^{j\text{-}1} \leq U \leq u_{it}^j \end{cases}$$

Function g_{it} consists of J_{it}^{max} segments. The j^{th} segment of this function is defined in the interval $[u_{it}^{j\text{-}1}, u_{it}^j)$ and is assumed to have a slope of a_{it}^j. An illustration of such a piecewise linear function, which is extracted from [19] and modified for multiple-item cases, is given in Figure 1. In particular, note that at point u_{it}^b, we have that $D_{it} = U_{it}$. This means that, in any feasible solution, we should have $U_{it} \geq u_{it}^b$. Otherwise, for some t, $D_{it} > U_{it}$ and this implies some demand can not be satisfied on time. This implies that the parts of the demand function below point u_t^b is not needed in any of our calculations. So, for convenience, we carry u_t^0 to the place of (or replace u_t^0 with) u_t^b. In other words, throughout the rest of this thesis, when we say u_t^0, we will be referring to the point where you see u_t^b in Figure 1. This makes indexing easier for us.

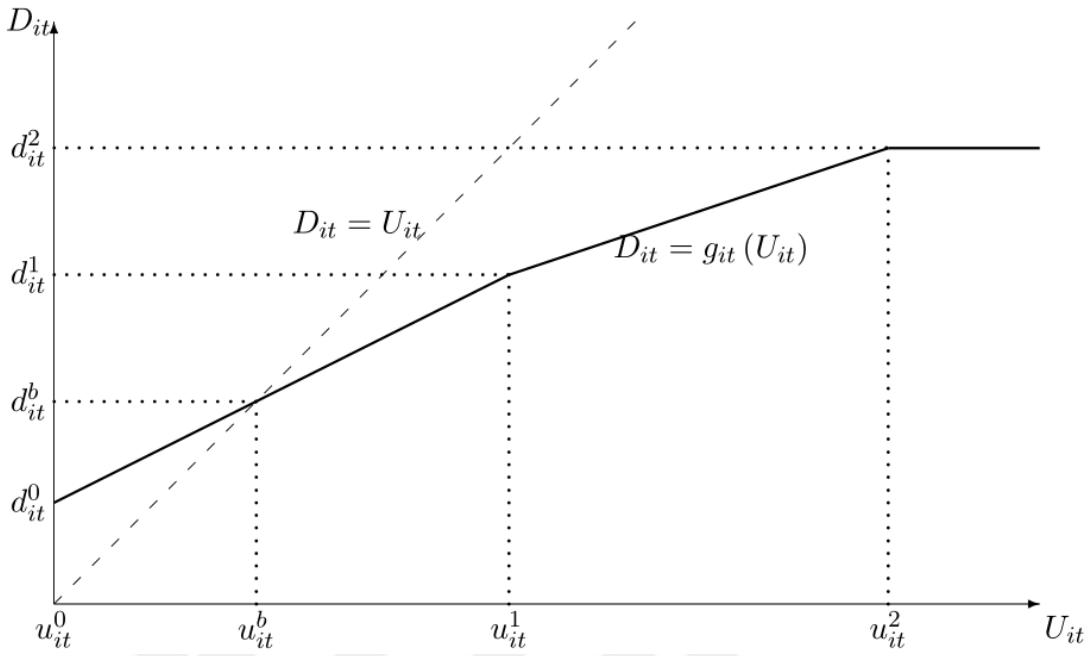


Figure 1: Piecewise linear demand function

With this detailed definition of the demand functions, we replace constraint (4) with constraints (8)-(16) and obtain a reformulation of the multiple-item ELSIDD problem (to use in commercial software) as follows:

$$max \sum_{i=1}^{N} \sum_{t=1}^{T} p_{it} D_{it} + \sum_{i=1}^{N} \sum_{t=1}^{T} (S_{it} y_{it} + c_{it} x_{it} + h_{it} I_{it})$$

subject to (ELSIDD)

$$I_{i,t\text{-}1} + x_{it} = D_{it} + I_{it} \qquad i = 1,\ldots,N, \quad t = 1,\ldots,T$$

$$U_{it} = I_{i,t\text{-}1} + x_{it} \qquad i = 1,\ldots,N \quad t = 1,\ldots,T$$

$$D_{it} = \sum_{j=1}^{J} a_{tj}^{i} d_{tj}^{i} \qquad i = 1,\ldots,N \quad t = 1,\ldots,T, \tag{8}$$

$$U_{it} = \sum_{j=1}^{J} a_{tj}^{i} u_{tj}^{i} \qquad i = 1,\ldots,N \quad t = 1,\ldots,T \tag{9}$$

$$a_{tj}^{i} \leq v_{tj}^{i} + v_{tj+1}^{i} \qquad i = 1,\ldots,N \quad t = 1,\ldots,T \qquad j = 1\ldots, J_{it}^{Jmax} \tag{10}$$

$$\sum_{j=1}^{J} v_{tj}^{i} = 1 \qquad i = 1,\ldots,N \quad t = 1,\ldots,T \tag{11}$$

$$\sum_{j=1}^{J} a_{tj}^{i} = 1 \qquad i = 1,\ldots,N \quad t = 1,\ldots,T \tag{12}$$

$$a_{t,0}^{i} \leq v_{t,1}^{i} \qquad i = 1,\ldots,N \tag{13}$$

$$a_{t,j_{it}^{max}}^{i} \leq v_{t,j_{it}^{max}}^{i}, \qquad i = 1,\ldots,N \tag{14}$$

$$a_{tj}^{i} \geq 0 \qquad i = 1,\ldots,N \quad t = 1,\ldots,T \qquad j = 1\ldots, J_{it}^{Jmax} \tag{15}$$

$$v_{tj}^{i} \in \{0,1\} \qquad i = 1,\ldots,N \quad t = 1,\ldots,T \qquad j = 1\ldots, J_{it}^{Jmax} \tag{16}$$

$$\sum_{i=1}^{N} x_{it} \leq C_t \qquad t = 1,\ldots,T$$

$$I_{it}, U_{it}, I_{it}, D_{it} \geq 0 \qquad i = 1,\ldots,N \quad t = 1,\ldots,T$$

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

$$I_{i0} = 0 \qquad i = 1,\ldots,N$$

Constraints between (8) and (16), are the standard constraints used to model piecewise linear functions. v_{tj}^{i} is a binary decision variable that indicates whether the variable U falls into the j^{th} segment of the g_{it} function in period t for the product i. As long as $u_{it}^{j\text{-}1} \leq U_{it} \leq u_{it}^{j}$ is satisfied, then v_{tj}^{i}=1 ; otherwise $v_{tj}^{i} = 0$. If the variable of U falls on the j^{th} segment of the g_{it} function, using a_{tj-1}^{i} and a_{tj}^{i} decision variables, can be determined where it falls. Also, binary decision variables y_{it} replace

$\delta(x_{it})$ such that if there is positive production in a period t for item i, y_{it} is equal to 1, otherwise, it is equal to 0.

As we mentioned earlier, [19] work on the single-item ELSIDD problem. In Theorem 2 of [19], for the uncapacitated single-item ELSIDD, they prove that in an optimal solution, between any two consecutive production periods t_1 and t_2, there exists a period t $(t_1 \leq t \leq t_2)$ such that the level of U_t is equal to some u_t^j for some $j \in \{0, 1, \ldots, J_{it}^{max}\}$, i.e, the level of U_t reaches one of the borders of the segments of the piecewise linear demand function $g_t()$. Or equivalently, the production in period t_1 raises U_t to a level such that the point (U_t, d_t) is at one of the breakpoints of the graph of the demand function $g_t()$. Given this property, [19] develop a dynamic programming algorithm that can be implemented in $\mathcal{O}(T^4 J^2)$ time. We utilize their algorithms to solve the sub-problems of our Lagrange Relaxation method for finding our initial solution. In addition to this, [19], prove that the problem is $\mathcal{NP}$-hard even production capacities are time-invariant. Since the capacitated version of the multiple-item problem is $\mathcal{NP}$-hard as well, the application of heuristics and metaheuristics is justified to solve this problem. In the following, we are going to implement a metaheuristic algorithm that can effectively solve the multiple-item ELSIDD problem.

CHAPTER III

SOLUTION METHODS

Most of the time, $\mathcal{NP}$-hard problems can not be solved efficiently by using exact algorithms. As a result, people often have a tendency to utilize heuristic algorithms, not to optimally solve such problems, but at least to find a reasonable solution to such problems in a reasonable amount of time. Metaheuristic algorithms are one of those heuristic algorithms that are tailored to search the solution set of a particular problem with the hope of finding a good enough solution in a reasonable amount of time. Multiple-item capacitated ELS problems are also among those $\mathcal{NP}$-hard problems where the application of metaheuristic methods has been proven to be useful in finding good solutions. Because of this, we decided to develop a metaheuristic algorithm for the multiple-item ELSIDD problem.

Local-guided search algorithms are a sub-field of metaheuristic search methods. Roughly, they work as follows. They start with an initial solution, which is also identified as the *current solution*. Initial solutions can be found in different alternative ways such as using a heuristic, random generation, or decomposition algorithms. Then, the solutions in the neighborhood of this current solution are evaluated. The neighborhood of a solution is a set of solutions that can be reached by making iterative small moves from the current solution. The best solution in the neighborhood of the current solution is selected and is defined as (or becomes) the new current solution. The process of evaluating the neighborhood is repeated on the new current solution, and this process goes on. Throughout this procedure, the best solution is stored and updated whenever a better solution is found. With the operations defined in this manner, it is easy to see that the process might get easily stuck in a local optimal

solution. One strategy to avoid such a detriment is creating a memory structure that forbids moves which lead back to local optimal solutions. Even when such strategies are implemented, it is still very likely that the process might get stuck in a local optimal solution. Hence, where one starts the algorithm might affect the final outcome. That is, initial solutions have a great influence on the performance of the local-guided search algorithms. When a predetermined criterion is met, or if it is not possible to improve the solution found in a predetermined number of iterations, the process is terminated and the best solution is presented.

For the multiple-item ELSIDD problem, we propose to use the Tabu Search algorithm(TSA), which is one of the local guided algorithms. To find our initial solution, we use the Lagrangian Relaxation(LR) method, the details of which we describe below.

3.1 Initial solution from the Lagrangian Relaxation Method

As we stated previously, the LR of a problem is obtained by removing a set of constraints and adding them to the objective function by multiplying them with a set of Lagrange multipliers. For any given set of Lagrange multipliers, the solution to LR provides a bound for the original problem. The aim is to find those sets of Lagrange multipliers that result in the tightest bound. As we mentioned earlier, the problem of finding the optimal set of multipliers (i.e. the multipliers that result in the tightest bound) is called the dual problem. The whole process of relaxing and solving the dual is called the LR method.

We would like to note that even with the optimal Lagrange multipliers, the solution of the LR might not be feasible for the original problem. However, it might be potentially close to a good solution for the original problem. A search in the neighborhood for this solution might be fruitful. Therefore, it can be considered as a good initial solution for local-guided search algorithms. In addition to this, if the LR of a

problem is easy to solve, then the application of the LR method becomes appropriate. As a matter of fact, multiple-item ELS problems have a special structure that makes them a good fit for the application of the LR method. In such problems, most of the time, capacity constraints tie a set of single-item (uncapacitated) ELS problems. Hence, if those tying capacity constraints are dualized, the corresponding LR boils down to solving a set of independent single-item ELS *sub-problems*. Most of the time, these sub-problems can be solved in polynomial time, which implies that the LR of multiple-item ELS problems can be solved very easily.

In the multiple-item ELSIDD problem, these tying constraints are Constraints (5), i.e., production capacity constraints. It is easy to see that if these constraints are removed, the problem decomposes into N independent single-item uncapacitated ELSIDD problems, for which [19], propose a polynomial time dynamic programming algorithm. In our LR method, we utilize the algorithm of [19] to solve single-item sub-problems. To solve the dual problem, we use the sub-gradient method. The solution obtained at the end of the sub-gradient method depends on the number of iterations. We pick two solutions from the sub-gradient method. One solution is obtained early in the sub-gradient method, i.e. much before the method converges. The other solution is the solution obtained when the method converges. Using these two distinct solutions, we create a hybrid solution and use this hybrid solution as an initial solution in a TSA. We explain the details of obtaining such a hybrid solution together with the details of our LR method and the TSA below.

3.1.1 Model of Lagrangian Relaxation

Consider the formulation 2 –7 of multiple-item ELSIDD problem. There, constraint (5) represents the production capacities. If this constraint (5) is removed, the whole problem decomposes into N independent single-item ELSIDD problems. For $i = 1, \ldots, N$, let $\mathcal{S}_i$ be the set of feasible solutions for the single item ELSIDD problem

of item i. Then, we can equivalently write the multiple-item ELSIDD formulation as follows:

$$z = max \sum_{i=1}^{N} \sum_{t=1}^{T} p_{it} D_{it} + \sum_{i=1}^{N} \sum_{t=1}^{T} (S_{it} y_{it} + c_{it} x_{it} + h_{it} I_{it})$$

$$\sum_{i=1}^{N} x_{it} \leq C_t \qquad t = 1, \ldots, T$$

$$(\mathbf{x}_i, \mathbf{y}_i, \mathbf{I}_i, \mathbf{D}_i, \mathbf{U}_i) \in \mathcal{S}_i \qquad i = 1, \ldots, N,$$

where $\mathbf{x}_i$, $\mathbf{y}_i$, $\mathbf{I}_i$, $\mathbf{D}_i$ and $\mathbf{U}_i$ are vectors that hold values of x_{it}, y_{it}, I_{it}, D_{it}, and U_{it} for $i = 1, \ldots, N$, $t = 1, \ldots, T$. If we dualize the tying capacity constraints with some vector $\Lambda = (\lambda_1, \ldots, \lambda_T) \geq 0$, we get the LR of the multiple-item ELIDD problem:

$$z(\Lambda) = max \sum_{i=1}^{N} \sum_{t=1}^{T} p_{it} D_{it} + \sum_{i=1}^{N} \sum_{t=1}^{T} (S_{it} y_{it} + c_{it} x_{it} + h_{it} I_{it}) + \sum_{t=1}^{T} \lambda_t (C_t - \sum_{i=1}^{N} x_{it})$$

$$(\mathbf{x}_i, \mathbf{y}_i, \mathbf{I}_i, \mathbf{D}_i, \mathbf{U}_i) \in \mathcal{S}_i \qquad i = 1, \ldots, N.$$

Then, if we refer to (c_{it}-λ_t) as c'_{it}, we can rewrite the objective function and obtain the following final form of the LR of the multiple-item ELSIDD problem:

$$z(\Lambda) = \sum_{t=1}^{T} \lambda_t C_t + max \sum_{i=1}^{N} \sum_{i=1}^{T} p_{it} D_{it} - \sum_{i=1}^{N} \sum_{i=1}^{T} (S_{it} y_{it} + c'_{it} x_{it} + h_{it} I_{it})$$

$$(\mathbf{x}_i, \mathbf{y}_i, \mathbf{I}_i, \mathbf{D}_i, \mathbf{U}_i) \in \mathcal{S}_i \qquad i = 1, \ldots, N.$$

Note that the above LR problem can be solved by solving N independent single-item ELSIDD problems. As mentioned earlier, these single-item problems can be easily solved by using the $\mathcal{O}(T^4 J^2)$ algorithm of [19]. For any $\Lambda \geq 0$, $z(\Lambda) \geq z$ means that the objective function of the LR is greater than the objective function value of the original problem. In other words, $z(\Lambda)$ is an upper bound for z. Therefore, we try

to find the smallest upper bound by solving the following Lagrangian Dual problem (LDP):

$$min\,\{z(\Lambda) : \Lambda \geq 0\}$$

The objective function of the LDP, $z(\Lambda)$ is a convex function of Λ. The sub-gradient optimization method can be used to determine the optimal vector Λ. Below we describe how we implement the sub-gradient method.

3.1.2 Sub-Gradient Optimization Method

The sub-gradient method starts with some vector $\Lambda_0 \geq 0$. We then solve the LR of the multiple-item ELSIDD problem for Λ_0 (i.e. determine $z(\Lambda_0)$ and find optimal values of vectors $\mathbf{x}_i^*$, $\mathbf{y}_i^*$, $\mathbf{I}_i^*$, $\mathbf{D}_i^*$ and $\mathbf{U}_i^*$ for $i = 1, \ldots, N$. Substituting these optimal values, we compute the gradient vector of the objective function $z(\Lambda)$ at $\Lambda = \Lambda_0$. Note that for any Λ and the corresponding optimal values $\mathbf{x}_i^*$, $\mathbf{y}_i^*$, $\mathbf{I}_i^*$, $\mathbf{D}_i^*$ and $\mathbf{U}_i^*$ for $i = 1, \ldots, N$, gradient vector of $z(\Lambda)$ is simply equal to $\nabla z(\Lambda) = \{\frac{\partial z(\Lambda)}{\partial \lambda_1}, \frac{\partial z(\Lambda)}{\partial \lambda_2}, \ldots, \frac{\partial z(\Lambda)}{\partial \lambda_T}\}$, where one can easily see that

$$\frac{\partial z(\Lambda)}{\partial \lambda_t} = C_t - \sum_{i=1}^{N} x_{it}^* \quad \text{for } t = 1, ..., T.$$

Then we move some number of steps in the direction of the gradient vector, where we arrive at another point Λ_1. We repeat the same process with Λ_1, and the process continues for some predetermined number of iterations. Determining the step length, i.e., the number of steps to move in the direction of the gradient vector, is one of the most important decisions in the sub-gradient method. In our algorithm, we utilize the formulations suggested in [32] in order to determine step length. In particular, we implement the sub-gradient algorithm as follows. We start with $\Lambda = \Lambda_0 = \mathbf{0}$. That

is, the initial lagrangian multiplier vector is a $\mathbf{0}$ vector of dimension T. For any given lagrangian multiplier vector Λ_k, we obtain Λ_{k+1} through formulation

$$\Lambda_{k+1} = \Big\{ \Lambda_k - \mu_k (C_1 - \textstyle\sum_{i=1}^{N} x_{i1}^*, C_2 - \sum_{i=1}^{N} x_{i2}^*, \ldots, C_T - \sum_{i=1}^{N} x_{iT}^*)$$

where, μ_k is a positive scalar that represents step length, which is computed via the formula below

$$\mu_k = \frac{\beta_k (Z^* - z(\Lambda_k))}{\left\| (C_1 - \sum_{i=1}^{N} x_{i1}^*, C_2 - \sum_{i=1}^{N} x_{i2}^*, \ldots, C_T - \sum_{i=1}^{N} x_{iT}^*) \right\|^2}. \tag{17}$$

In (17), β_k is a scalar chosen from the interval $[0, 2]$. Z^* is an upper bound for LDP and illustrates the minimum LDP value found so far (at the end of the $(k-1)^{st}$ iteration). In our algorithm, we start with β_0=2. At iteration k, if $(Z^* - z(\Lambda_k))$ is below a certain threshold value, we set $\beta_{k+1} = \beta_k/2$.

In LR, step length has an important role in the convergence speed of the algorithm. There are alternative ways to calculate step lengths in the literature. We did extensive experiments to determine the step length for our algorithm. In our experiments, we observed that choosing a constant step length does not decrease $z(\Lambda)$ as much as possible and the algorithm converges slowly. Similarly, some variable step lengths (e.g. $\xi/\sqrt{k}$, $1/\xi^k$ where ξ is a constant) have a lower convergence speed for the multiple-item ELSIDD problem. However, the formula (17) we use in determining step length, results in the fastest decrease in $z(\Lambda)$ before convergence compared to other formulations.

In the sub-gradient method, continue the calculation of (λ_k) as many as the number of iterations. The solution obtained from LDP for the multiple-item ELSIDD problem, might not be a good solution or even not feasible solution in some cases. Therefore, we develop a heuristic to improve the initial solution in the LR method.

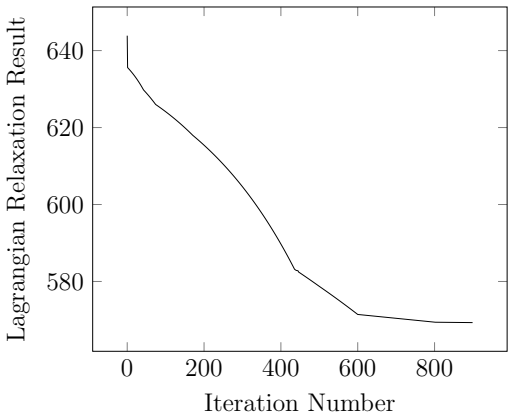
3.1.3 Heuristic Method for Lagrangian Relaxation

The performance of the TSA depends on the quality of the initial solution, which we find by applying the sub-gradient method to the LR of the problem. In turn, the solution found by the sub-gradient algorithm also depends on the number of iterations.

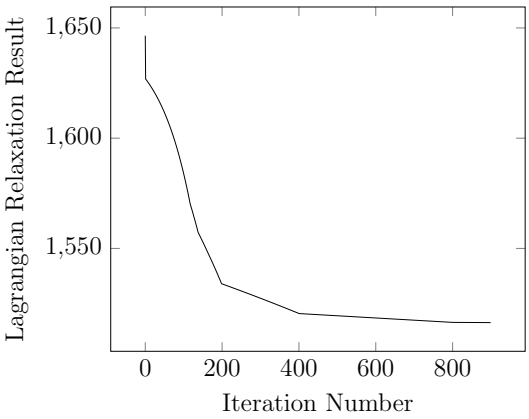
Clearly, it is best to run the sub-gradient algorithm as many iterations as possible to obtain the tightest bound. To see how the (upper) bound changes with the number of iterations in the sub-gradient algorithm, see Figure 2. In the figure, we present four graphs that represent the change in the value of the objective function of LR through iterations for four different synthetic test instances. While segment numbers(J) are the same in all instances, the item numbers and period numbers are different. For all instances, the sub-gradient is run for 900 iterations. The common point in these graphs is that after a certain iteration number, the LR results no longer decrease and starts to converge. We observe that most of the instances converge around 100 and sometimes around 600 iterations.

However, the solution obtained with the optimal Lagrange multipliers might not be the best solution to start the TSA. As a matter of fact, it may not even be feasible for the original problem. Therefore, we propose to follow the following procedure to determine the initial solution for our TSA. We take the $\mathbf{y}$ vectors (i.e., setup vectors), of two different solutions obtained during the sub-gradient algorithm: one from the solution obtained after the 100^{th} iteration, and one from the solution obtained at the end of the 400^{th} iteration. Let's refer to the first vector as $\mathbf{y}_{100}$, and the latter as $\mathbf{y}_{400}$. You can see from the graphs that there is a huge gap between the bounds obtained at the end of the 100^{th} and 400^{th} iterations. So one can expect the majority of the entries of vectors $\mathbf{y}_{100}$ and $\mathbf{y}_{400}$ are the same, but they might also have some number of different entries. So, taking these two vectors, we run (on commercial software) the formulation for the multiple-item ELSIDD problem where we fix the y_{it} values

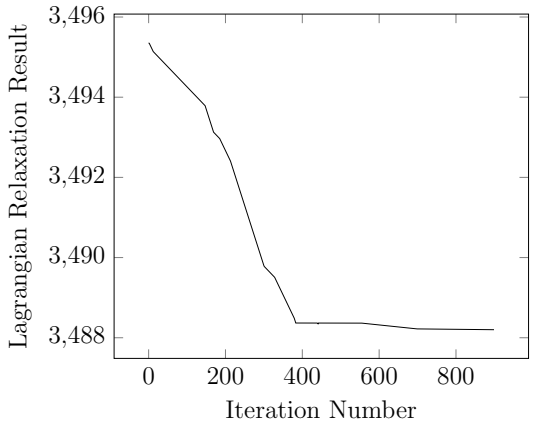
to the corresponding values in the vector $\mathbf{y}_{400}$ (or $\mathbf{y}_{100}$) only if those values are the same in both $\mathbf{y}_{100}$ and $\mathbf{y}_{400}$. We observe that the resulting multiple-item ELSIDD problem, where the majority of y_{it} values are fixed in this manner, can be solved by commercial solvers very easily (in a few seconds) even for large-sized instances (as we show in Chapter IV). We then use the resulting solution as an initial solution in our TSA. As we describe in detail in Section 3.2, we define neighborhood with regard to the setup vector. In other words, our TSA searches for the best setup vector. So, when we say we use a solution as an initial solution, we mean that we use the setup vector of that solution as an initial setup vector to fix in the MIP formulation for the multiple-item ELSIDD problem. Similarly, when we say we *evaluate* a solution, we mean that we solve the MIP formulation for a multiple-item ELSIDD problem with the values in the corresponding setup vector fixed.



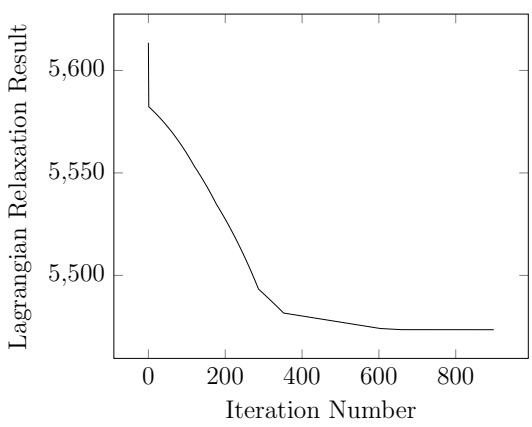
(a) Instance of $T = 40$, $N = 5$, $J = 5$



(b) Instance of $T = 60$, $N = 5$, $J = 5$



(c) Instance of $T = 80$, $N = 5$, $J = 5$



(d) Instance of $T = 80$, $N = 10$, $J = 5$

Figure 2: Lagrangian Relaxation results in each iteration number

3.2 Tabu Search Algorithm (TSA)

Tabu search is a metaheuristic optimization algorithm that uses local search to find good, but not necessarily optimal, solutions to difficult problems. It is often used when it is not possible or practical to find the optimal solution using traditional exact solution methods.

Roughly speaking the algorithm starts with an initial solution, which is also called the current solution. Then it evaluates the solutions in the neighborhood of this solution and assigns the neighboring solution with the best objective function value as the new current solution. The algorithm performs the same search operation on the new current solution and the procedure goes on until a predetermined stopping criterion is reached. During all these processes, the best solution (or a number of high-quality solutions) that has been visited so far is stored, and presented as the final outcome.

In the TSA, as in all local-guided search algorithms, it is important to determine how the solutions will be represented. For instance, for Economic Lot Sizing Problems, usually, a solution is represented by its (binary) setup vector. Once the solution representation is defined, the neighborhood of a solution should be defined. The

neighborhood of a solution is the set of all solutions that can be reached by some small changes in the representation of the current solution. Then, we can say that the set of all possible moves, defines the set of neighboring solutions. By making such small changes to the current solution, called moves, or movements, neighboring solutions are obtained. Consider for instance the binary setup vector to represent a solution to an ELS problem. For such a solution, changing a 0 entry (in the setup vector) to 1 can be considered as a move. It creates a neighboring solution. In the TSA, all possible solutions that can be obtained with all such moves are evaluated. As we stated earlier, the best neighboring solution becomes the new current solution and the process continues in this manner.

In the TSA, we maintain a list of tabu, or forbidden, moves, which are moves that have recently been done and are not allowed to be redone for a certain number of iterations. This helps the algorithm avoid getting stuck in a local optima by forbidding the moves that will take it back to a local optima it has recently visited.

The following sections present components of the introduced TSA for solving multiple-item ELSIDD problems.

3.2.1 Solution Representation

In metaheuristic algorithms, the solution representation is a key component of the overall algorithm design. The solution representation determines how the problem is represented and how the algorithm will search for solutions. A good solution representation should be able to effectively encode the problem being solved and should allow the algorithm to efficiently search for and evaluate potential solutions. In our TSA, we represent the solution by the binary setup vector $\mathbf{y}$. This vector holds binary setup variables y_{it} for $i = 1, \ldots, N$, $t = 1, \ldots, T$. So it is a vector of dimension $N \times T$. We determine the value of such a solution by solving the MIP formulation of the corresponding multiple-item ELSIDD problem where the binary setup variables

are fixed to the values in the binary setup vector $\mathbf{y}$ that represents that particular solution. Recall that in our TSA, the initial solution is obtained at the end of the sub-gradient optimization method. As a matter of fact, we only use the setup vector $\mathbf{y}$ which represents that solution. We then reevaluate the objective function value of that solution by solving the MIP formulation of the multiple-item ELSIDD where we fix y_{it} values to the ones in $\mathbf{y}$.

We would like to remark here that the mathematical model of the multiple-item ELSIDD problem has two sets of binary decision variables: y_{it}, $i = 1, \ldots, N$, $t = 1, \ldots, T$; and v_{tj}^{i}, $i = 1, \ldots, N$, $t = 1, \ldots, T$, $j = 1, \ldots, J_{it}^{max}$. Due to these binary variables, it is not an easy task for commercial solvers to find a solution to the multiple-item ELSIDD problem in a reasonable time. However, we observed that when y_{it} variables are fixed, commercial solvers can solve multiple-item ELSIDD problems in a matter of seconds due to decreasing number of binary variables in the problem. Therefore, we can say that given its representing $\mathbf{y}$ vector, the evaluation of a solution is quite fast.

3.2.2 Local Search Evaluation Procedure and Moves

As we stated previously, the local search algorithm uses a set of small moves (or changes to the current solution) to explore the neighborhood of the current solution. Consider a current solution that is represented by vector $\mathbf{y}$. Every single move results in a change in vector $\mathbf{y}$ and creates a vector $\mathbf{y}'$, which is a different representation for one of the neighboring solutions. The objective function value corresponding to such a representation is computed by solving the formulation of the multiple-item ELSIDD problem by a commercial solver where the y_{it} variables are fixed to the values in vector $\mathbf{y}'$ obtained by such moves that correspond to that particular neighboring solution. Below, we describe the moves we implement to explore the neighborhood of the current solution:

Insert production period: Converting a non-production period to a production period. If the index of i and t on y_{it} vector is equal to 0, then setting that index to 1.

Remove production period: Converting a production period to a non-production period. If the index of i and t on y_{it} vector is equal to 1, then setting that index to 0.

Forward flip: Transferring the production period to the forward periods. If the index of i and t on y_{it} vector is 1, then setting this index to 0 and y_{it+1} index to 1.

Backward flip: Transferring the production period to the previous periods. If the index of i and t on y_{it} vector is 1, then setting this index to 0 and y_{it-1} index to 1.

3.2.3 Tabu List and Tabu Tenure

The tabu list is a key feature of the TSA and is used to prevent the algorithm from getting stuck in local optima and cycling through the same solutions repeatedly. It is a memory structure that stores information about the differences between the current solution and the best solution in its neighborhood set, which includes the period number and item number of change. For instance, the difference between the current solution and the best solution is on item 2 and period 5. These item and period numbers are added to the tabu list before updating every current solution.

The size of the tabu list is an important parameter in the performance of the TSA, as it determines how long the information is stored in the list. The tabu tenure is a parameter that controls the size of the tabu list by specifying how many iterations the information is stored in the list. A larger tabu tenure will allow the algorithm to explore a wider range of the solution space, however, good solutions can be skipped and larger tabu tenure increases the computational complexity of the algorithm, as it requires more memory to store the solutions in the tabu list. On the other hand, a smaller tabu size increases the chance of stuck in local optima. Therefore, we decide to use a different fixed tabu list size throughout the algorithm according to instance sets.

3.2.4 Aspiration and Termination Criteria

The aspiration criterion is a technique used in TSA to allow the algorithm to explore better solutions that may be in the tabu list. It works by allowing the algorithm to accept a move that is in the tabu list if it leads to a solution that is better than the current best solution. This technique can be useful in helping the algorithm escape from local optima and find better solutions.

The algorithm stops after reaching the run-time limit and this is the only termination criterion for the algorithm.

CHAPTER IV

COMPUTATIONAL RESULTS

In this chapter, the analysis of computational results for proposed algorithms is going to be explained for performance evaluation. We generate different test instances to test the performance of the TSA using the same parameter generator structure as [19]. In total, 90 instances are generated and these instances have 40, 60, and 80 periods. After the implementation of algorithms on different instance sets, the comparison tables are prepared to observe whether any improvements are successful or not. We use Cplex as commercial software that works on a mixed integer programming formulation of the problem. Three of the comparison tables interpret the result of the TSA and Cplex. Another table illustrates the comparison of gap results between TSA and Cplex. Both TSA and Cplex have a 15−minute time limit without any gap limitation.

4.1 Generating Test Instances

For this search, test instances were generated from scratch using [19]'s generator structure. In their search, parameters are drawn randomly from the specific uniform distribution boundaries. For making an experiment and testing the performance of algorithms, we generate 90 instances. One-third of these instances have 40 periods ($T = 40$), another one-third have 60 periods ($T = 60$), remaining have 80 periods ($T = 80$). There are $10, 15, 20$ items and $5, 10$ segments. Each table has unique period numbers, however, combinations of items and segment numbers are the same for each table. The parameter setting that uses in comparison tables, is illustrated in Table [1].

Table 1: Summary of parameters

Parameters			
T	40	60	80
N	10	15	20
J	5	10	

4.2 Selecting Upper Bound and Gap Calculation

LR is a method that can be used to find upper bounds for optimization problems. As we mentioned in the LR section, $z(\Lambda)$ always keeps getting smaller until reaching a certain point. By minimizing the $z(\Lambda)$, we can provide a tighter upper bound for the original problem, which can improve the performance of the overall algorithm. We utilize this upper bound when calculating the gap rates of TSA and CPLEX.

The formulations of gap rates are:

$$GapRate_{TSA} = \frac{UpperBound_{LR} - Result_{TSA}}{UpperBound_{LR}}$$

$$GapRate_{Cplex} = \frac{UpperBound_{LR} - Result_{Cplex}}{UpperBound_{LR}}$$

4.3 Experimental Results

In this section, we conduct an experimental evaluation to compare the performance of a metaheuristic algorithm (TSA) with the MIP model which is solved using Cplex. To determine the tabu list size, we perform a series of experiments with different tabu list sizes on a representative set of problem instances, and to compare the performance of the algorithm with different tabu list sizes. Then this helped to identify the tabu list size that provides the best trade-off between performance and computational complexity. According to the number of created current solutions, in trying to prevent stuck in local optima and not allow skipping good solutions, we decided to use different

tabu list sizes for each period number(e.g. 30 for the 40 periods, 20 for the 60 periods, and 15 for 80 periods).

The following tables illustrate and summarize the result of the experiments with 30 instances for each table. The experimental results include a time limit of 15 minutes for both algorithms, and the results are compared across a range of test instances with different numbers of periods, segments, and items. While the period number is unique for each table, segment and item number combinations for each period number are the same. In Table [2], there are minor differences between the performance of TSA and Cplex on the test instances. In some instances especially at the lower item and segment numbers, Cplex has better performance. As the number of items and segments increases, the performance of Cplex decreases. In Table [3] and [4], the difference between results can be observed apparently and the TSA has better performance rather than Cplex. The common point of these tables is that some instances which have 10 item and 5 segments have better results on Cplex. However, as the number of items and segment number increases in each table, TSA has better results and the difference between results also increases. At higher item and segment numbers of each table, Cplex almost never outperformed the TSA, and the number of results where Cplex has better than TSA is also decreasing. In Table [2], [3], and [4], each period, item, and segment number has 5 instances. In Table [5], the average gap rates of the algorithms are illustrated and this table is created by calculating the average gap rates of every 5 instance. The results and gap rates can vary according to segment, item, and period numbers. TSA performs better against Cplex, even if only one parameter is increased between item, segment, and period number. The TSA has lower average gap rates in every group of instances, and also in the general average gap rate. Nevertheless, increasing item, segment, and period numbers decrease the performance of algorithms that cause the increasing gap rates. In the end, the TSA gives better results within a certain time limit when compared to Cplex.

Table 2: Comparison of algorithm results (40 periods (*T=40*))

T	N	J	TSA Results	Cplex Results	LR Upper Bound
40	10	5	6115.28	6225.62	6442.98
			6290.84	6283.53	6494.98
			6803.78	6862.21	7076.65
			6604.94	6611.43	6955.08
			5854.69	5890.75	6050.4
40	10	10	14780.41	14142.32	15204.03
			15161.17	14895.76	15577.32
			14446.61	13973.52	14895.48
			15598.96	15530.49	15987.62
			15566.41	15468.53	15973.16
40	15	5	2200.92	2258.67	2456.3
			2861.23	2828.12	3068.01
			1066.85	1014.73	1101.58
			2490.93	2499.92	2693.51
			2928.93	2940.88	3208.55
40	15	10	32602.84	31512.28	36469.74
			28691.57	26985.05	31075.45
			27598.13	28464.85	31542.47
			25423.46	26498.71	28500.25
			21403.81	20226.74	23066.84
40	20	5	1669.19	1546.95	1882.06
			1357.18	1266.43	1535.58
			2805.87	2425.28	3092.27
			3005.96	2797.03	3451.09
			2211.19	2030.87	2412.56
40	20	10	34874.86	29784.44	39973.33
			33620.42	27972.91	38075.58
			26003.98	24004.86	29086.27
			32106.40	29874.59	37011.77
			31539.54	26205.53	35911.98

Table 3: Comparison of algorithm results (60 periods (T=*60*))

T	N	J	TSA Results	Cplex Results	LR Upper Bound
			2999.03	2907.06	3097.11
			2356.78	2372.06	2478.12
60	10	5	205.12	209.42	220.97
			480.65	481.05	520.13
			2365.64	2388.3	2600.81
			20838.96	19517.21	21466.91
			21855.85	20096.05	22500.35
60	10	10	23786.61	22088.19	24365.03
			23728.95	23260.34	24359.76
			22634.91	21740.46	23150.69
			2992.26	2843.91	3188.34
			5604.48	5785.64	6680.24
60	15	5	5934.32	5408.69	6508.01
			6676.81	6141.47	7267.78
			7099.47	6372.63	7705.07
			31129.29	29949.04	33917.82
			30128.78	29138.04	33629.87
60	15	10	30478.29	28536.9	35847.02
			29876.23	28827.43	32967.1
			31128.56	29470.42	33767.69
			4848.12	4020.14	5241.56
			4843.34	4692.27	5540.94
60	20	5	5883.29	5256.24	6568.52
			4607.23	4471.54	5653.53
			6302.24	5090.22	6587.71
			39139.12	35717.22	46744.11
			35199.82	32911.69	44026.65
60	20	10	39912.88	35259.48	48876.87
			39688.17	35581.27	47198.56
			42555.98	41039.49	49865.12

Table 4: Comparison of algorithm results (80 periods (T=*80*))

T	N	J	TSA Results	Cplex Results	LR Upper Bound
			984.35	993.8	1048.38
			2045.67	2035.3	2112.34
80	10	5	975.2	989.34	1078.65
			1135.78	1141.42	1203.3
			1979.12	1992.64	2104.67
			31782.35	30153.63	35651.56
			31920.06	30233.37	35682.73
80	10	10	32103.01	31920.03	36873.13
			29678.34	26637.57	34951.16
			28238.51	27057.26	34351.53
			1629.12	1554.73	1849.42
			1888.12	1777.82	2082.32
80	15	5	2078.34	1902.42	2333.39
			5081.98	4921.44	6240.32
			3812.45	3767.85	4619.89
			45671.34	41574.82	51911.18
			46201.98	44165.53	54123.73
80	15	10	45486.33	41715.58	54349.32
			36782.32	35862.23	45519.83
			40471.36	35779.69	44474.12
			-2231.47	-2128.24	-1691.61
			-8591.87	-8622.02	-6942.91
80	20	5	-17835.65	-20758.34	-15125.14
			-32829.69	-35360.29	-25792.65
			-7843.23	-8289.68	-6556.89
			39581.43	37203.52	55792.45
			43427.45	41945.42	57417.46
80	20	10	52117.56	53871.44	68778.69
			38451.21	34103.6	58582.56
			52318.75	48918.77	69834.19

Table 5: Comparison of algorithms' average gap results

T	N	J	Average TSA Gaps (%)	Average Cplex Gaps (%)
40	10	5	4.07	3.45
	10	10	2.69	4.71
	15	5	7.31	7.86
	15	10	9.76	11.17
	20	5	10.69	18.34
	20	10	12.10	23.16
Average			7.77	11.45
60	10	5	6.37	6.27
	10	10	2.60	7.94
	15	5	9.41	14.78
	15	10	10.16	14.15
	20	5	10.67	20.45
	20	10	17.05	23.80
Average			9.38	14.56
80	10	5	6.09	5.52
	10	10	13.44	17.83
	15	5	13.64	17.72
	15	10	14.23	20.46
	20	5	24.10	30.15
	20	10	27.42	30.73
Average			16.49	20.40
Total Average			11.21	15.47

CHAPTER V

CONCLUSIONS

In this study, we analyzed the multiple-item ELSIDD problem, an extension of the ELS problem, where the demands for some number of items depend on their stock quantities. Our aim is to find a production plan such that the resulting stock levels (hence demands) maximize total profit over a finite planning horizon. The single-item version of this problem has been analyzed in [19]. They prove that the single-item ELSIDD problem is $\mathcal{NP}$-hard even when production capacities are time-invariant, and they propose a poly-time running algorithm for the uncapacitated problem. Obviously, the multiple-item capacitated ELSIDD problem is $\mathcal{NP}$-hard. Therefore, we implement a Tabu Search metaheuristic algorithm where we use the solution from the Lagrange Relaxation method as an initial solution. We show that the Lagrangian Relaxation of the multiple-item ELSIDD problem can be decomposed into independent uncapacitated single-item ELSIDD problems. Therefore the LR can be easily solved by using the algorithm of [19]. As a result, a good initial solution to start our TSA can be easily obtained.

We then test the performance of our solution procedure on a group of synthetic test instances. For these instances, we compare the quality of the solutions obtained by our algorithm with the solutions obtained by CPLEX (a commercial software), which solves the MIP formulations corresponding to these instances. A summary of our findings is given in the Tables [2], [3] and [4]. Their gap comparison is represented in Table [5]. In total, 90 instances were tested on the performance of TSA and Cplex. Since the Lagrangian upper limit is tighter than Cplex, gap comparisons are calculated by using the Lagrangian upper limit. As observed in the tables, the TSA

gives better performance especially in the larger item, period, and segment numbers when compared to Cplex.

Although the TSA finds better solutions, gap results are not good enough, especially for the higher item, segment, and period numbers. In future research, another algorithm or a metaheuristic can develop for creating better solutions. In addition, this problem only considers production capacities, inventory capacities also can be added to the problem as well as production capacities. As another future research, our research deals with only the production side of the retail sector and the transportation side can be added to this model which makes the problem a multi-level problem.

Our critical contribution to the literature:

- We extend the single-item model of [19] by considering multiple-item and develop a metaheuristic approach to solve this problem.

REFERENCES

[1] A. Balakrishnan, M. S. Pangburn, and E. Stavrulaki, "Stack them high, let 'em fly: Lot-sizing policies when inventories stimulate demand," in *Management Science*, vol. 50, May 2004.

[2] H. M. Wagner and T. M. Whitin, "Dynamic version of the economic lot size model," in *Management Science*, vol. 5, December 1958.

[3] M. Florian and M. Klein, "Deterministic production planning with concave costs and capacity constraints," in *Management Science*, vol. 18, September 1971.

[4] C. P. M. van Hoesel and A. Wagelmans, "An $o(t)^3$ algorithm for the economic lot-sizing problem with constant capacities," in *Management Science*, vol. 42, p. 142–150, 1996.

[5] M. Florian, J. Lenstra, and A. Kan, "Deterministic production planning: algorithms and complexity," in *Management Science*, vol. 26, September 1980.

[6] G. R. Bitran and H. H. Yanasse, "Computational complexity of the capacitated lot sizing problem," in *Management Science*, vol. 18, October 1982.

[7] W. I. Zangwill, "A backlogging model and a multi-echelon model of a dynamic economic lot size production system—a network approach," in *Management Science*, vol. 15, 1969.

[8] S. F. Love, "Bounded production and inventory model with piecewise concave costs," in *Management Science*, vol. 3, 1973.

[9] V. R. Songwut Prakaiwichien, "Solving dynamic multi-product multi-level capacitated lot-sizing problems with modified part period balancing heuristics method," in *International Journal of Applied Engineering Research*, vol. 13, October 2018.

[10] J. O.Cunha, I. Konstantaras, R. A. Melo, and A. Sifaleras, "On multi-item economic lot-sizing with remanufacturing and uncapacitated production," in *Applied Mathematical Modelling*, vol. 50, pp. 772–780, October 2017.

[11] N. Absi and S. Kedad-Sidhoum, "The multi-item capacitated lot-sizing problem with safety stocks and demand shortage costs," in *Computers Operations Research*, vol. 36, pp. 2926–2936, October 2009.

[12] W. Chen and J. Thizy, "Analysis of relaxations for the multi-item capacitated lot-sizing problem," in *Annuals of Operations Research*, vol. 27, pp. 29–72, 1990.

[13] H. Kunreuther and L. Schrage, "Joint pricing and inventory decisions for constant priced items," in *Management Science*, vol. 7, March 1973.

[14] J. Geunes, H. Romeijn, and K. Taaffe, "Requirements planning with pricing and order selection flexibility," in *Operations Research*, vol. 54, March 2006.

[15] J. Geunes, Y. Merzifonluoglu, , and H. Romeijn, "Requirements planning with pricing and order selection flexibility," in *European Journal of Operational Research*, vol. 54, 2009.

[16] M. Onal and H. E. Romeijn, "Two-echelon requirements planning with pricing decisions," in *Journal Of Industrial And Management Optimization*, vol. 5, November 2009.

[17] M. Onal and H. E. Romeijn, "Multi-item capacitated lot-sizing problems with setup times and pricing decisions," in *Wiley InterScience*, December 2009.

[18] V. D. H. W., Wagelmans, and A. P. M., ""a comparison of methods for lot-sizing in a rolling horizon environment," in *Operations Research Letters*, vol. 33, 12005.

[19] Onal and Albey, "Economic lot sizing problem with inventory dependent demand," in *Optimization Letters*, vol. 14, 2020.

[20] M. L. Fisher, "The lagrangian relaxation method for solving integer programming problems," in *Management Science*, vol. 27, December 1981.

[21] L. A. Wolsey, "Lagrangian duality," in *Integer Programming*, pp. 167–181, 1998.

[22] M. Han, "Computational study of the step size parameter of the sub-gradient optimization method," 2013.

[23] Z.-H. Zhang, H. Jiang, and X. Pan, "A lagrangian relaxation based approach for the capacitated lot sizing problem in closed-loop supply chain," in *Int. J. Production Economics*, January 2012.

[24] M. Diaby, H. C. Bahl, M. H. Karwan, and S. Zionts, "A lagrangian relaxation approach for very-large-scale capacitated lot-sizing," in *Management Science*, vol. 38, September 2012.

[25] H. Sural, M. Denizel, and L. N. V. Wassenhove, "Lagrangean relaxation based heuristics for lot sizing with setup times," in *European Journal of Operational Research*, 2007.

[26] S. Ali and O. K. Gupta, "Lagrangian relaxation procedure for the capacitated dynamic lot sizing problem," in *Thirteenth AIMS International Conference*, February 2005.

[27] L. Özdamar and G. Barbarosoğlu, "Hybrid heuristics for the multi-stage capacitated lot sizing and loading problem," in *Journal of the Operational Research Society*, vol. 50, pp. 810–825, 1999.

[28] R. Ajmera, "A review of applications of genetic algorithms in lot sizing," in *Journal of Intelligent Manufacturing*, 2010.

[29] K. D. M. Gopalakrishnanand, J. Bourjolly, and S.Mohan, "A tabu-search heuristic for the capacitated lot-sizing problem with set-up carryover," in *Management Science*, vol. 47, p. 851–863, 2001.

[30] P. Piñeyro and O. Viera, "The economic lot-sizing problem with remanufacturing: analysis and an improved algorithm," in *Journal of Remanufacturing*, vol. 5, 2015.

[31] A. Sifaleras and I. Konstantaras, "General variable neighborhood search for the multi-product dynamic lot sizing problem in closed-loop supply chain," in *Electronic Notes in Discrete Mathematics*, vol. 47, pp. 69–76, 2015.

[32] M. Held, P. Wolfe, and H. P. Crowder, "Validation of sub-gradient optimization," in *Mathematical Programming*, vol. 6, pp. 62–88, 1974.

VITA

Duru Balpınarlı graduated from Marmara Akcan Anatolian High School in 2014. She received her BSc. in Industrial Engineering from Özyeğin University in June 2020. Shortly after, she started her MSc. program in Industrial Engineering at Özyeğin University and conducted research on production planning with Asst. Prof. Mehmet Önal. She worked as a graduate lecturer in Applied Statistics and Operations Research I courses. Her research interests are production planning, inventory management, and optimization. Her research focuses on a meta-heuristic approach for multiple-item economic lot sizing problems.