9-7-2018

Lot-Sizing Problem for a Multi-Item Multi-level Capacitated Batch Production System with Setup Carryover, Emission Control and Backlogging using a Dynamic Program and Decomposition Heuristic

Nusrat Tarin Chowdhury
University of Windsor

Lot-Sizing Problem for a Multi-Item Multi-level Capacitated Batch Production System with Setup Carryover, Emission Control and Backlogging using a Dynamic Program and Decomposition Heuristic

By

Nusrat Tarin Chowdhury

A Dissertation
Submitted to the Faculty of Graduate Studies
through the Department of Mechanical, Automotive and Materials Engineering
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
at the University of Windsor

Windsor, Ontario, Canada

2018

Lot-Sizing Problem for a Multi-Item Multi-level Capacitated Batch Production System with Setup Carryover, Emission Control and Backlogging using a Dynamic Program and Decomposition Heuristic

by

Nusrat Tarin Chowdhury

APPROVED BY:

K. Huang, External Examiner
McMaster University

E. Selvarajah
Odette School of Business

R. Caron
Department of Mathematics and Statistics

G. Zhang
Department of Mechanical, Automotive and Materials Engineering

M. F. Baki, Co- advisor
Odette School of Business

A. Azab, Co-advisor
Department of Mechanical, Automotive and Materials Engineering

August 1, 2018

DECLARATION OF CO-AUTHORSHIP / PREVIOUS PUBLICATION

Co-Authorship Declaration:

I hereby declare that the key ideas, primary contributions, experimental designs, data analysis and interpretation, in the papers mentioned in the table below, were performed by the author, and supervised by Dr. M. Fazle Baki and Dr. Ahmed Azab as co-advisors.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my dissertation, and have obtained written permission from each of the coauthors to include the published, and submitted papers in my thesis. I certify that, with the above qualification, this dissertation, and the research to which it refers, is the product of my own work.

Declaration of Previous Publication:

This dissertation includes 1 original paper that has been previously published, 2 submitted papers for publication in peer reviewed journals.

Publication Title	Publication Status
Nusrat T. Chowdhury, M.F. Baki, A. Azab, "Dynamic Economic Lot-Sizing Problem: A new O(T) Algorithm for the Wagner-Whitin Model", 2018, **Computers and Industrial Engineering**. 117, pp. 6-18.	Published
Nusrat T. Chowdhury, M.F. Baki, A. Azab, "A Modelling and Hybridized Decomposition Approach for a Multi-level Capacitated Lot-Sizing Problem with Set-up Carryover, Backlogging, and Emission Control".	Submitted

Nusrat T. Chowdhury, M.F. Baki, A.Azab, “Lot-Sizing Problem of Maximizing Setup Cost Savings: An application of Maximum Weighted Independent Set Problem”.	Submitted

I declare that, to the best of my knowledge, my dissertation does not infringe upon anyone’s copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my dissertation, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owners to include such materials in my dissertation. I declare that this is a true copy of my dissertation, including any final revisions, as approved by my doctoral committee and the Graduate Studies office, and that this dissertation has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Wagner and Whitin (1958) develop an algorithm to solve the dynamic Economic Lot-Sizing Problem (ELSP), which is widely applied in inventory control, production planning, and capacity planning. The original algorithm runs in $O(T^2)$ time, where T is the number of periods of the problem instance. Afterward few linear-time algorithms have been developed to solve the Wagner-Whitin (WW) lot-sizing problem; examples include the ELSP and equivalent Single Machine Batch-Sizing Problem (SMBSP). This dissertation revisits the algorithms for ELSPs and SMBSPs under WW cost structure, presents a new efficient linear-time algorithm, and compares the developed algorithm against comparable ones in the literature.

The developed algorithm employs both lists and stacks data structure, which is completely a different approach than the rest of the algorithms for ELSPs and SMBSPs. Analysis of the developed algorithm shows that it executes fewer number of basic actions throughout the algorithm and hence it improves the CPU time by a maximum of 51.40% for ELSPs and 29.03% for SMBSPs. It can be concluded that the new algorithm is faster than existing algorithms for both ELSPs and SMBSPs.

Lot-sizing decisions are crucial because these decisions help the manufacturer determine the quantity and time to produce an item with a minimum cost. The efficiency and productivity of a system is completely dependent upon the right choice of lot-sizes. Therefore, developing and improving solution procedures for lot-sizing problems is key. This dissertation addresses the classical Multi-Level Capacitated Lot-Sizing Problem (MLCLSP) and an extension of the MLCLSP with a Setup Carryover, Backlogging and Emission control. An item Dantzig Wolfe (DW) decomposition technique with an embedded Column Generation (CG) procedure is used to solve the problem. The original problem is decomposed into a master problem and a number of

subproblems, which are solved using dynamic programming approach. Since the subproblems are solved independently, the solution of the subproblems often becomes infeasible for the master problem. A multi-step iterative Capacity Allocation (CA) heuristic is used to tackle this infeasibility. A Linear Programming (LP) based improvement procedure is used to refine the solutions obtained from the heuristic method. A comparative study of the proposed heuristic for the first problem (MLCLSP) is conducted and the results demonstrate that the proposed heuristic provide less optimality gap in comparison with that obtained in the literature.

The Setup Carryover Assignment Problem (SCAP), which consists of determining the setup carryover plan of multiple items for a given lot-size over a finite planning horizon is modelled as a problem of finding Maximum Weighted Independent Set (MWIS) in a chain of cliques. The SCAP is formulated using a clique constraint and it is proved that the incidence matrix of the SCAP has totally unimodular structure and the LP relaxation of the proposed SCAP formulation always provides integer optimum solution. Moreover, an alternative proof that the relaxed ILP guarantees integer solution is presented in this dissertation. Thus, the SCAP and the special case of the MWIS in a chain of cliques are solvable in polynomial time.

DEDICATION

I dedicate this dissertation to my parents

ACKNOWLEDGEMENTS

First and foremost, I would like to thank God Almighty for giving me the strength, knowledge, ability and opportunity to undertake this research study and to persevere and complete it satisfactorily. Without his blessings, this achievement would not have been possible.

I would like to dedicate my sincere gratitude to my thesis advisors Dr. Fazle Baki and Dr. Ahmed Azab for giving me this opportunity to conduct research with them. Thanks to their patience and continuous support, my knowledge have been broadened and deepened, I have also acquired the essential attitude toward academic research. Most importantly, I appreciate all their contribution of time and ideas to make my research experience productive and stimulating. Without their guidance and persistent help, this thesis would not have been possible.

I am grateful to my committee members Dr. R. Caron, Dr. G. Zhang and Dr. E. Selvarajah for their comments and suggestions. I have benefitted greatly from their advice. My special thank goes to my external examiner, Dr. K. Huang for his kindness and patience in going through my manuscript.

I owe thanks to a very special person, my husband, Md. Imrul Kaes for his continued support and understanding during my pursuit of Ph.D. degree that made the completion of this dissertation possible. He was always around at times I thought that it is impossible to continue, he helped me to keep things in perspective. I greatly value his contribution and deeply appreciate his belief in me. I also dedicate this Ph.D. thesis to my three lovely sons,

Zawad Kaes, Safwan Kaes, and Mohid Kaes, who are the pride and joy of my life. I love you more than anything and I appreciate all your patience and support during mommy's Ph.D. studies. I consider myself the luckiest in the world to have such a lovely and caring family, standing beside me with their love and unconditional support.

My acknowledgement would be incomplete without thanking the biggest source of my strength, my parents Shirin Akhter Chowdhury and Abdul Mohin Chowdhury, who supported me and helped me throughout my life and during this study. Mom, dad I do not know how to thank you enough for providing me with the opportunity to be where I am today. I would also like to thank my mother in law Arifa Khanam and father in law Liakat Ali Miah for their continuous support and encouragement to achieve my goal.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CA	Capacity Allocation
CG	Column Generation
CLSP	Capacitated Lot-Sizing Problem
DPR	Dynamic Programming Recursion
DW	Dantzig Wolfe
ELSP	Economic Lot-sizing and Scheduling Problem
EOQ	Economic Order Quantity
GHG	Greenhouse Gases
ILP	Integer Linear Programming
LR	Lagrangean Relaxation
MISLCLSP	Multi-Item Single-Level Capacitated Lot-Sizing Problem
MISLULSP	Multi-Item Single-Level Uncapacitated Lot-Sizing Problem
MLCLSP	Multi-Level Capacitated Lot-Sizing Problem
MP	Master Problem
MWIS	Maximum Weighted Independent Set
SCAP	Setup Carryover Assignment Problem
SCBE	Setup Carryover, Backlogging and Emission control
SISLULSP	Single-Item Single-Level Uncapacitated Lot-Sizing Problem
SMBSP	Single Machine Batch-Sizing Problem
SP	Subproblem
WW	Wagner Whitin

CHAPTER 1

INTRODUCTION

Production planning is an activity that considers the best utilization of production resources to meet production requirements and enhance customer satisfaction over a certain period of time. Lot-sizing is one of the production planning problems that involves the decision regarding when to manufacture the production orders and the size of these orders. Lot-sizing or batching is defined by Kuik, Salomon, and van Wassenhove (1994) as "the clustering of items for transportation or manufacturing processing at the same time." Lot-sizing problems arise in production facility whenever the resources need to be set up to produce a new product. Setup tasks can be of many different forms; this can be any of the required cleaning of resources, part fixation, machine adjustments, preheating, inspection, calibration, test runs, and/or tool changes between the different batches. Every setup is associated with a setup cost, which involves the cost to configure a machine for a production run. This also includes the additional workforce needed to set up the equipment, the idle time and production loss during setup operations, and any materials consumed during the setup operations. It is obvious that large lot-sizes can minimize the setup costs and times and maximize the utilization of the production resources. However, this generates inventory as the production is higher than the actual demand. As a result, inventory holding cost occurs to hold the excess products produced until they are used to satisfy the demand.

Thus, the lot-sizing problem is to determine an optimum production or replenishment policy for a manufacturing or inventory system in order to meet market demand with the least possible expenditure. The decision regarding optimum production or replenishment policy is very crucial and hence, a matter of interest for many researchers since the beginning of the twentieth

century when Harris (1913) introduces his well-known and the most fundamental Economic Order Quantity (EOQ) model. In inventory management, the EOQ is the fixed order quantity that minimizes the total holding costs and ordering costs. In this model, demand is assumed to be constant over time. It is quite straightforward to derive the optimal solution using the EOQ model, but because of the rather strong assumptions and simplifications made in development of the model, its practical relevance may be questioned.

A first extension of the EOQ model is the Economic Lot-sizing and Scheduling Problem (ELSP), where multiple items with a constant demand rate share the same production resource with a limited capacity. In the ELSP, the objective is to find a production schedule, which minimizes the total setup and inventory cost. However, a special case of ELSP is addressed by Wagner and Whitin (1958), where discrete periods of time are considered and demand in each of these periods is assumed to be known in advance. They consider a single-item with a dynamic demand that has to be produced on a facility with an unlimited capacity. Wagner and Whitin (1958) develop a forward-recursion dynamic programming algorithm to obtain a minimum total cost inventory management scheme.

1.1 Characteristics of Lot-Sizing Models:

Lot-sizing problems can be classified based on the features taken into account by the model. The complexity of lot-sizing problems depends on these features. The following characteristics are generally used to classify the lot-sizing problem and to decide the complexity of the associated model.

1.1.1 Planning horizon:

The planning horizon is defined as the time interval on which the master production schedule extends into the future. The planning horizon may be either finite or infinite; finite demand is usually accompanied by a dynamic demand whereas that of infinite, is accompanied by static one. Also, the system can be observed continuously or at discrete time points, which then classifies it as either a continuous or discrete-type system. As for the time-period terminology, Lot-sizing problems can also be categorized as big bucket or small bucket problems. Big bucket problems are those where the planning horizon is long enough to produce more than one item in a time period, whereas for small bucket problems, the planning horizon is so short that only one item can be produced in each time period.

1.1.2 Number of levels:

Production systems may be classified as either a single-level or a multi-level system. Single-level systems can be defined as producing the end item directly from the raw materials or the purchased parts through a single operation such as machining, casting, or else. In other words, there is no intermediate subassemblies in the transformation process of raw material to the finished product. For single-level system, product demands are assessed directly from customer orders or market forecasts. Wagner and Whitin (1958), Wagelmans et al. (1992), Aggarwal and Park (1993) and Albers and Brucker (1993) deal with single-level systems. In multi-level systems, there is a parent–child relation among the items. Raw materials are processed using several operations and hence, change to an end products. The output of an operation (level) is input for another. Therefore, the demand at one level depends on the demand for its parents' at the level. This kind of demand is named dependent demand. Multi-level problems are more difficult to solve than single-level

problems. Wu et al. (2011) and Tempelmeier and Derstroff (1996) study the multi-level lot-sizing problem. Multi-level systems are further distinguished by the type of product structure, which includes serial, divergent, assembly and general. The four types of product structures are illustrated in Figure 1.1. In serial product structures, every item has at most one predecessor and one successor. In divergent (assembly) product structures, each item has at most one predecessor (successor), but can have an unlimited number of successors (predecessors). General product structures, which represent multiple assemblies, are the most complex since there is no limit on the number of predecessors or successors. In regards to the process structure, cyclic and acyclic production processes can be distinguished. If the items are produced on a different resource other than their predecessor or successor it is called acyclic system. If some parent items are produced on the same resource as their component, it is called cyclic system.

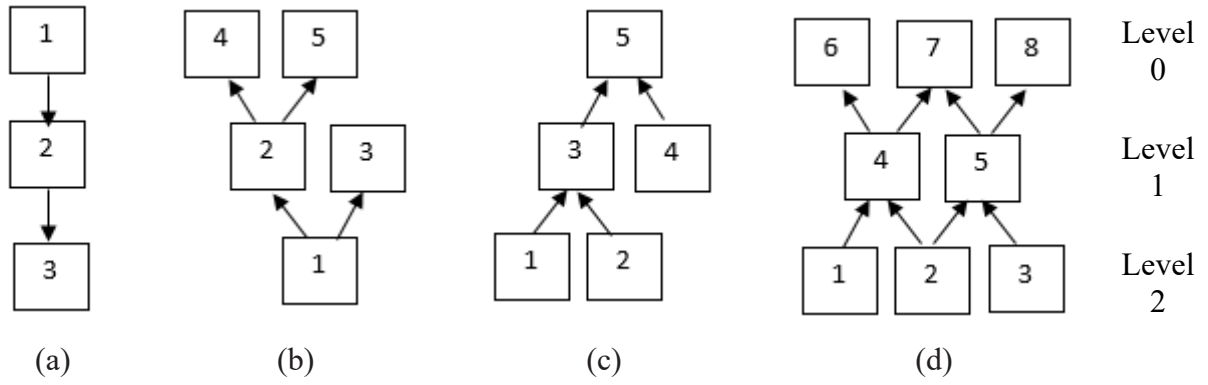


Figure 1.1: (a) serial, (b) divergent, (c) assembly, and (d) general product breakdown structure

1.1.3 Number of products:

Lot-sizing models can be classified as single-item or multi-item lot-sizing problem based on the number of end-items or finished products. In single-item lot-sizing problems, there is only one final item for which the planning activity has to be performed, while in multi-item lot-sizing

problems, there are several end items. The complexity of multi-item problems is much higher than that of single-item problems.

1.1.4 Capacity or resource constraints

Resources or capacities in a production system include manpower, equipment, machines, budget, space, etc. When there is no restriction on resources, the problem is said to be uncapacitated, and when capacity constraints are explicitly stated, the problem is named capacitated. Capacity restriction is important, and directly affects problem complexity.

1.1.5 Demand

The demand for the items to be produced or purchased is used as a parameter in the lot-sizing models. Demand may be classified as deterministic or probabilistic. If the value of the demand is known in advance, it is termed as deterministic, but if it is not known exact with certainty and the values are based on some probabilities, then it is probabilistic. Deterministic demand can be further distinguished as static (demand rate does not change over time) or dynamic (demand rate changes over time). Probabilistic demand can also be further classified as stationary (probability distribution function remains unchanged over time) or non-stationary (probability distribution function varies in time). Furthermore, another important classification of demand is dependent demand and independent demand. In independent demand cases, an item's requirements do not depend on decisions regarding another item's lot size. This kind of demand can be seen in single-level production systems. In multi-level lot-sizing, where there is a parent–child relationship among the items, because the demand at one level depends on that of its parents (pervious level), it is called dependent. A brief classification of demand is illustrated in Figure 1.2. Problems with dynamic and dependent demands are much more complex than problems with static and

independent demands. Also, problems with probabilistic demand are more complex than those with deterministic demand.

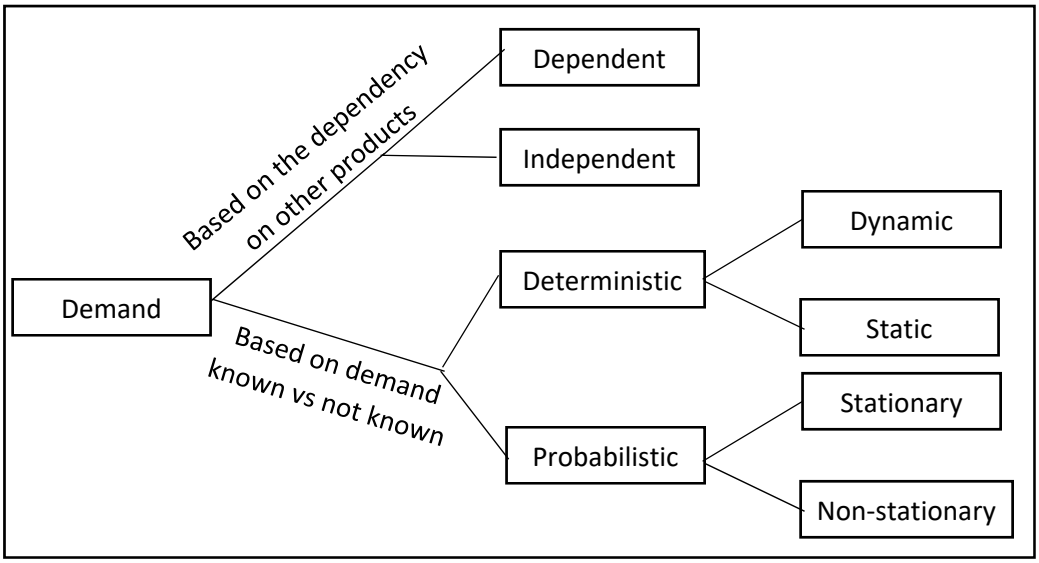


Figure 1.2: Classification of demand

1.1.6 Setup structure

Setup structure is another important characteristic that directly affects problem complexity. Setup costs and/or times, are usually modelled by introducing zero–one variables in the mathematical model of the problem and cause problem solving to be more difficult. Usually, production changeover between different products can incur setup time and hence, a setup cost. The setup time and costs may be constant, product dependent or sequence dependent. If setup time/cost depends solely on the task to be performed, regardless of its preceding task, it is called sequence independent. On the other hand, in the sequence dependent type, setup time depends on both the task and its preceding task (Allahverdi & Soroush, 2008). Other considered characteristics of setups are setup carryover and setup crossover. If same item is produced in two consecutive periods, machine setup state for that item can be fully maintained over periods; this

is denoted as setup carryover (Briskorn, 2006). More specifically, setup carryover permits a setup state to be conserved between two consecutive periods. If the machine is being set up and the setup procedure itself crosses over period boundaries; i.e., the incomplete setup state of the machine is preserved between periods, it is called setup crossover.

1.1.7 Inventory shortage

Inventory shortage is another characteristic, which affects the modelling complexity of the lot-sizing problems. If shortage is allowed, it means that it is possible to satisfy the demand of the current period in future periods (backlogging case), or it may be allowable for demand not to be satisfied at all (lost sale case). The combination of backlogging and lost sales is also possible. Wee (1999) develops a deterministic inventory model based on a Weibull distribution by integrating the backlogging and lost sales case. Inventory shortage generally introduces a penalty cost in the objective function. Problems with shortage are more difficult to solve than those without.

1.2 Variants of lot-sizing and scheduling problems

1.2.1 Single-Item Single-Level Uncapacitated Lot-Sizing Problem (SISLULSP):

Single-Item Single-Level Uncapacitated Lot-Sizing Problem (SISLULSP) is discussed by many researchers. SISLULSP is one of the basic lot-sizing models. The major assumptions used in SISLULSP are as follows:

- Planning horizon is finite
- Demand is known in each period and is satisfied at the beginning of the period.
- Lead time is known and constant (without loss of generality it is set to zero).
- Backlog is not allowed; *i.e.,* system is uncapacitated.
- Setup cost for each production lot is constant over time.

- Inventory holding cost is linear and is charged to the ending inventories.
- Production cost is time-varying.
- Beginning and ending inventories are set to zero.
- A setup of the resource for each produced item in each period is necessary

Indices:

t Planning period ($t = 1,2,3, \dots, T$)

The decision variables are as follows:

I_t Inventory level at the end of period t

X_t Production quantity in period t

$Y_t = \begin{cases} 1 & \text{if product is produced in period } t \\ 0 & \text{otherwise} \end{cases}$

The parameters used are as follows:

D_t Demand in period t

h Holding cost

c_t Setup cost in period t

P_t Variable unit production cost in period t

I_0 Initial inventory level

M A large enough number, where M *takes a value of at least the summation* $\sum_{k=t}^{T} D_k$

The single-item uncapacitated lot-sizing problem can be formulated as follows:

Model SISLULSP:

$$Min \sum_{t=1}^{T} (P_t X_t + h I_t + c_t Y_t) \quad (1)$$

Subject to:

$$I_t = I_{(t-1)} + X_t - D_t \quad \forall t \quad (2)$$

$$X_t \leq M Y_t \quad \forall t \quad (3)$$

$$Y_t \in [0,1] \qquad \forall\, t \qquad (4)$$

$$I_t, X_t \geq 0 \qquad \forall\, t \qquad (5)$$

The objective function in Equation (1) is to minimize the sum of production, inventory holding and setup cost. Constraints (2) ensure the inventory balance condition. Constraints (3) ensure that production takes place in period t only if there is a setup during that period. Constraints (4) and (5) provide the logical binary and non-negativity necessities for the decision variables.

Many authors have studied the SISLULSP. One of the oldest classical production scheduling models is the Economic Order Quantity (EOQ) model, which is introduced by Harris (1913). In EOQ model, demand is assumed to be a continuous function over time. However, a different approach to solve the SISLULSP has been provided by Wagner and Whitin (1958), where discrete periods in time are considered and demand in each of these periods is assumed to be known in advance. Wagner and Whitin (1958) develop a forward-recursion algorithm, which is well known as WW algorithm, for the SISLULSP to obtain a minimum total cost inventory management scheme. The computational complexity of the WW algorithm is $O(T^2)$ time, where T denotes the number of periods. During the 1980s and 1990s, a lot of research is directed at improving the computational complexity of the lot-sizing algorithms for SISLULSPs. Evans (1985) presents an efficient computer implementation of the WW algorithm, which also runs in $O(T^2)$ time. Later, Federgruen and Tzur (1991) develop a simple forward algorithm, which can be implemented in $O(TlogT)$ time and $O(T)$ space. Wagelmans *et al.* (1992) and Aggarwal and Park (1993) both develop dynamic programming recursion for the SISLULSP that runs in $O(T)$ time for the WW case.

1.2.2 Single-Item Single-Level Capacitated Lot-Sizing Problem (SISLCLSP):

In the context of single-level production planning, with finite planning horizon and a known dynamic demand without incurring inventory shortage, the classical capacitated lot-sizing problem (CLSP) is to determine the production quantity and timing while satisfying the capacity restriction. This is the most used model in the literature. It is derived directly from the model of the SISLULSP (Section 1.2.1). To get the new model replace constraint (3) by the set of capacity constraints as follows:

$$\sum_{t=1}^{T}(p_t X_t + s_t Y_t) \leq R_t \quad \forall\, t \qquad (6)$$

Here p_t, s_t, and R_t are the processing time, setup time, and available capacity in period t respectively. Limited resource capacity is reflected by constraints (6).

1.2.3 Multi-Item Single-Level Uncapacitated Lot-Sizing Problem (MISLULSP):

Multi-item extension of the uncapacitated lot-sizing problem does not consider production capacity but often considers the inventory bounds in which a production plan for multiple items has to be determined considering that they share a storage capacity. This problem is addressed by Minner (2009). Akbalik, Penz, & Rapine (2015) study the complexity of this problem and prove that the problem is NP-hard even with no holding and fixed setup costs. Recently, Melo & Ribeiro (2017) study the mathematical formulations for the MIULSP with inventory bounds and provide two effective heuristics based on a rounding scheme and a relax-and-fix approach to solve the problem. The mathematical model for the classical MISLULSP presented by Melo & Ribeiro (2017) is as follows:

Indices:

t Planning period $(t = 1,2,3, \dots, T)$

j Item $(j = 1,2,3, \dots, n)$

The decision variables are as follows:

I_{jt} Inventory level for item j at the end of period t

X_{jt} Production quantity for item j in period t

$Y_{jt} = \begin{cases} 1 & \text{if item } j \text{ is produced in period } t \\ 0 & \text{otherwise} \end{cases}$

The parameters used are as follows:

D_{jt} Demand in period t

h_{jt} Holding cost of item j in period t

c_{jt} Setup cost of item j in period t

P_{jt} Variable unit production cost of item j in period t

H_t Total amount of stock available in period t

M A large enough number

Melo & Ribeiro (2017) assume that there are no initial and final stocks and that the demands and costs are nonnegative The mathematical formulation proposed by Melo & Ribeiro (2017) is as follows:

Model MISLULSP:

$$Min \sum_{j=1}^{n} \sum_{t=1}^{T} (P_{jt} X_{jt} + h_{jt} I_{jt} + c_{jt} Y_{jt}) \quad (7)$$

Subject to:

$$I_{jt} = I_{j(t-1)} + X_{jt} - D_{jt} \; \forall j, t \quad (8)$$

$$X_{jt} \leq \mathrm{M} * Y_{jt} \quad \forall j, t \quad (9)$$

$$\sum_{j=1}^{n} I_{jt} \leq H_t \; \forall t \quad (10)$$

$$I_{jt}, X_{jt} \geq 0 \qquad \forall\, j, t \tag{11}$$

$$Y_{jt} \in [0,1] \qquad \forall\, j, t \tag{12}$$

The objective function (7) minimizes the sum of storage costs, variable production costs and fixed production costs. Constraints (8) are inventory balance constraints. Constraints (9) are setup enforcing constraints. Constraints (10) limit the total stock at a given period. Constraints (11) and (12) are, respectively, nonnegativity and integrality constraints on the variables.

1.2.4 Multi-Item Single-Level Capacitated lot-Sizing Problem (MISLCLSP):

Multi-Item Single-Level Capacitated Lot-Sizing Problem (MISLCLSP) is an extension of the MISLULSP. MISLCLSP is a well-studied problem in which timing and lot-sizes are planned for the production of multiple items which share a single capacity constrained resource. Trigeiro, Thomas, and McClain (1989) are the first to attempt to solve the MISLCLSP with setup time.The mathematical model for the classical MISLCLSP proposed by Trigeiro et al. (1989) is as follows:

The multi-item uncapacitated lot-sizing problem can be formulated as follows:

Model MISLCLSP:

Min (8)

Subject to:

(9), (10), (12), (13)

$$\sum_{j=1}^{n} \sum_{t=1}^{T} (p_{jt} X_{jt} + s_{jt} Y_{jt}) \leq R_t \;\; \forall\, t \tag{14}$$

Here p_{jt}, and s_{jt} are processing time and setup time associated with item j in period t and R_t is the available capacity in period t. The objective of the model MISLCLSP is to minimize the total setup, holding and production cost. Limited resource capacity is reflected by constraints (14).

1.2.5 Multi-Level Capacitated lot-Sizing Problem (MLCLSP):

The multi-level extension of the CLSP, known as Multi-Level Capacitated Lot-Sizing Problem (MLCLSP) deals with the production of multiple items when interdependence among the different items at the different production levels is imposed due to the product structure. The classical MLCLSP is introduced by Billington, McClain, and Thomas (1983), which describes the following scenario. The planning horizon is finite and divided into T discrete time periods (e.g.,weeks). There are n items with period-specific external demands, which must be met without delay. The items are produced on m non-identical resources with limited period-specific capacities. Each resource comprises of one or more resource units, such as similar machines or workers, which are treated as a single entity. The mathematical formulation of the classical MLCLSP is presented in Chapter 3 Section 3.3.1.

1.3 Solution Approaches for lot-sizing problems:

Lot-sizing decisions are crucial because these decisions help the manufacturer determine the quantity and time to produce an item with a minimum cost. The efficiency and productivity of a system are completely dependent upon the right choice of lot-sizes. Therefore, developing and improving solution procedures for lot-sizing problems is key. The solution approaches of lot-sizing problems can be divided into three main areas: (i) Exact methods, (ii) Heuristic methods, and (iii) Metaheuristic methods. Florian et al. (1980) have proved that the single-item CLSP is NP-hard. Later, Bitran and Yanasse (1982) show that even special cases which are solvable in polynomial time become NP-hard when introducing a second item. Therefore to tackle the intractable nature of the lot-sizing problems, different heuristic and metaheuristic methods have been used.

1.3.1 Exact Methods:

Exact methods are useful to explore the underlying difficulties in solving the lot-sizing problems. For single item lot-sizing problems the mostly used exact methods include branch and bound (Erenguc & Aksoy, 1990), valid inequalities (Barany, Van Roy, & Wolsey, 1984; Miller, Nemhauser, & Savelsbergh, 2003), extended reformulations (Eppen & Martin, 1987; Rardin & Wolsey, 1993), Lagrangian relaxation (Billington, McClain, & Thomas, 1986; Chen & Thizy, 1990; Diaby, Bahl, Karwan, & Zionts, 1992) and Dantzig-Wolfe decomposition (Degraeve & Jans, 2007). Akartunalı and Miller (2012) study the computational complexities of the multi-level extension of the lot-sizing problems. Pochet and Wolsey (2006) provide an extensive discussion of the mathematical programming techniques used for lot-sizing problems.

1.3.2 Heuristic Approaches:

A heuristic is a strategy that is designed for solving a problem more quickly when classic methods are too slow, or for finding an approximate solution when classic methods fail to find an exact solution. This is achieved by trading optimality, completeness, accuracy, or precision for speed. Although exact methods are powerful since they provide a guarantee on solution quality, they exhibit an important drawback on the computational end; even with the modern fast computers and the state-of-the-art optimization packages, solving large-scale lot-sizing problems is a very complicated (and often an impossible) task. To compensate for the computational shortcomings of exact methods and to provide real-time solutions to practical problems, heuristic methods have been extensively used in this area.

Chen and Thizy (1990) have proved that multi-item CLSP is NP-hard. Therefore, different approaches are addressed in the literature to find near-optimal heuristic solutions for the

MISLCLSP. Trigeiro et al. (1989) are the first to attempt to solve the MICLSP with setup time to obtain near-optimal solutions. They propose a Lagrangian heuristics, which are iterative solution approaches applying Lagrangian Relaxation (LR). Thizy and Van Wassenhove (1985) , Trigeiro et al. (1989) and Sox and Gao (1999) suggested a Lagrangian relaxation based heuristics to solve a multi-item CLSP. Later Absi, Detienne, and Dauzère-Pérès (2013) apply LR to the capacity constraints and propose a non-myopic heuristic based on a probing strategy and a refining procedure. A number of set partitioning and column generation heuristics are proposed by Cattrysse, Maes, and Van Wassenhove (1990). Many researchers propose Relax-and-fix (RF) heuristic (Belvaux & Wolsey, 2000; Stadtler, 2003), which solves relaxed MIP subproblems sequentially and fixes binary variables throughout the process to speed up the solution procedure of the lot-sizing problems. Dantzig-Wolfe (DW) decomposition is applied for CLSP for the first time by Manne (1958). Later Jans and Degraeve (2004), Duarte & de Carvalho (2015) and Araujo et al. (2015) implemented DW decomposition-based heuristic to solve the lot-sizing problems. Fiorotto, de Araujo, and Jans (2015) combine LR and DW decomposition in a hybrid form for the MICLSP and show the competitiveness of the hybrid methods over other methods from the literature.

1.3.3 Metaheuristic Approaches:

The fundamental characteristics of metaheuristics are presented by Blum and Roli (2003) which are as follows:

- Metaheuristics are general strategies that guide the solution procedure of the optimization problems to find a sufficiently good solution.
- Metaheuristics are not problem-specific.

- Metaheuristics use the domain-specific knowledge in the form of problem-specific heuristics that are controlled by the upper level strategy.
- Metaheuristics are usually non-deterministic and may incorporate mechanisms to avoid getting trapped in confined areas of the search space. Furthermore, the search space may also include infeasible solutions, where the violation of constraints is charged with penalty cost.
- Metaheuristics belong to the group of improvement procedures starting from a given initial solution.
- The two basic principles that largely determine the behavior of a metaheuristic are intensification and diversification. The latter enhances the exploration of the search space, while the former allows for the exploitation of the accumulated search experience.

In recent years, there is a huge advancement in the implementation of metaheuristic approaches to solve the lot-sizing problems, such as the hybrid genetic algorithm (Dellaert & Jeunet, 2000), the simulated annealing (Raza & Akgunduz, 2008), the particle swarm optimization (Han, Tang, Kaku, & Mu, 2009), the variable neighborhood search (Xiao, Kaku, Zhao, & Zhang, 2011), the soft optimization approach based on segmentation (Kaku, Li, & Xu, 2008), the hybrid simulated annealing based tabu search (Berretta, Franca, & Armentano, 2005), the memetic algorithm (Berretta & Rodrigues, 2004), and the ant colony optimization system (Pitakaso, Almeder, Doerner, & Hartl, 2006). It has been reported that these algorithms can provide highly cost-efficient solutions within a reasonable time. Recently Duda (2017) applies Genetic Algorithms (GAs) hybridized with variable neighborhood search (VNS) to solve multi-item CLSP with setup times.

1.4 Scope of the Research:

This dissertation is concerned with the study of a SIULSP , which is motivated by the fact that many solution approaches of complex lot-sizing problems, which range from the single-item CLSPs to the multi-item MLCLSPs, lead to subproblems involving SIULSP. For example, the application of DW decomposition (Jans & Degraeve, 2004) and Lagrangian relaxation (Sox & Gao, 1999) to CLSP lead to the consideration of SIULSP as a subproblem. An efficient linear time algorithm for the SIULSPs will, hence,accelerate the convergence of such solution approaches.

The SIULSP is further extended to MLCLSP with setup carryover, backlogging and emission control. To the best of the author's knowledge, no attempt has been made to this point to tackle the MLCLSP while implementing emission control. DW decomposition has its application for single-level multi-item CLSP. But for multi-level extension of CLSP, it has never been implemented. Moreover the problem of determining setup carryover variable gives rise to a Maximum Weighted Independent Set (MWIS), which is a new area of application for MWIS.

1.5 Contributions of the Research:

The contributions of this piece of research could be summarized as:

First, the WW algorithm and its various improvements are revisited to develop a more efficient linear time algorithm for the single-level SIULSPs. The theoretical properties of the developed algorithm are derived and an experimental comparison with the similar algorithms existing in the literature is conducted. The analysis shows that the developed linear time algorithm outperforms its comparable algorithms in the literature given the various employed metrics of analysis.

Second, an item DW decomposition of the classical MLCLSP is presented. The MLCLSP is extended by allowing setup carryover and backlogging. An emission capacity constraint is also included, and the problem is referred to as MLCLSP with Setup Carryover, Backlogging, and Emission control (MLCLSP-SCBE). A Mixed Integer Linear Programming (MILP) model for the MLCLSP-SCBE is formulated, and an item DW decomposition of the proposed MILP formulation is proposed. Column Generation (CG) approach is used along with a novel Capacity Allocation (CA) heuristic to obtain feasible setup plans and an Integer Linear Programming (ILP) model to determine the setup carryover assignment to optimality. The method is hybridized with an LP-based improvement procedure, which helps to refine the solution further. The overall solution procedure reduces the optimality gap which is used as a benchmark to compare the performance of the proposed approach.

Third, it is shown that the Setup Carryover Assignment Problem (SCAP) is equivalent to the problem of finding the Maximum Weighted Independent Set (MWIS) in a chain of cliques. An ILP is formulated to determine the setup carryover variable and, it has been demonstrated that the SCAP and the special case of MWIS problem is solvable in Polynomial time.

1.6 Outline of the Dissertation:

This dissertation is comprised of five independent chapters. The definition of the lot-sizing problem along with its different characteristics and variants are presented in Chapter 1 (Introduction). Chapter 2 provides an efficient linear-time algorithm for the WW dynamic program and its implementation along with computational results assessing its performance. An MILP formulation and application of DW decomposition heuristic for an MLCLSP and its extensions is presented in Chapter 3. An experimental design and analysis for performance evaluation of the

proposed DW decomposition heuristics is also included in this chapter. Chapter 4 presents the problem of Setup Carryover Assignment (SCAP) for inventory lot-sizing as the problem of finding a Maximum Weighted Independent set. Finally, Chapter 5 concludes the dissertation and ends with some directions for the future research.

REFERENCES

Absi, N., Detienne, B., & Dauzère-Pérès, S. (2013). Heuristics for the multi-item capacitated lot-sizing problem with lost sales. *Computers & Operations Research, 40*(1), 264-272. doi: https://doi.org/10.1016/j.cor.2012.06.010

Aggarwal, A., & Park, J. K. (1993). Improved algorithms for economic lot size problems. *Oper. Res., 41*(3), 549-571. doi: 10.1287/opre.41.3.549

Akartunalı, K., & Miller, A. J. (2012). A computational analysis of lower bounds for big bucket production planning problems. *Computational Optimization and Applications, 53*(3), 729-753. doi: 10.1007/s10589-012-9465-z

Akbalik, A., Penz, B., & Rapine, C. (2015). Multi-item uncapacitated lot sizing problem with inventory bounds. *Optimization Letters, 9*(1), 143-154. doi: 10.1007/s11590-014-0746-6

Albers, S., & Brucker, P. (1993). The complexity of one-machine batching problems. *Discrete Applied Mathematics, 47*(2), 87-107. doi: http://dx.doi.org/10.1016/0166-218X(93)90085-3

Allahverdi, A., & Soroush, H. M. (2008). The significance of reducing setup times/setup costs. *European Journal of Operational Research, 187*(3), 978-984. doi: https://doi.org/10.1016/j.ejor.2006.09.010

Araujo, S. A. d., Reyck, B. D., Degraeve, Z., Fragkos, I., & Jans, R. (2015). Period Decompositions for the Capacitated Lot Sizing Problem with Setup Times. *INFORMS Journal on Computing, 27*(3), 431-448. doi: 10.1287/ijoc.2014.0636

Barany, I., Van Roy, T., & Wolsey, L. A. (1984). Uncapacitated lot-sizing: The convex hull of solutions. In B. Korte & K. Ritter (Eds.), *Mathematical Programming at Oberwolfach II* (pp. 32-43). Berlin, Heidelberg: Springer Berlin Heidelberg.

Belvaux, G., & Wolsey, L. A. (2000). bc — prod: A Specialized Branch-and-Cut System for Lot-Sizing Problems. *Management Science, 46*(5), 724-738. doi: 10.1287/mnsc.46.5.724.12048

Berretta, R., Franca, P. M., & Armentano, V. A. (2005). Metaheuristic Approaches For The Multilevel Resource-Constrained Lot-Sizing Problem With Setup And Lead Times. *Asia-Pacific Journal of Operational Research, 22*(02), 261-286. doi: 10.1142/s0217595905000510

Berretta, R., & Rodrigues, L. F. (2004). A memetic algorithm for a multistage capacitated lot-sizing problem. *International Journal of Production Economics, 87*(1), 67-81. doi: https://doi.org/10.1016/S0925-5273(03)00093-8

Billington, P. J., McClain, J. O., & Thomas, L. J. (1983). Mathematical Programming Approaches to Capacity-Constrained MRP Systems: Review, Formulation and Problem Reduction. *Management Science, 29*(10), 1126-1141. doi: 10.1287/mnsc.29.10.1126

Billington, P. J., McClain, J. O., & Thomas, L. J. (1986). Heuristics for Multilevel Lot-Sizing with a Bottleneck. *Manage. Sci., 32*(8), 989-1006. doi: 10.1287/mnsc.32.8.989

Bitran, G. R., & Yanasse, H. H. (1982). Computational Complexity of the Capacitated Lot Size Problem. *Management Science, 28*(10), 1174-1186. doi: 10.1287/mnsc.28.10.1174

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv., 35*(3), 268-308. doi: 10.1145/937503.937505

Briskorn, D. (2006). A note on capacitated lot sizing with setup carry over. *IIE Transactions, 38*(11), 1045-1047. doi: 10.1080/07408170500245562

Cattrysse, D., Maes, J., & Van Wassenhove, L. N. (1990). Set partitioning and column generation heuristics for capacitated dynamic lotsizing. *European Journal of Operational Research, 46*(1), 38-47. doi: https://doi.org/10.1016/0377-2217(90)90296-N

Chen, W. H., & Thizy, J. M. (1990). Analysis of relaxations for the multi-item capacitated lot-sizing problem. *Annals of Operations Research, 26*(1), 29-72. doi: 10.1007/BF02248584

Degraeve, Z., & Jans, R. (2007). A New Dantzig-Wolfe Reformulation and Branch-and-Price Algorithm for the Capacitated Lot-Sizing Problem with Setup Times. *Operations Research, 55*(5), 909-920. doi: 10.1287/opre.1070.0404

Dellaert, N., & Jeunet, J. (2000). Solving large unconstrained multilevel lot-sizing problems using a hybrid genetic algorithm. *International Journal of Production Research, 38*(5), 1083-1099. doi: 10.1080/002075400189031

Diaby, M., Bahl, H. C., Karwan, M. H., & Zionts, S. (1992). Capacitated lot-sizing and scheduling by Lagrangean relaxation. *European Journal of Operational Research, 59*(3), 444-458. doi: https://doi.org/10.1016/0377-2217(92)90201-J

Duarte, A. J. S. T., & de Carvalho, J. M. V. V. (2015). A Column Generation Approach to the Discrete Lot Sizing and Scheduling Problem on Parallel Machines. In J. P. Almeida, J. F. Oliveira & A. A. Pinto (Eds.), *Operational Research: IO 2013 - XVI Congress of APDIO, Bragança, Portugal, June 3-5, 2013* (pp. 157-170). Cham: Springer International Publishing.

Duda, J. (2017). A hybrid genetic algorithm and variable neighborhood search for multi-family capacitated lot-sizing problem. *Electronic Notes in Discrete Mathematics, 58*(Supplement C), 103-110. doi: https://doi.org/10.1016/j.endm.2017.03.014

Eppen, G. D., & Martin, R. K. (1987). Solving Multi-Item Capacitated Lot-Sizing Problems Using Variable Redefinition. *Operations Research, 35*(6), 832-848.

Erenguc, S. S., & Aksoy, Y. (1990). A branch and bound algorithm for a single item nonconvex dynamic lot sizing problem with capacity constraints. *Computers & Operations Research, 17*(2), 199-210. doi: https://doi.org/10.1016/0305-0548(90)90043-7

Evans, J. R. (1985). An efficient implementation of the Wagner-Whitin algorithm for dynamic lot-sizing. *Journal of Operations Management, 5*(2), 229-235. doi: http://dx.doi.org/10.1016/0272-6963(85)90009-9

Federgruen, A., & Tzur, M. (1991). A Simple Forward Algorithm to Solve General Dynamic Lot Sizing Models with n Periods in 0(n $\log$ n) or 0(n) Time. *Management Science, 37*(8), 909-925. doi: 10.2307/2632555

Fiorotto, D. J., de Araujo, S. A., & Jans, R. (2015). Hybrid methods for lot sizing on parallel machines. *Computers & Operations Research, 63*(Supplement C), 136-148. doi: https://doi.org/10.1016/j.cor.2015.04.015

Florian, M., Lenstra, J. K., & Kan, A. H. G. R. (1980). Deterministic Production Planning: Algorithms and Complexity. *Management Science, 26*(7), 669-679. doi: 10.1287/mnsc.26.7.669

Han, Y., Tang, J., Kaku, I., & Mu, L. (2009). Solving uncapacitated multilevel lot-sizing problems using a particle swarm optimization with flexible inertial weight. *Computers & Mathematics with Applications, 57*(11), 1748-1755. doi: https://doi.org/10.1016/j.camwa.2008.10.024

Harris, F. W. (1913). How Many Parts to Make at Once. *The Magazine of Management, 10*(2), 135-136,152.

Jans, R., & Degraeve, Z. (2004). Improved lower bounds for the capacitated lot sizing problem with setup times. *Operations Research Letters, 32*(2), 185-195. doi: https://doi.org/10.1016/j.orl.2003.06.001

Kaku, I., Li, Z., & Xu, C. (2008). *Solving Large Multilevel Lot-Sizing Problems with an Effective Heuristic Algorithm Based on Segmentation* (Vol. 6).

Kuik, R., Salomon, M., & van Wassenhove, L. N. (1994). Batching decisions: structure and models. *European Journal of Operational Research, 75*(2), 243-263. doi: https://doi.org/10.1016/0377-2217(94)90072-8

Manne, A. S. (1958). Programming of Economic Lot Sizes. *Management Science, 4*(2), 115-135. doi: 10.1287/mnsc.4.2.115

Melo, R. A., & Ribeiro, C. C. (2017). Formulations and heuristics for the multi-item uncapacitated lot-sizing problem with inventory bounds. *International Journal of Production Research, 55*(2), 576-592. doi: 10.1080/00207543.2016.1215567

Miller, A. J., Nemhauser, G. L., & Savelsbergh, M. W. P. (2003). On the polyhedral structure of a multi–item production planning model with setup times. *Mathematical Programming, 94*(2), 375-405. doi: 10.1007/s10107-002-0325-y

Minner, S. (2009). A comparison of simple heuristics for multi-product dynamic demand lot-sizing with limited warehouse capacity. *International Journal of Production Economics, 118*(1), 305-310. doi: https://doi.org/10.1016/j.ijpe.2008.08.034

Pitakaso, R., Almeder, C., Doerner, K. F., & Hartl, R. F. (2006). Combining population-based and exact methods for multi-level capacitated lot-sizing problems. *International Journal of Production Research, 44*(22), 4755-4771. doi: 10.1080/00207540600620963

Pochet, Y., & Wolsey, L. A. (2006). *Production Planning by Mixed Integer Programming (Springer Series in Operations Research and Financial Engineering)*: Springer-Verlag New York, Inc.

Rardin, R. L., & Wolsey, L. A. (1993). Valid inequalities and projecting the multicommodity extended formulation for uncapacitated fixed charge network flow problems. *European Journal of Operational Research, 71*(1), 95-109. doi: https://doi.org/10.1016/0377-2217(93)90263-M

Raza, A. S., & Akgunduz, A. (2008). A comparative study of heuristic algorithms on Economic Lot Scheduling Problem. *Computers & Industrial Engineering, 55*(1), 94-109. doi: https://doi.org/10.1016/j.cie.2007.12.004

Sox, C. R., & Gao, Y. (1999). The capacitated lot sizing problem with setup carry-over. *IIE Transactions, 31*(2), 173-181. doi: 10.1023/a:1007520703382

Stadtler, H. (2003). Multilevel Lot Sizing with Setup Times and Multiple Constrained Resources: Internally Rolling Schedules with Lot-Sizing Windows. *Operations Research, 51*(3), 487-502. doi: 10.1287/opre.51.3.487.14949

Tempelmeier, H., & Derstroff, M. (1996). A Lagrangean-Based Heuristic for Dynamic Multilevel Multiitem Constrained Lotsizing with Setup Times. *Management Science, 42*(5), 738-757.

Thizy, J. M., & Van Wassenhove, L. N. (1985). Lagrangean Relaxation for the Multi-Item Capacitated Lot-Sizing Problem: A Heuristic Implementation. *IIE Transactions, 17*(4), 308-313. doi: 10.1080/07408178508975308

Trigeiro, W. W., Thomas, L. J., & McClain, J. O. (1989). Capacitated Lot Sizing with Setup Times. *Management Science, 35*(3), 353-366. doi: 10.1287/mnsc.35.3.353

Wagelmans, A., Hoesel, S. V., & Antoon, K. (1992). Economic Lot Sizing: An O(n log n) Algorithm That Runs in Linear Time in the Wagner-Whitin Case. *Operations Research, 40*, S145-S156. doi: 10.2307/3840844

Wagner, H. M., & Whitin, T. M. (1958). Dynamic Version of the Economic Lot Size Model. *Management Science, 5*(1), 89-96. doi: 10.1287/mnsc.5.1.89

Wee, H.-M. (1999). Deteriorating inventory model with quantity discount, pricing and partial backordering. *International Journal of Production Economics, 59*(1), 511-518. doi: https://doi.org/10.1016/S0925-5273(98)00113-3

Wu, T., Shi, L., Geunes, J., & Akartunalı, K. (2011). An optimization framework for solving capacitated multi-level lot-sizing problems with backlogging. *European Journal of Operational Research, 214*(2), 428-441. doi: https://doi.org/10.1016/j.ejor.2011.04.029

Xiao, Y., Kaku, I., Zhao, Q., & Zhang, R. (2011). A variable neighborhood search based approach for uncapacitated multilevel lot-sizing problems. *Computers & Industrial Engineering, 60*(2), 218-227. doi: https://doi.org/10.1016/j.cie.2010.11.003

CHAPTER 2

DYNAMIC ECONOMIC LOT-SIZING PROBLEM: A NEW $O(T)$ ALGORITHM FOR THE WAGNER-WHITIN MODEL

Wagner and Whitin (1958) develop an algorithm to solve the dynamic Economic Lot-Sizing Problem (ELSP), which is widely applied in inventory control, production planning, and capacity planning. The original algorithm runs in $O(T^2)$ time, where T is the number of periods of the problem instance. Subsequently, other researchers develop linear-time algorithms to solve the Wagner-Whitin (WW) lot-sizing problem; examples include the ELSP and equivalent Single Machine Batch-Sizing Problem (SMBSP). This chapter revisits the algorithms for the ELSP and SMBSP under WW cost structure, presents a new efficient linear-time algorithm, and compares the developed algorithm with equivalent algorithms in the literature. The developed algorithm employs a lists and stacks data structure, which is a completely different approach than that of the comparable algorithms for the ELSP and SMBSP. Analysis of the developed algorithm shows that it executes fewer different actions throughout and hence it improves execution time by a maximum of 51.40% for the ELSP and 29.03% for the SMBSP.

2.1 Introduction:

The economic lot-sizing problem (ELSP) is an important issue in production and inventory control. Typically, a product is created or purchased in batch quantities and placed in stock. As the stock is depleted, more production or procurement must take place to replenish it. The main objective of the ELSP is to determine an optimum production or replenishment policy for a manufacturing or inventory system to meet the required market demand with the least possible expenditure. This policy decision is crucial, so it is a matter of interest for many researchers. Harris

(1913) introduces his well-known and fundamental Economic Order Quantity model, in which he assumes demand to be a continuous function over time. However, Wagner and Whitin (1958) provide a different approach to solving the lot-sizing problem. They consider time in discrete periods and assume that demand in each period is known in advance.

Wagner and Whitin (1958) develop a forward recursion algorithm to obtain a minimum total cost inventory management scheme, which satisfies demand known a priori in every period. They consider uncapacitated (i.e., without bounds on production and inventory) lot-sizing problems for a single-item inventory system. Their algorithm's main assumption is that an item produced in a period can satisfy the demand in that and subsequent periods. Any item incurs setup and unit production costs, and any item carried to the next period incurs a unit inventory holding cost. The goal is to find a minimum cost production plan. The Wagner-Whitin (WW) algorithm runs in $O(T^2)$ time, where T is the number of periods of the problem instance. Wagelmans et al. (1992) develop a linear-time algorithm (based on a geometric approach) for special cases of the WW problem where production and holding costs remain constant. Aggarwal and Park (1993) identify that the ELSP gives rise to Monge arrays (a special type of 2 × 2 array in which the four elements at the intersection points are such that the sum of the upper-left and lower-right elements across the main diagonal is less than or equal to the sum of the lower-left and upper-right elements across the antidiagonal). Employing the properties of a Monge array, Aggarwal and Park provide a linear-time algorithm for the WW problem. Albers and Brucker (1993) study the complexity of the single machine batch-sizing problem (SMBSP) and develop an algorithm for the shortest path problem that can be solved in linear time. The SMBSP can be defined as follows. Suppose there are n jobs, with given processing times, to be processed in batches on one machine. A batch is a set of jobs that is processed together. The number of jobs in a batch is called the *batch size*. The

production of a batch requires machine setups, which are assumed to be both sequence- and machine-independent. The problem is to find the optimal batch size that minimizes the total flow time. Flow time of a batch is the sum of the processing times of all jobs in that batch plus the machine setup time. Therefore, all jobs in a batch have the same flow time.

The Wagelmans et al. (1992) and Aggarwal and Park (1993) algorithms are famous in the field of ELSP and obtain excellent results in terms of time complexity. This chapter revisits these algorithms and presents a new linear-time algorithm for the ELSP under WW cost structure. The developed algorithm employs a lists and stacks data structure, which is a completely different approach than that of the existing algorithms (Aggarwal & Park, 1993; Wagelmans et al., 1992) in the literature. We match our result with the other algorithms (Aggarwal & Park, 1993; Wagelmans et al., 1992) for the ELSP and find that the new algorithm takes less CPU time and performs fewer various operations. The ELSP is equivalent to the SMBSP (see Section 2.4), so the developed algorithm is also applicable for solving the SMBSP. The developed algorithm is compared with the Albers and Brucker (1993) algorithm for the SMBSP and demonstrates its superiority in terms of various metrics of comparison. For the ELSP, we assume that holding costs are stationary but setup costs are time variant. However, for the SMBSP, we assume that setup costs for every job are constant.

The rest of this chapter is organized as follows. Section 2.2 reviews the related work in the literature. Section 2.3 provides a simpler linear-time algorithm for the WW dynamic program and its proofs. Section 2.4 illustrates how the developed algorithm can be implemented for the SMBSP. Section 2.5 presents a numerical example showing the implementation of the developed algorithm. Section 2.6 illustrates the computational results assessing the new algorithm's performance. Finally, Section 2.7 is the conclusion.

2.2 Literature review:

During the 1980s and 1990s, many researchers improve the computational complexity of the algorithms for the simple uncapacitated ELSP. Evans (1985) presents an efficient computer implementation of the WW algorithm, which is an $O(T^2)$ time dynamic programming recursion, where T denotes the number of periods. He exploits the special structure of the problem, which requires low core storage, enabling it to be potentially useful and efficient for solving lot-sizing problems.

There are many studies in the literature that discuss the improvement opportunities of the Wagner-Whitin algorithm to solve the single-item uncapacitated dynamic ELSP. Federgruen and Tzur (1991) develop a simple forward algorithm, which can be implemented in $O(TlogT)$ time and $O(T)$ space for the dynamic ELSP. They also provide linear-time algorithms for two distinct cases: (i) models without speculative motives for carrying stock and ii) models with nondecreasing setup costs. Wagelmans et al. (1992) develop a backward dynamic programming recursion for the uncapacitated ELSP that runs in $O(T)$ time for the WW case and $O(TlogT)$ time for a more general case, where marginal production costs differ between periods and all cost coefficients are unrestricted in sign. Aggarwal and Park (1993) show that the dynamic programming formulation of the uncapacitated ELSP gives rise to the Monge array, and they prove that the structure of the Monge arrays can be exploited to obtain a significantly faster algorithm. They present an $O(TlogT)$ time algorithm for both basic and backlogging ELSPs when the production, inventory, and backlogging costs are linear, and they show that for the special case of the WW model, this algorithm runs in $O(T)$ time.

Van Hoesel et al. (1994) also consider the Wagner and Whitin (1958) dynamic ELSP and generalize the algorithms developed by Federgruen and Tzur (1991) and Wagelmans et al. (1992) by introducing two basic geometric techniques to solve the ELSP in $O(TlogT)$ time. They discuss the forward and backward recursions for lot-sizing problems and the extension to the model, which allows backlogging, lot-sizing with start-up costs, and a generalized version of the model with learning effects in setup costs. They also show that the techniques used by Federgruen and Tzur (1991) and Wagelmans et al. (1992) are essentially the same.

Albers and Brucker (1993) study the complexity of the SMBSP for a fixed job sequence and develop a backward recursion algorithm that runs in $O(n)$ time, where n denotes the number of jobs. Baki and Vickson (2003) consider a lot-sizing problem in which a single operator completes a set of n jobs requiring operations on two machines. They develop an efficient algorithm for minimizing maximum lateness that can be solved in $O(n)$ time for both open and flow-shop cases. Mosheiov and Oron (2008) address the SMBSP to minimize total flow time for bounded batch sizes. They assume identical processing time for all jobs and identical setup time for all batches and introduce an efficient solution approach for both cases of an upper and a lower bound on the batch sizes. Li et al. (2012) extend Mosheiov and Oron (2008) by introducing a flexible upper bound for batch sizes, with the objectives of maximizing customer satisfaction and minimizing maximum completion time and flow time.

Teksan and Geunes (2015) provide a polynomial-time algorithm for the dynamic ELSP with convex costs in the production and inventory quantities. They consider a classic discrete-time, finite-horizon, uncapacitated, single-stage, dynamic lot-sizing problem with no backlogging. The resulting time complexity of their algorithm is $O(T^2logT)$.

Archetti et al. (2014) investigate an uncapacitated ELSP with two different cost discount functions. The first is the modified all unit discount cost function, which is piecewise and linear. They show that the problem can be solved in $O(I^2T^3)$ time complexity, where I is the number of echelons and T is the length of the discrete finite horizon. The second is the incremental discount cost function, which is increasing, piecewise, and linear. They show that the ELSP can be solved using a more efficient polynomial algorithm with an $O(T^2)$ time complexity.

Akbalik and Rapine (2013) study the complexity of a single-item uncapacitated lot-sizing problem with batch delivery, focusing on the general case of time-dependent batch sizes. They allow incomplete batches (fractional batches) in their model, with known demand over the planning horizon. They do not allow backlogging. They establish that if the cost parameters (setup cost, fixed cost per batch, unit procurement cost, and unit holding cost) are allowed to be time dependent, the problem is NP hard. By contrast, if all cost parameters are stationary and no unit holding cost is assumed, the problem is polynomially solvable in $O(T^3)$ time. They also show that in the case of divisible batch sizes, the problem of time-varying setup costs can be solved in time $O(T^3 logT)$ if there are no unit procurement or holding cost elements.

Wang et al. (2011) also study a single-item uncapacitated lot-sizing problem . They develop an $O(T^2)$ time algorithm to determine the lot sizes for manufacturing, remanufacturing, and outsourcing that minimizes the total cost, which consists of the holding costs for returns, manufactured and remanufactured products, setup, and outsourcing costs. Chu, Chu, Zhong, and Yang (2013) consider an uncapacitated single-item lot-sizing problem with outsourcing/subcontracting, backlogging, and limited inventory capacity. The backlogging level at each period is supposed to be limited. The authors show that this problem can be solved in

$O(T^4 logT)$ time. Fazle Baki, Chaouch, and Abdul-Kader (2014) discuss the ELSP with product return and remanufacturing and show that this kind of problem is NP hard. Retel Helmrich, Jans, van den Heuvel, and Wagelmans (2015) study the ELSP with an emission constraint. They show that ELSP with emission constraint is NP hard and propose several solution methods.

Hsu (2000) introduces an $O(T^4)$ time algorithm for the dynamic uncapacitated ELSP with perishable inventory under age-dependent holding costs and deterioration rates, where all cost functions are nondecreasing concave. Hsu (2003) extends Hsu (2000) by allowing backlogging in the model and gives an algorithm that runs in $O(T^4)$ time under some assumptions on cost functions and demand. Sargut and Işık (2017) extend Hsu (2003) by incorporating production capacity in the dynamic ELSP and provide a dynamic-programming-based heuristic for the solution of the overall problem.

Studies are ongoing to incorporate capacity constraints as an extension to the WW algorithm. Bitran and Yanasse (1982) show that Capacitated Lot-Sizing Problems (CLSPs) belong to the class of NP-hard problems. However, CLSPs with constant capacity can be solved in polynomial time (Florian & Klein, 1971). Okhrin and Richter (2011) explore a single-item CLSP with minimum order quantity and constant capacity. They assume constant unit production and holding cost elements and no stock-out. Considering this restriction, they derive an $O(T^3)$ time algorithm, where T is the length of the planning horizon. Later, Hellion et al. (2012) extend Okhrin and Richter's (2011) result to the problem of concave production and holding costs. They present an optimal algorithm with a time complexity $O(T^5)$. Akbalik and Rapine (2012) develop two polynomial-time algorithms for two versions of a constant CLSP with a constant batch size and a WW cost structure. They develop an $O(T^4)$ time algorithm for cases where production capacity is

a multiple of batch size and another $O(T^6)$ time algorithm for cases with an arbitrarily fixed capacity. Chu et al. (2013) study a single-item CLSP with production, holding, backlogging, and outsourcing cost functions. Assuming linear cost functions, they provide an $O(T^4 logT)$ time algorithm. Table 2.1 shows a summary of the relevant works in the literature related to lot-sizing algorithms.

Table 2.1: Summary of the relevant works in the literature related to lot-sizing algorithms

	Authors	Problem description/Assumptions	Complexity Result
Uncapacitated ELSP	Wagner and Whitin (1958)	Production cost is fixed. All period demands and costs are nonnegative.	$O(T^2)$
	Federgruen and Tzur (1991)	Holding costs proportional to the end-of-the-period inventory levels.	$O(TlogT)$
		Without speculative motives for carrying stock.	$O(T)$
		With non-decreasing setup costs.	
	Wagelmans et al. (1992)	All setup costs are nonnegative; marginal production costs differ between periods.	$O(TlogT)$
		Marginal production costs are identical, and holding costs are nonnegative.	$O(T)$
	Aggarwal and Park (1993)	The marginal cost of producing in period i is at most the marginal cost of producing in period $i-1$ plus the marginal cost of storing inventory from period $i-1$ to period i (WW cost structure).	$O(T)$
		The marginal cost of producing in period i and the marginal cost of storing inventory from period $i-1$ to period i is an arbitrary constant.	$O(TlogT)$
		Production and inventory cost functions are arbitrary and concave.	$O(T^2)$

	Hsu (2000)	Addresses the ELSP with perishable inventory under age-dependent holding costs and deterioration rates.	$O(T^4)$
	Wang et al. (2011)	Addresses the ELSP with manufacturing, remanufacturing, and outsourcing.	$O(T^2)$
	Retel Helmrich, Jans, van den Heuvel, and Wagelmans	Addresses the ELSP with emission constraint.	$O(T^4)$
Capacitated ELSP	Okhrin and Richter (2011)	Constant unit production cost, and no stock-out.	$O(T^3)$
	Hellion et al. (2012)	Constant capacity and constraint on minimum order quantity.	$O(T^5)$
	Akbalik and Rapine (2012)	Production capacity is a multiple of batch size.	$O(T^4)$
		Constant capacity.	$O(T^6)$

2.3 A new, simpler linear-time algorithm for the WW problem:

The WW-type dynamic program recursively computes

$$G(i) = \min_j \{C_{i,j} + G(j) | j = (i+1), (i+2) \dots, (T+1)\} \quad (1)$$

$\forall i = 1, \dots, T$, where $G(i)$ represents the minimum total cost to satisfy all demands in the consecutive periods i to T and $G(T+1)$ is initialized to 0. The problem is a special case of the shortest path problem in an acyclic directed network, where the cost of satisfying all demand of periods $i, \dots (j-1)$ in period i and continuing up to period $(j-1)$ is $C_{i,j}$. WW algorithm requires $O(T^2)$ of time and space to solve this shortest path problem. However, since $C_{i,j}$ has some special properties, many researchers have developed $O(T)$ and $O(TlogT)$ time algorithms for WW-type

dynamic programs. Our goal is to provide a further simplified and faster algorithm for the original WW case.

An important characteristic of the WW algorithm is the zero-inventory property (Wagner & Whitin, 1958), which implies that an optimal lot includes the summation of some complete period demands. If an order is placed in period $\forall k = 1,2,\dots,T$, it is optimal to order for the demands of periods $k, k+1, k+2, \dots . S^*(k) - 1$, where $S^*(k) \leq (T+1)$ and $S^*(k)$ is the successor of period k. This general idea is taken into account to determine the lot-size. It is straightforward to compute $S^*(k)\ \forall k = 1,2,\dots,T$ in $O(T^2)$ time. Researchers have developed algorithms to compute $S^*(k)$ in $O(T)$ time. However, we will discuss an alternate and faster approach to compute $S^*(k)$ in $O(T)$ time.

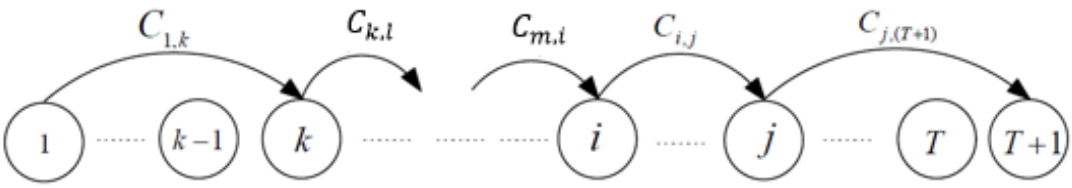


Figure 2.1: A network structure of the WW problem

Figure 2.1 shows a network representation of the WW problem. There are nodes 1,2,…, $(T+1)$; each represents the time period for a finite planning horizon. For each pair of nodes i and j, such that $1 \leq k < l < m < i < j \leq (T+1)$, there is an arc with cost C_{ij}. The WW problem is equivalent to the problem of finding the shortest path from node 1 to node $(T+1)$. Let d_k and f_k be the demand and setup costs for all periods $k = 1,2,\dots,T$. Holding costs are assumed to be fixed over the planning horizon; that is, $h_k = h\ \forall\ k = 1,2,\dots,T$. $C_{i,j}$ is computed using Equation 2.

$C_{i,j} = f_i + \sum_{l=i+1}^{j-1}(l-i)hd_l$ (2)

The total cost savings of an optimal path from node k to $(T+1)$ resulting from the use of arc (k,j) over arc (k,i) is the advantage of node j over node i as a successor of node k, denoted by $\Delta_k^{i,j}$ $\forall 1 \leq k < i < j \leq T+1$.

Definition 1: $\Delta_k^{i,j} = C_{k,i} + G(i) - C_{k,j} - G(j)$ $\forall 1 \leq k < i < j \leq T+1$.

Let $\delta(k,j)$ be the advantage of node $(j+1)$ over node j as a successor of node k.

Definition 2: $\delta(k,j) = \Delta_k^{j,j+1} = C_{k,j} + G(j) - C_{k,j+1} - G(j+1)$ $\forall 1 \leq k < j \leq T$.

Let $a(k)$ be the advantage of node $(k+2)$ over node $(k+1)$ as a successor of k.

Definition 3: $a(k) = \delta(k,k+1) = \Delta_k^{k+1,k+2}$ $\forall 1 \leq k \leq (T-1)$.

By Definition 1, $a(k) = C_{k,k+1} + G(k+1) - C_{k,k+2} - G(k+2) = f_k + G(k+1) - f_k - hd_{k+1} - G(k+2) = G(k+1) - G(k+2) - hd_{k+1}$.

Therefore, $a(k) = G(k+1) - G(k+2) - hd_{k+1}$. (3)

The advantage of node j over node i as a successor of node k $\forall 1 \leq k < i < j \leq T+1$ can be expressed as a summation of the advantages of each node i' over node $i'+1$ as a successor of node k $\forall i \leq i' < j$.

Lemma 1: The following statements hold true:

(i) $\Delta_k^{i,j} = \sum_{i'=i}^{j-1} \delta(k,i')$ $\forall 1 \leq k < i < j \leq T+1$.

(ii) $\Delta_k^{i,j+1} = \Delta_k^{i,j} + \delta(k,j)$ $\forall 1 \leq k < i < j \leq T$.

Proof: Consider Statement (i). From Definition 2, $\delta(k,i) = \Delta_k^{i,i+1} = C_{k,i} + G(i) - C_{k,i+1} - G(i+1)$. Similarly, $\delta(k,i+1) = \Delta_k^{i+1,i+2} = C_{k,i+1} + G(i+1) - C_{k,i+2} - G(i+2)$, and if it is expanded up to period $(j-1)$, the last term will be $\delta(k,j-1) = \Delta_k^{j-1,j} = C_{k,j-1} + G(j-1) - C_{k,j} - G(j)$. Summing all terms, $\delta(k,i) + \delta(k,i+1) + \cdots + \delta(k,j-1) = C_{k,i} + G(i) - C_{k,j} - G(j) = \Delta_k^{i,j}$.

⇨ $\Delta_k^{i,j} = \sum_{i'=i}^{j-1} \delta(k,i'); \forall 1 \le k < i < j \le T+1$.

Considering Statement (ii), $\Delta_k^{i,j+1} = \sum_{i'=i}^{j} \delta(k,i') = \sum_{i'=i}^{j-1} \delta(k,i') + \delta(k,j) = \Delta_k^{i,j} + \delta(k,j)$. ■

Now, we show that the advantage of node $(j+1)$ over node j as a successor decreases by a constant rate hd_j when it is searched from node k in lieu of $(k+1)$. Using this fact, we also show that the advantage of node j over node i $\forall i < j$ decreases at a rate $hv\sum_{i'=i}^{j-1} d_i$ when it is searched from $(k-v)$ in lieu of node k, where $1 \le v < k$.

Lemma 2: The following statements hold true:

(i) $\delta(k,j) = \delta(k+1,j) - hd_j, \forall 1 \le k < j \le T$, and

(ii) $\Delta_{k-v}^{i,j} = \Delta_k^{i,j} - hv\sum_{i'=i}^{j-1} d_{i'}, \forall 1 \le v < k < i < j \le T+1$.

Proof: From Definition 2, $\delta(k,j) = \Delta_k^{j,j+1} = C_{k.j} + G(j) - C_{k,j+1} - G(j+1)$.

Similarly, $\delta(k+1,j) = C_{k+1.j} + G(j) - C_{k+1,j+1} - G(j+1)$.

Now, $\delta(k+1,j) - \delta(k,j) = C_{k+1,j} - C_{k+1,j+1} - C_{k,j} + C_{k,j+1}$.

Or, $\delta(k+1,j) - \delta(k,j) = f_{k+1} + \sum_{i'=k+2}^{j-1}(i'-k-1)hd_{i'} - f_{k+1} - \sum_{i'=k+2}^{j}(i'-k-1)hd_{i'} - f_k - \sum_{i'=k+1}^{j-1}(i'-k)hd_{i'} + f_k + \sum_{i'=k+1}^{j}(i'-k)hd_{i'}$ (see Equation 2)

$= (j-k)hd_j - (j-k-1)hd_j = hd_j$.

Therefore, $\delta(k,j) = \delta(k+1,j) - hd_j \; \forall 1 \le k < j \le T$.

This proves Statement (i).

So, $\delta(k-v,j) = \delta(k-v+1,j) - hd_j = \delta(k-v+2,j) - 2hd_j = \cdots = \delta(k,j) - vhd_j$ (4)

Using Lemma 1(i), $\Delta_{k-v}^{i,j} = \sum_{i'=i}^{j-1} \delta(k-v,i')$.

Using Equation 4, $\Delta_{k-v}^{i,j} = \sum_{i'=i}^{j-1}\{\delta(k,i') - vhd_{i'}\} = \sum_{i'=i}^{j-1} \delta(k,i') - \sum_{i'=i}^{j-1} vhd_{i'}$.

Using Lemma 1(i), $\Delta_{k-v}^{i,j} = \Delta_k^{i,j} - \sum_{i'=i}^{j-1} vhd_{i'}$.

This proves Statement (ii) ■

Let $b(k)$ be the rate by which the advantage of node $(k+2)$ decreases over node $(k+1)$ as a successor when it is searched from node $(k-1)$ instead of k. From Lemma 2(i), we know that this rate is hd_{k+1}.

Definition 4: $b(k) = hd_{k+1} \; \forall 1 \le k \le T-1$.

Corollary 1: $\delta(k-u,k+1) = a(k) - ub(k) \; \forall 0 \le u < k \le T-1$.

Proof: From Lemma 2(i) and Definition 4, $\delta(k-u,k+1) = \delta(k-u+1,k+1) - b(k) =$

$= \delta(k-u+2,k+1) - 2b(k) = \cdots = \delta(k-u+u,k+1) - ub(k) = a(k) -$

$ub(k)$. (see Definition 3). ■

The discussion in the beginning of Section 2.3 shows that the WW problem is equivalent to finding $S^*(k) \; \forall k = 1,2,\ldots,T-1$. Lemma 3 provides a few rules on how to find $S^*(k)$.

Lemma 3: The following statements hold true:

(i) $S^*(k) = k+1$, if $\delta(k,j) \le 0 \; \forall 1 \le k < j \le T$.

(ii) $S^*(k) = k+1$, if and only if $\Delta_k^{k+1,j} \leq 0 \; \forall 1 \leq k < k+1 < j \leq T+1$.

(iii)$S^*(k) = r$, if and only if $\Delta_k^{i,r} \geq 0$ and $\Delta_k^{r,j} \leq 0 \;\; \forall 1 \leq k < i < r < j \leq T+1$.

(iv)$S^*(k-v) \leq r$, if $S^*(k) = r \; \forall 0 \leq v < k < r \leq T+1$.

Proof**:** By Definition 2, if $\delta(k,j) \leq 0$, then j is not worse than $(j+1)$ as a successor of k. Therefore, if $\delta(k,j) \leq 0, \forall 1 \leq k < j \leq T$, then $k+1$ is the best successor of k. This proves Statement (i).

By Definition 1, $\Delta_k^{k+1,j} \leq 0$ is equivalent to the fact that $k+1$ is not worse than j as a successor of k. Hence, if and only if $\Delta_k^{k+1,j} \leq 0 \; \forall \, j = k+2, \ldots, T+1$, then $S^*(k) = k+1$. This proves Statement (ii).

By Definition 1, $\Delta_k^{i,r} \geq 0$ is equivalent to the fact that r is not worse than i as a successor of k and $\Delta_k^{r,j} \leq 0$ is equivalent to the fact that r is not worse than j as a successor of k. Therefore, if and only if $\Delta_k^{i,r} \geq 0$ and $\Delta_k^{r,j} \leq 0 \; \forall k+1 \leq i < r < j \leq T+1$, then $S^*(k) = r$. This proves Statement (iii).

Statement (iv) is trivially true for $r = T+1$. Hence, let us consider $r < T+1$. If for some k and r such that $1 \leq k < r \leq T$, $S^*(k) = r$, then either $r = k+1$ or $k+1 < r \leq T$. If $r = k+1$, then from Statement (ii), $\Delta_k^{k+1,j} \leq 0 \;\; \forall \, j = k+2, .., T+1$. If $k+2 \leq r \leq T$, then from Statement (iii), $\Delta_k^{r,j} \leq 0 \;\; \forall k+1 < r < j \leq T+1$. In either case, $\Delta_k^{r,j} \leq 0 \;\; \forall r < j \leq T+1$. Now, applying Lemma 2(ii), $\Delta_{k-v}^{r,j} \leq \Delta_k^{r,j} \leq 0 \; \forall 1 \leq v < k$. However, $\Delta_{k-v}^{r,j} \leq 0$ means that r is not worse than j as a successor of $k-v$. Therefore, $\Delta_{k-v}^{r,j} \leq 0 \;\; \forall r < j \leq T+1$ implies that $S^*(k-v) \leq r \; \forall 1 \leq v \leq k-1$. This proves Statement (iv) ■

Statement (iv) of Lemma 3 allows us to delete nodes during the search for the best successor.

Let Matrix A be an upper triangular matrix whose structure appears in Figure 2.2. Matrix A contains the advantage of node $j+1$ over node j as a successor of node k, where $1 \leq k < j \leq T+1$.

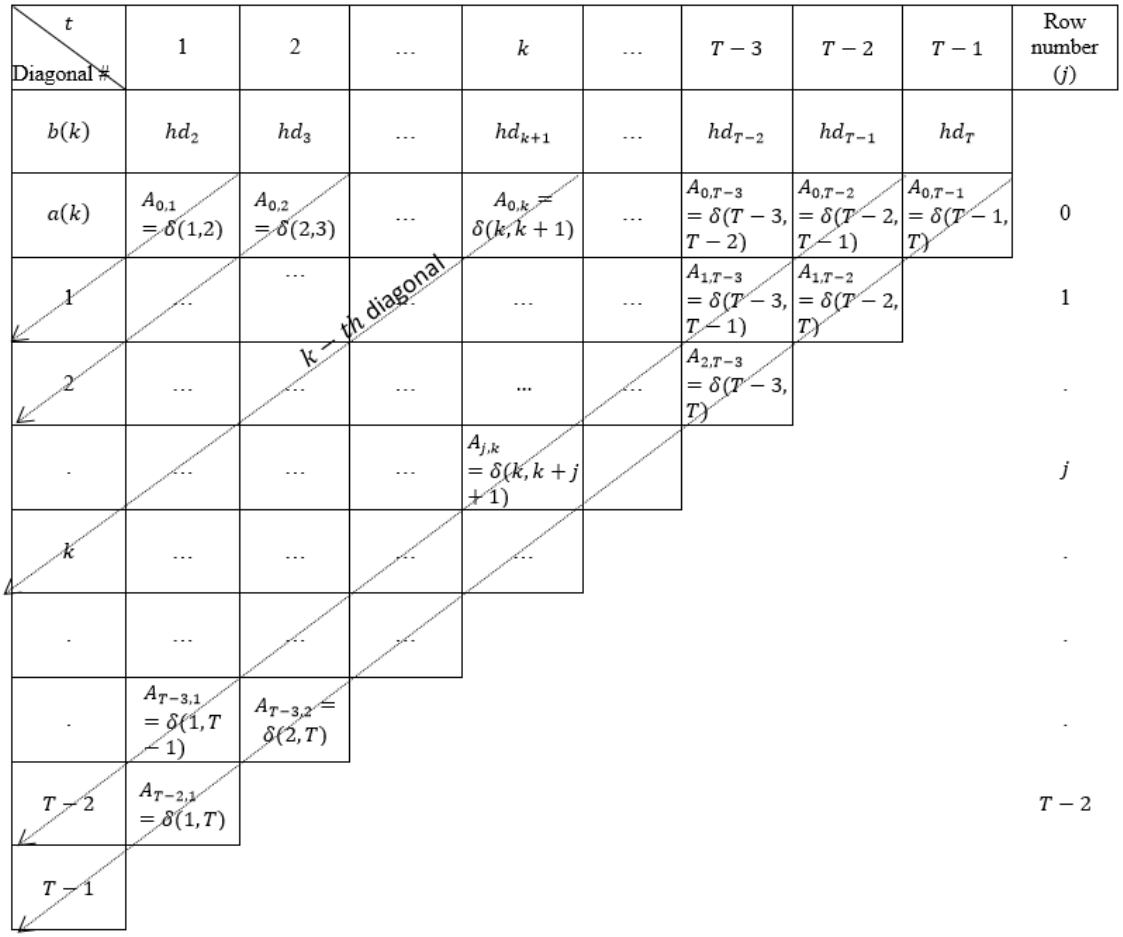


Figure 2.2: Structure of Matrix A

Definition 5: $A = \{A_{j,k} | k = 1,2,..(T-1); j = 0,1,..(T-1-k); A_{j,k} = \delta(k, k+j+1)\}$

Any cell of this matrix that is in the $k-$th column and $j-$th row is positioned in the $(k+j)-$th diagonal, and its value is $\delta(k,\ k+j+1)$ (see Definition 5). For example, in Figure

2.2, $\delta(T-3,\ T-1)$ is located in column $(T-3)$ and row 1. Thus, we can say that $\delta(T-3,\ T-1)$ is located in the $(T-2)-$th diagonal. Each column of the matrix represents the time period for the planning horizon.

Let Matrix B be an upper triangular matrix, as illustrated in Figure 2.3. Matrix B contains the cumulative advantages of Matrix A.

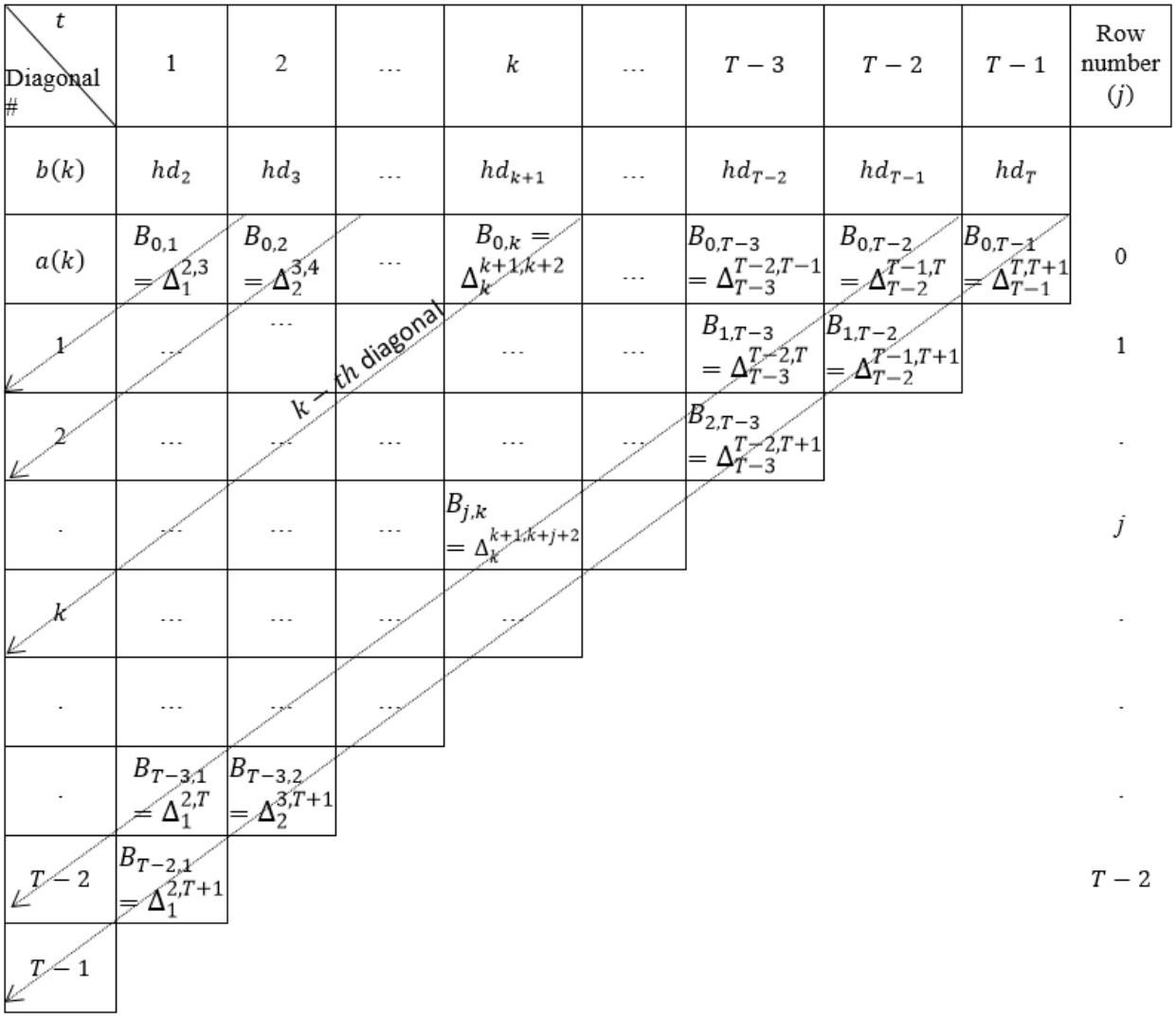


Figure 2.3: Structure of Matrix B

Definition 6: $B = \left\{ B_{j,k} \middle| k = 1,2,..(T-1); j = 0,1,..(T-1-k); B_{j,k} = \sum_{j'=0}^{j} A_{j',k} \right\}$.

$B_{j,k}$ can also be represented as follows:

$B_{j,k} = \sum_{j'=0}^{j} A_{j',k}$ (see Definition 6) $= \sum_{j'=0}^{j-1} A_{j',k} + A_{j,k} = B_{j-1,k} + A_{j,k}$ (5)

Any cell of Matrix *B* that is in the $k-$th column and $j-$th row is positioned in the $(k+j)-$th diagonal and its value is $\Delta_k^{k+1,k+j+2}$ (see Lemma 4). Therefore, the following results are obtained:

Lemma 4: $B_{j,k} = \Delta_k^{k+1,k+j+2}$.

Proof: This statement holds for $j = 0;\ B_{0,k} = A_{0,k} = \delta(k, k+1) = \Delta_k^{k+1,k+2}$. Suppose it holds true for $j < m$ for some $m > 0$. Therefore, $B_{m-1,k} = \Delta_k^{k+1,k+m+1}$. Now, for $j = m$, using Equation 5, $B_{j,k} = B_{j-1,k} + A_{j,k} = \Delta_k^{k+1,k+j+1} + \delta(k, k+j+1) = \Delta_k^{k+1,k+j+2}$ (See Lemma 1(ii)). ■

Theorem 1: $\Delta_k^{k+1,r}$ is located in the $(r-2)$ −th diagonal of Matrix $B\ \forall k+2 \le r \le T+1$.

Proof: It is known that $\Delta_k^{k+1,k+j+2}$ is located in the $(k+j)-$th diagonal.

Let $r = k+j+2$. So, $(k+j) = r-2$.

Therefore, $\Delta_k^{k+1,r}$ is located in the $(r-2)$ −th diagonal of Matrix *B*. ■

Theorem 2: $S^*(k) = r\ \forall\ 1 \le k < k+1 < r \le T+1$, if and only if $\max_{k+1<p\le T+1} \Delta_k^{k+1,p}$ is $\Delta_k^{k+1,r}$ and $\Delta_k^{k+1,r} \ge 0$.

Proof: Using Definition 1, it can be shown that $\Delta_k^{k+1,r} = \Delta_k^{k+1,i} + \Delta_k^{i,r} = \Delta_k^{k+1,j} - \Delta_k^{r,j}$. Thus, the above statement follows from Lemma 3, Statements (ii) and (iii) ■

Lemma 3, Statement (ii) characterizes cases when $S^*(k) = k + 1$. Theorem 2 characterizes the cases when $S^*(k) > k + 1$. Note that $\max_{k+1<p\leq T+1} \Delta_k^{k+1,p}$ is the largest cell of column k in Matrix B. Lemma 3, Statement (ii) and Theorem 2 together imply that the best successor of k, $S^*(k)$ is $k + 1$ if and only if all entries of column k of Matrix B are nonpositive. Otherwise, if there is at least one nonnegative entry in column k of Matrix B, $S^*(k) = r > k + 1$ if and only if the largest cell of column k lies in the $(r - 2)$ −th diagonal. Thus, the WW problem is equivalent to finding the largest cell in each column of Matrix B. Now, we will discuss an algorithm to find the largest cell of each column of Matrix B without calculating any entry of Matrix B, but calculating entries of Matrix A on an as-needed basis.

The algorithm tracks the best diagonal i^* that contains the largest cell of the k −th column of Matrix B. According to Theorem 1 and Theorem 2, $S^*(k) = i^* + 2$.

The developed algorithm uses a list $L(k)$ $\forall k = 1,2, \dots T - 1$.

Definition 7: If $j \in L(k)$, $\forall 1 \leq k \leq j \leq T - 1$, then $A_{j-k,k} \leq 0$ and either $j = k$ or $j > k$ and $A_{j-k-1,k+1} > 0$.

Therefore, whenever $j \in L(k)$, the j −th diagonal can be deleted from the search of the largest cell in columns $1, \dots, k$ of Matrix B. Initially, $L(k)$ is empty: $\forall k = 1,2, \dots T - 1$.

Theorem 3: If $u = max\left(min\left(\left\lceil\frac{a(k)}{b(k)}\right\rceil, T\right), 0\right) \leq k - 1$, then $k \in L(k - u)$.

Proof: If $0 \leq u \leq k-1$, then $a(k) - ub(k) \leq 0$. From Corollary 1, $\delta(k-u, k+1) \leq 0$. Note $\delta(k-u, k+1)$ is located in the $k-$th diagonal. According to Lemma 2(i), the other members of the $k-$th diagonal $\delta(k-u', k+1) \leq 0 \ \forall\, u \leq u' \leq k-1$. Hence, the $k-$th diagonal can be deleted from the search for the largest cell in columns $k-u \ \forall 0 \leq u \leq k-1$. Therefore, by Definition 7, $k \in L(k-u)$ ■

If k' is the successor of k, then k is the predecessor of k'. Let $S(k)$ and $P(k)$ be the successor and predecessor of node k, respectively. In our algorithm, we initialize $S(k) = k+1$ $\forall k = 1,2,\dots,T-1$ and $P(k) = k-1 \ \forall k = 2,3,\dots T$ (see Line 1 of Algorithm 1).

Definition 8: A stack is a set of contiguous cells in the same column of Matrix A.

The diagonal of the topmost cell of a stack is the *head*, and the diagonal of the bottommost cell is the *tail* (see Figure 2.4). The algorithm ensures that $S\big(P(p)\big) = p$ if the $p-$th diagonal is not deleted from the previous iterations (see Line 7 of Algorithm 1). Deletion of diagonals starts if $p \in L(k)$, $p \leq i^*$, and p is not deleted in previous iterations. At this point, we start a stack with *head* p (see Line 9 of Algorithm 1), and we search for a *tail*. The *tail* is the first cell below the *head* such that the sum of all cells from *head* to *tail* is positive. Definition 9 more precisely defines *head* and *tail*.

Definition 9: $head = p$, if $p \in L(k)$, $p \leq i^*$ and p is not deleted in previous iterations, and $tail = min_{p" > head} \sum_{p'=head}^{p"} A_{p'-k,k} > 0$.

If a *tail* does not exist, then the search fails, and i^* is updated as $i^* = P(head)$ (see Line 23 of Algorithm 1), which is equivalent to deleting all diagonals below the *head*.

However, if a *tail* is found, then i^* remains unchanged, and all diagonals of the stack except the *tail* are deleted (see Line 12 of Algorithm 1). The whole stack is considered as one cell, and the *tail* information is updated with the stack information. More precisely, we initialize δ on Line 8 and keep updating δ in Line 14 until a *tail* is found when δ in Line 14 gives $\sum_{p'=head}^{tail} A_{p'-k,k}$. Theorem 4 explains how the update of $a(tail)$ and $b(tail)$ in Line 16 ensures that the δ in subsequent iterations correctly calculates $\sum_{p'=head}^{tail} A_{p'-k,k}$.

Theorem 4: If a *tail* is found, all diagonals of the stack except the *tail* should be deleted, and $b(tail)$ and $a(tail)$ should be updated as follows: $b(tail) = \sum_{p'=head}^{tail} b(p')$ and $a(tail) = \sum_{p'=head}^{tail} A_{p'-k,k} + (tail - k)b(tail)$.

Proof: The largest cell in columns 1, … , k of Matrix B cannot be in any diagonal of the stack except the *tail*. Therefore, all diagonals of the stack except the *tail* should be deleted.

$$\sum_{p'=head}^{tail} A_{p'-k,k} - \sum_{p'=head}^{tail} A_{p'-k-1,k+1}$$

$$= \sum_{p'=head}^{tail} \delta(k, p'+1) - \sum_{p'=head}^{tail} \delta(k+1, p'+1)$$

$= \sum_{p'=head}^{tail} hd_{p'+1}$ (See Lemma 2(i))

$= \sum_{p'=head}^{tail} b(p')$ (See Definition 4)

More generally, $\sum_{p'=head}^{tail} A_{p'-k',k'} - \sum_{p'=head}^{tail} A_{p'-k'-1,k'+1} = \sum_{p'=head}^{tail} b(p') \; \forall 2 \le k' \le k$.

Hence, the sum of all diagonals in the stack decreases at the rate of $\sum_{p'=head}^{tail} b(p')$ from column k' to column $(k'-1)$ $\forall 2 \le k' \le k$, with a value of $\sum_{p'=head}^{tail} A_{p'-k,k}$ at column k. Therefore,

when $b(tail)$ and $a(tail)$ are updated as $b(tail) = \sum_{p'=head}^{tail} b(p')$ and $a(tail) = \sum_{p'=head}^{tail} A_{p'-k,k} + (tail - k)b(tail)$,

we get, $\sum_{p'=head}^{tail} A_{p'-k',k'} = a(tail) - (tail - k')b(tail)\ \forall 1 \le k' \le k$. ■

Figure 2.5 illustrates the flow of different definitions, lemmas, and theorems, showing that the lemmas are used to derive the theorems and that the results of the theorems are directly used in Algorithm 1.

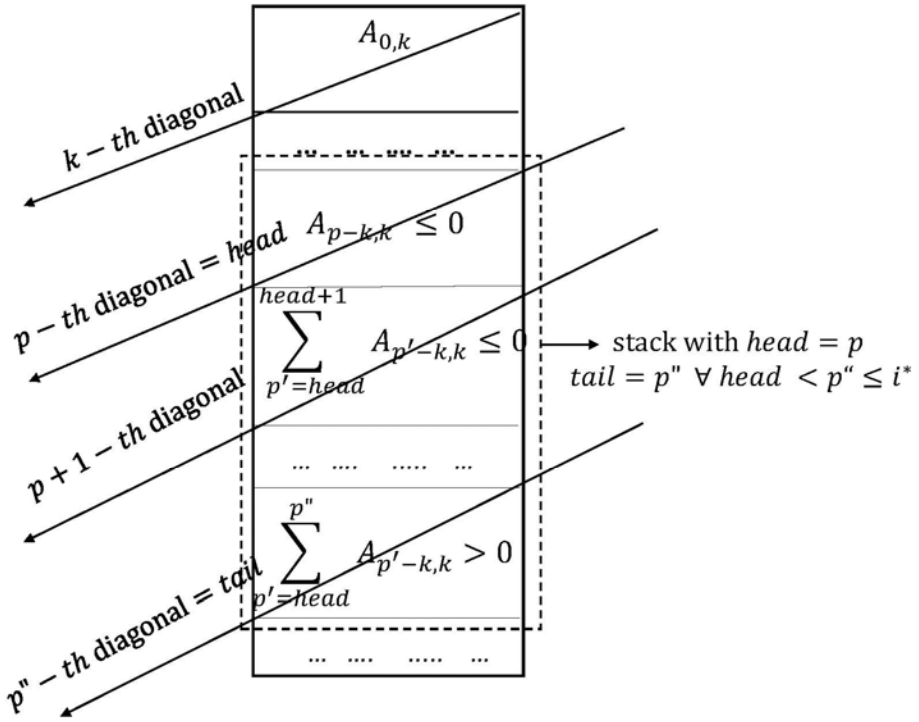


Figure 2.4: A stack in the k –th column when $p \in L(k)$

Algorithm 1: A new $O(T)$ Algorithm for Dynamic Economic Lot-Sizing (WW case)

Input: $T, h, d_k, f_k;\ \forall\ k = 1,2, \ldots, T$

Output: $S^*(k)$

Initialization:

$P(k) \coloneqq k-1, \forall k = 2, \dots, T;\ S(k) \coloneqq k+1, \forall k = 1,2,3, \dots, T-1$

$G(T+1) \coloneqq 0;\ \ a(T) \coloneqq 0; G(T) \coloneqq f_T; i^* \coloneqq T-1$

Iterations: For $k = T-1$ *down to* 1

$a(k) \coloneqq G(k+1) - G(k+2) - hd_{k+1}$ (see Equation 3); $b(k) \coloneqq hd_{k+1}$

Let $u \coloneqq max\left(min\left(\left\lceil \frac{a(k)}{b(k)} \right\rceil, T\right), 0\right)$

$if\ u \le (k-1), L(k-u) += \{k\}$ (see Theorem 3).

$For\ all\ p\ in\ L(k) |\ p \le i^*\ and\ S\big(P(p)\big) = p\ \ do$

$\delta \coloneqq a(p) - (p-k)b(p);\ \ \bar{b} \coloneqq b(p)$

$head \coloneqq p,$

while $\delta \le 0$ and $p \le i^*$

$if\ p < i^*$

$S\big(P(p)\big) \coloneqq S(p); P\big(S(p)\big) \coloneqq P(p);$

$p \coloneqq S(p)$

$\delta \coloneqq \delta + a(p) - (p-k)b(p); \bar{b} \coloneqq \bar{b} + b(p)$

$if\ \delta > 0$

$a(p) \coloneqq \delta + (p-k) * \bar{b};\ \ b(p) \coloneqq \bar{b}$ (see Theorem 4).

Let $u \coloneqq min\left(\left\lceil \frac{\delta}{b(p)} \right\rceil, T\right)$

$if\ u \le (k-1)$

$L(k-u) += \{p\}$

$end - if$

$end - if$

$else$

$i^* \coloneqq P(head)$

end - if

$end - while$

$end - for$

$S^*(k) \coloneqq i^* + 2$; (see Theorem 1 and 2).

$G(k) \coloneqq f_k + h\sum_{i=k+1}^{S^*(k)-1}(i-k) * d_i + G\big(S^*(k)\big)$

end - for

Backtracking for finding the optimal ordering period:

$k \coloneqq 1, m \coloneqq 1$

$While\ k \le T\ do$

$x(m) \coloneqq k$

$m \coloneqq m+1$

$end - while$

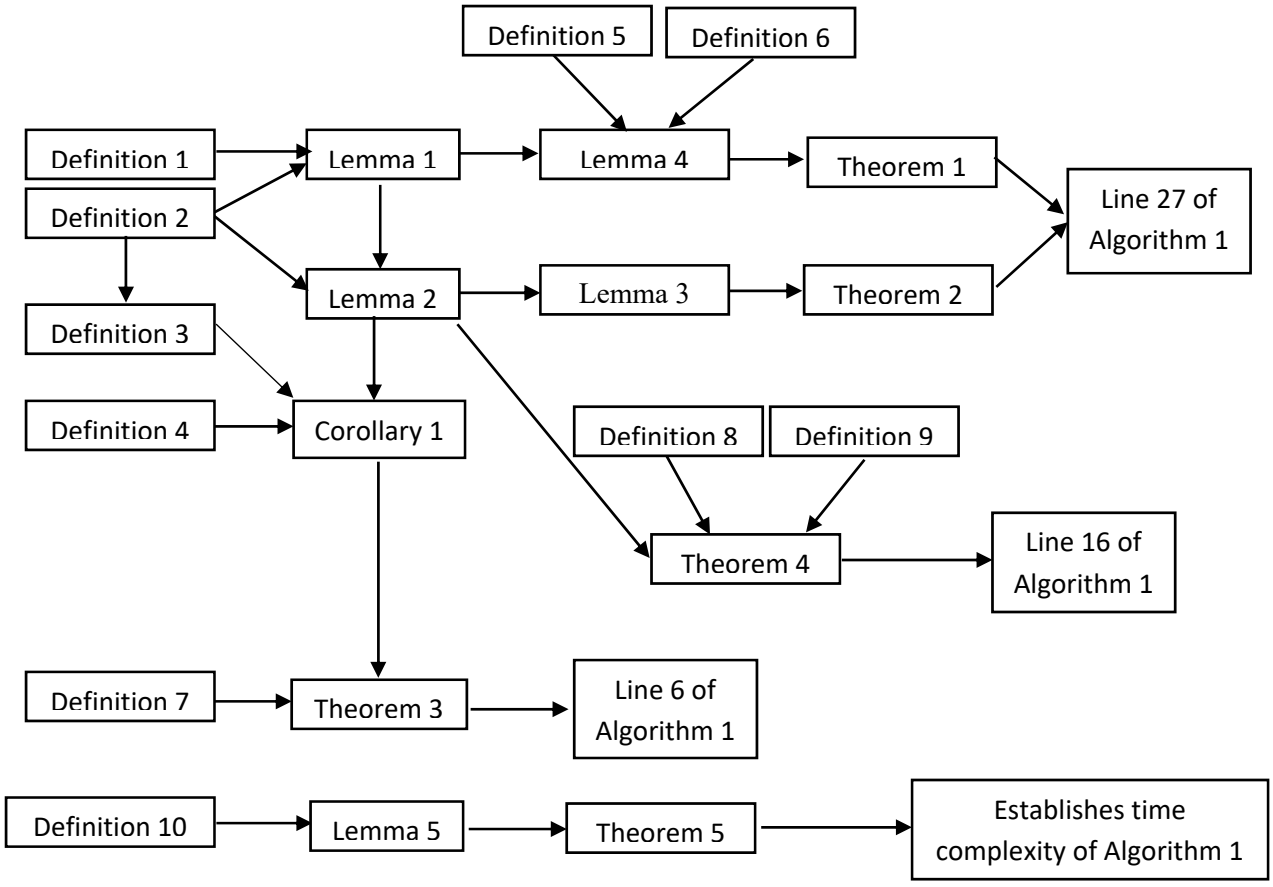


Figure 2.5: Road map toward the application of different definitions, lemmas, and theorems

Now we will evaluate the time complexity of Algorithm 1.

Definition 10: For a T period WW problem, $L = L(1) \cup L(2) \cup \ldots \cup L(T-1)$.

Lemma 3: For a T period WW problem,

$i. if\ T = 2\ then\ |L| \leq 1.$

$ii. if\ T \geq 3\ then\ |L| \leq (2T-4).$

Proof: Case i is easy to check. Let us consider case ii. A new element may be added to the list $L(k)\ \forall\ k = 1, 2, \ldots\ (T-1)$ in Lines 6 and 19 of the above pseudocode. Line 6 is executed $(T-1)$ times. Below, we show that Line 19 is executed at most $(T-3)$ times. In the pseudocode, δ is

calculated only in Lines 8 and 14. The δ in Line 8 represents $\delta(k, k+j+1)\ \forall\ k = 1,2,..(T-1); j = 0,1,..(T-1-k)$, and the δ in Line 14 represents $\Delta_k^{k+1,k+j+2}\ \forall\ k = 1,2,..(T-1); j = 0,1,..(T-1-k)$. The If statement in Line 15 is not executed for $k = (T-1)$ because when $k = (T-1)$, $\delta(T-1,\ T)$ can be either positive or negative. If $\delta(T-1,\ T) > 0$, the inner For loop is not executed because according to Line 5, $u > 0$. Thus, $L(T-1)$ remains empty. If $\delta(T-1,\ T) \leq 0$, the While loop is executed because $u < 0$, and hence, $L(T-1) = T-1$, but the If statement in Line 15 is not executed because $\delta < 0$. Thus, Line 18 does not add any element to $L(T-1)$.

Again, when $k = 1$, the If statement in Line 15 may run, but according to Line 17, $u > 0$. Line 19 is executed only if $u \leq (k-1) = 1-1 = 0$. Hence, Line 19 is not executed for $k = 1$ and $k = T-1$. For $k = 2, 3, ...(T-2)$, every time the While loop runs, at least one diagonal is deleted. Therefore, the While loop, the If statement in Line 15, and Line 19 run at most $(T-3)$ times.

So, $|L| \leq (T-1) + (T-3) = (2T-4)$. ■

Theorem 1: Algorithm 1 requires $O(T)$ time.

Proof: The outer For loop runs $(T-1)$ times. The While loop runs at least once every time the inner For loop runs. In every iteration of the While loop, at least one diagonal is deleted. There are $(T-1)$ diagonals, so the While loop, as well as the inner For loop, runs at most $(T-1)$ times.

All statements of the pseudocode require constant time except the condition of the inner For loop. The total number of times the condition is checked is the same as $|L|$, and, according to Lemma 5(ii), $|L| \leq 2T-4$.

Thus, Algorithm 1 runs in $O(T)$ time. ■

2.4 Single Machine Batch-Sizing Problem (SMBSP):

In the SMBSP, a fixed but arbitrary job sequence $JS = J_1,\ J_2, \dots ,\ J_n$ is given such that the processing time of job J_i is $p_i\ \ \forall\ i = 1,2, \dots n$. The problem is to determine the optimal batch sizes with the objective of minimizing the total flow time, $\mathbb{F} = \sum_{i=1}^{n} F_i$, where F_i is the flow time for job $J_i\ \ \forall\ i = 1,2, \dots n$. Batch sizes are between 1 and n, and all jobs in a batch are completed after the last job of the batch is completed. Thus, all jobs in a batch have the same flow time. A batching schedule is of the following type:

$$BS = \underbrace{SJ_{j_1}, \dots J_{j_2-1}}_{\text{Batch 1}}\underbrace{SJ_{j_2} \dots J_{j_3-1}}_{\text{Batch 2}} \dots \dots \dots \dots \underbrace{SJ_{j_k} \dots J_{j_{k+1}-1}}_{\text{Batch k}} SJ_{j_{k+1}} \dots J_n$$

where J_{j_k} is the first job in the k −th batch and S is the setup time. Note that $j_1 < j_2 < \cdots \leq n$, where $j_1 = 1$. The problem of minimizing the total flow time reduces to a shortest path problem.

Let us introduce a dummy job J_{n+1}. Every job $J_i \in \{J_1,\ J_2, \dots J_n,\ J_{n+1}\}$ corresponds to node i, and every batch $(SJ_{j_k}, \dots J_{j_{k+1}-1})$ for some $k \geq 1$ corresponds to an arc (j_k, j_{k+1}) with weight $C_{j_k,\, j_{k+1}}$. Thus, every solution of the scheduling problem corresponds to a path in the form $j_1 - j_2 - \cdots - (n+1)$ (see Figure 2.6).

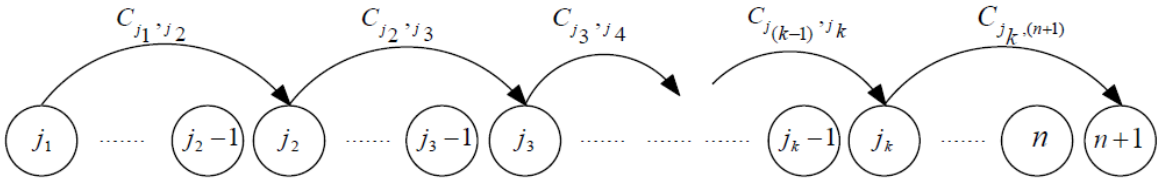


Figure 2.6: A network structure for the SMBSP

Arc weight $C_{i,j}$ is computed as follows:

$$C_{i,j} = \mathrm{S} \times [(\mathrm{n}+1) - \mathrm{i}] + \sum_{l=i}^{j-1}(l-i)p_l \ . \qquad (6)$$

Let $G(i)$ be the length of the shortest path from node i to $(n+1)\ \forall\ i = 1,2,\dots n$. $G(i)\ \forall i = 1,2,\dots,n$ can be computed using Equation 1 and substituting $T = n$. Once a shortest path is computed from node 1 to $(n+1)$, the minimum flow time can be obtained as follows:

$$\mathbb{F}^* = G(1) + \sum_{l=1}^{n}(n-l+1)p_l.$$

Every arc (i,j) of a shortest path corresponds to a batch $(J_i \dots J_{j-1})$.

Now we discuss how the new algorithm (Algorithm 1) can be adopted for this batch scheduling problem.

Let $a(k)$ be the advantage of node $(k+2)$ over node $(k+1)$ as a successor of k.

By Definition 1, $a(k) = C_{k,k+1} + G(k+1) - C_{k,k+2} - G(k+2) = \mathrm{S} \times [(\mathrm{n}+1) - \mathrm{k}] + G(k+1) - \mathrm{S} \times [(\mathrm{n}+1) - \mathrm{k}] - \sum_{l=k}^{k+1}(l-k)p_l \ - G(k+2) = G(k+1) - G(k+2) - p_{k+1}$.

Let $b(k)$ be the rate by which the advantage of node $(k+2)$ decreases over node $(k+1)$ as a successor when it is searched from node $(k-1)$ instead of k.

Therefore, $b(k) = \delta(k, k+1) - \delta(k-1, k+1)$.

Using Definition 2, $b(k) = C_{k,k+1} + G(k+1) - C_{k,k+2} - G(k+2) - C_{k-1,k+1} - G(k+1) + C_{k-1,k+2} + G(k+2)$. After cancelling the common terms, $b(k) = C_{k,k+1} - C_{k,k+2} - C_{k-1,k+1} + C_{k-1,k+2}$.

Using Equation 6, $b(k) = \mathrm{S} \times [(\mathrm{n}+1) - \mathrm{k}] + \sum_{l=k}^{k}(l-k)p_l - \mathrm{S} \times [(\mathrm{n}+1) - \mathrm{k})] - \sum_{l=k}^{k+1}(l-k)p_l - \mathrm{S} \times [(\mathrm{n}+1) - (\mathrm{k}-1)] - \sum_{l=k-1}^{k}(l-k+1)p_l + \mathrm{S} \times [(\mathrm{n}+1) - (\mathrm{k}-1)] + \sum_{l=k-1}^{k+1}(l-k+1)p_l$. After cancelling the common terms, rearranging, and simplifying, $b(k) = -p_{k+1} + 2p_{k+1} = p_{k+1}\ \forall 1 \leq k \leq n-1$.

With the above changes to $a(k)$ and $b(k)$, Line 4 of Algorithm 1 is replaced by $a(k) = G(k+1) - G(k+2) - p_{k+1}$ and $b(k) = p_{k+1}$.

Now, we can use Algorithm 1 to determine the optimum batch size by substituting the input parameters $T = n,\ d_k = p_k, f_k = S\ \forall k = 1,2, \ldots n,$ and $h = 1$.

2.5 A sample illustration of the developed algorithm:

This section explains the new algorithm, with numerical examples for the ELSP and SMBSP. Table 2.2 displays the input data for the ELSP, and Table 2.3 shows the corresponding results. Tables 2.6 and 2.7 present the input data for the SMBSP and the results obtained from the implementation of the new algorithm, respectively.

Table 2.2: Input Data (h=1)

k	1	2	3	4	5	6	7	8	9	10	11	12
d_k	69	29	36	61	61	26	48	67	45	67	79	56
f_k	85	102	102	101	98	114	105	86	119	110	98	114

Table 2.3: Results of Algorithm 1

k	$a(k)$	$b(k)$	u	$L(k)$	δ	$head$	$tail$	i^*	$S^*(k)$	$G(k)$
12	-	-	-	-	-	-		11	13	114
11	58	56	2	-	-	-		11	13	154
10	-39	79	0	10	-39,-37	10		9	11	264
9	43	67	1	11	-	-		9	11	340
8	31	45	1	9	-24	-		8	10	395
7	-12	67	0	7,8	-14, -12	8,7		6	8	500
6	57	48	2	-		-		6	8	557
5	31,89	26,87	2	-		-		6	8	615
4	-3	61	0,1	4,6	-39, -3, 5	6,4	5	5	7	714
3	38	61	1	5	-85	5		3	5	778
2	28	36	1	3	-23	3		2	4	852
1	45	29	2	2	-8	2		1	3	892

The first column of every row of Table 2.3 contains period k for which $G(k)$ is calculated. The second, third, fourth, and fifth columns show the corresponding calculation of $a(k)$, $b(k)$, u and the starting point for the beginning of the corresponding iteration, respectively. The sixth, seventh, and eighth columns show the value of δ, $head$, and $tail$, respectively. The ninth column shows the best diagonal for each iteration, and the tenth column shows the optimum

node for that period. To get the shortest path, we add 2 to i^*(see Theorems 1 and 2). The optimum policy is to produce in periods 1, 3, 5, 8, 10, and 11, and the total cost for this policy is 892.

Table 2.4: Matrix A

k	1	2	3	4	5	6	7	8	9	10	11	
$b(k)$	29	36	61	61	26	48	67	45	67	79	56	j
$a(k)$	45	28	38	-3*	31	57	-12	31	43	**-39**	58	0
1	**-8**	**-23**		**5***			**-14**	**-24**		**2**		1
2			**-21**	**-39**								2
3												3
4												4
5												5
6												6
7												7
8												8
9												9
10												10
11												11

Diagonal number

*cells $A_{0,4}$ and $A_{1,4}$ form a *stack*.

According to the algorithm, the initial value of $i^* = 11$. For k $= 11$, u $= 2$ and L$(11 - 2) =$ L$(9) = \{11\}$. This means $A_{2,9}$ (highlighted in grey in Table 2.4) is the first cell in diagonal 11, which is negative where 11>9 and $A_{1,10} > 0$ (Definition 7). From Table 2.5, which shows Matrix B, we see that the largest cell in columns 1…9 does not belong to diagonal 11. Thus, the algorithm eliminates this diagonal from this point for searching for the best diagonal. For $k = 10$, $a(k) < 0$. Hence, $u = 0$ and $L(10) = \{10\}$. According to the algorithm, $head = 10$, $\delta = -39 < 0$, and 10 $< i^*$(=11). Thus, the While loop in Line 10 is executed, and $S(P(10)) = S(9) = 11, P(S(10)) = P(11) = 9$. This eliminates diagonal 10. Line 14 calculates the cumulative sum ($\delta = -39 + 2 = -37$), and the Else condition in Line 22 sets $i^* = P(head) = P(10) = 9$. For k $= 9$, $p = 11$ ($\in L(9)) > i^*$. Thus, the For loop in Line 7 of the algorithm does not run, and the i^* remains unchanged. The procedure continues similarly. When $k = 4$, $L(4) = \{6,4\}$, let $p = 6 = i^*$ and

$head = 6,$ so i^* will change to $P(head) = P(6) = 5$. Matrix B in Table 2.5 shows that the largest cell in column 4 is located in the fifth diagonal. Now, let $p = 4(\in L(4)) < i^*$, $\delta = -3$. This satisfies the condition of Line 11, so $\delta = 5$. According to Definition 8 (highlighted in Table 2.4), cells $A_{0,4}$ and $A_{1,4}$ of Matrix A form a stack because $\sum_{p=0}^{1} A_{p,4} = -3 + 5 = 2 > 0$. Therefore, $tail = 5$ (Definition 9), and $a(5) = 89$ and $b(5) = 87$ are updated according to Theorem 4. The algorithm evaluates only the cells that are shown in bold letters in Table 2.4. Thus, the algorithm finds the largest cell of each column of Matrix B without calculating any entry of Matrix B and calculating entries of Matrix A as needed.

Table 2.5: Matrix B

k	1	2	3	4	5	6	7	8	9	10	11	
$b(k)$	29	36	61	61	26	48	67	45	67	79	56	j
$a(k)$	45	28	38	-3	31	57	-12	31	43	-39	58	0
1	37	5	-26	2	40	-22	-26	7	-75	-37		1
2	-47	-120	-47	-37	-106	-81	-117	-190	-129			2
3	-233	167	-134	-250	-210	-239	-393	-300				3
4	-306	-302	-414	-399	-435	-594	-559					4
5	-489	-649	-608	-691	-869	-816						5
6	-903	-888	-967	-1204	-1147							6
7	-1187	-1314	-1559	-1538								7
8	-1680	-1985	-1949									8
9	-2430	-2431										9
10	-2932											10
11												11

Diagonal number

Table 2.6: Input data (S = 10) for an example of the SMBSP

k	1	2	3	4	5	6	7	8	9	10	11	12
p_k	12	6	8	10	14	11	19	1	15	2	17	12

Table 2.7: Results of Algorithm 1

k	$a(k)$	$b(k)$	u	$L(k)$	δ	$head$	$tail$	i^*	$S^*(k)$	$G(k)$
12	0	0	-	-	-	-	-	11	13	10
11	-2	12	0	-	-2	11	-	10	12	30
10	3	17	1	10	-	-	-	10	12	57
9	25,40[§]	2, 17[§]	12, 2	11	-14	10	-	9	11	72
8	0	15	0	9	0, 23[*]	8	9	9	11	99
7	26, 54[§]	1, 20[§]	12, 1	8,7	-	-	-	9	11	127
6	9	19	1	-	-11	9	-	7	9	163
5	25	11	3	-	-	6	7	7	9	
					10,14[*]					204
4	27	14	2	6,4	-6	7	-	5	7	253
3	39	10	4	5	-	-	-	5	7	298
2	37	8	5	3	-8,-1	5, 4	-	3	5	342
1	38	6	7	2	-	-	-	3	5	376

[*] δ =23 and δ =14 are obtained by running the inner While loop of the algorithm.

[§] $a(9) = 40$ and $b(9) = 17$ are updated according to Line 16 of Algorithm 1 because a $tail(= 9)$ is found when $k = 8$. This forms a $stack$ with $head = 8$ and $tail = 9$. Hence, $i^*(= 9)$ remains unchanged. Similarly, when $k = 5$, a $tail(= 7)$ is found and $a(7) = 54$ and $b(7) = 20$ are updated (Theorem 4).

The optimum policy is to produce in periods 1, 5, 9, 11, and 12, and the total cost for this policy is 376.

2.6 Comparison with state-of-the-art algorithms:

This section presents a numerical experiment of the new algorithm's performance. Algorithm 1 is implemented using Fico's Mosel (Xpress) modeling language. All test instances

are run on a PC with an Intel Core i7 3.4 GHz processor and 8 GB of RAM. To compare the efficiency of Algorithm 1, it is compared with $O(T)$ time algorithms developed by Wagelmans et al. (1992), Aggarwal and Park (1993), and Albers and Brucker (1993), respectively, which are also coded using Mosel modeling language.

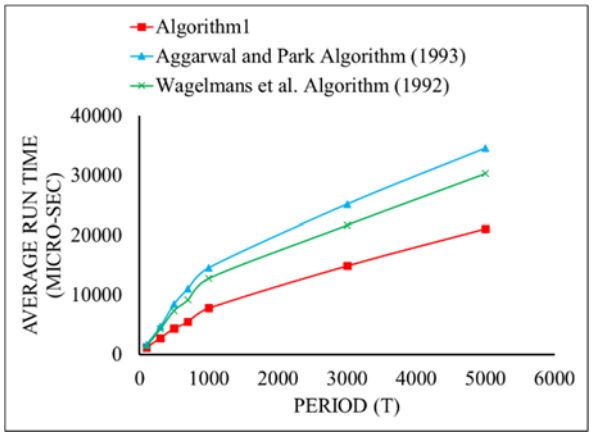
The performance of the new algorithm is tested using several data sets with different demand patterns, including random demand data; demand with positive trend; demand with seasonality effect; and demand with trend, seasonality, and variability effects. Both time-variant and fixed setup costs over the planning horizon are used. The size of the test instances is increased as $= 100, 300, 500, 700, 1000,\ 3000,$ and 5000. For each case, 30 instances are generated, and the average CPU processing time and the standard deviation of the run times are observed. Table 2.8 presents the result of the experiment for an inventory replenishment problem. CPU time increases linearly as T is increased for all test data sets. However, Table 2.8 indicates that Algorithm 1 shows a performance improvement with respect to CPU time of a maximum of 40.54% and 51.40% and an average of 29.84% and 39.27% when compared with the Wagelmans et al. (1992) and Aggarwal and Park (1993) algorithms, respectively. Figure 2.7 compares the three algorithms for the data sets with (a) random demand; (b) increasing linear trend; (c) seasonality; and (d) increasing linear trend, seasonality, and variability effects. For all cases, setup costs are time variant and holding costs are fixed. Figure 2.7 illustrates that CPU time increases linearly and the standard deviation (SD) remains almost stable as the problem size increases.

Algorithm 1 executes the "If" statements fewer times than the other two algorithms for all test data sets. The Wagelmans et al. (1992) algorithm runs the "If" statements exactly $3T$ times for all test instances. Furthermore, the Wagelmans et al. (1992) algorithm uses a "List," as we do, and we track the number of times the list operations (insert/delete elements in/from the list) are

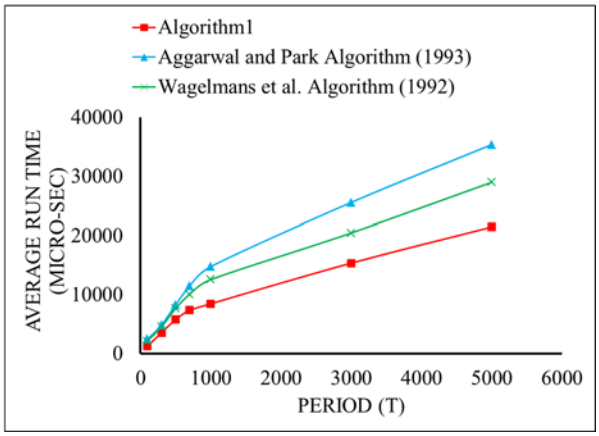
performed. Algorithm 1 has fewer list operations than the Wagelmans et al. (1992) algorithm. The Aggarwal and Park (1993) algorithm uses a matrix instead of the list data structure. Therefore, we compare the number of times their algorithm needs to evaluate the value of a particular matrix cell with the number of times our algorithm computes the same. In every metric of comparison, Algorithm 1 shows a better result than the others, proving its competitiveness.

Albers and Brucker (1993) develop a linear-time algorithm for an SMBSP. The developed Algorithm 1 performs better than Albers and Brucker's (1993) algorithm (see Figure 2.8). The performance of Algorithm 1 is compared by varying the number of jobs such that $n = 100, 300, 500, 700, 1000, 3000,$ and $5{,}000$. In each test case, 30 instances are generated, and the average and the SD of CPU time along with the number of times list operations (delete or insert) performed are observed. Table 2.9 shows the test results; Algorithm 1 shows an improvement in terms of CPU time of a maximum of 29.03% and an average of 25.75%, as well as fewer list operations, when compared with Albers and Brucker's (1993) algorithm.

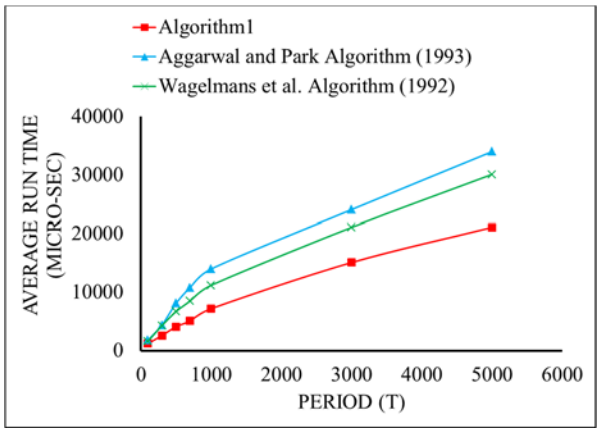
The data set used in this experiment is plotted against the number of periods in Figure 2.9. The demand is not stationary. Demand is considered with random data (Figure 2.9a); increasing linear trend (Figure 2.9b); seasonality with a pattern repeating every six periods (Figure 2.9c); and increasing linear trend, seasonality, and variability (Figure 2.9d).



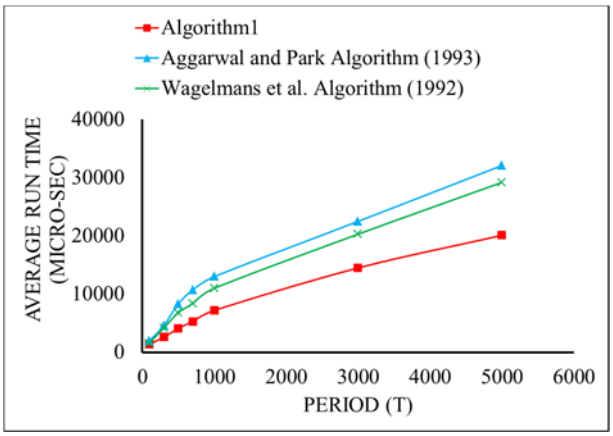
(a) Random demand



(b) Increasing linear trend



(c) Seasonality effect



(d) Increasing linear trend, seasonality, and variability effects in demand.

Figure 2.7: CPU time comparisons among the three algorithms for the ELSP

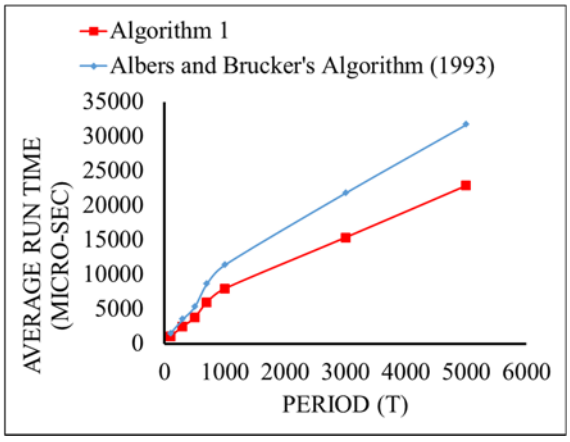


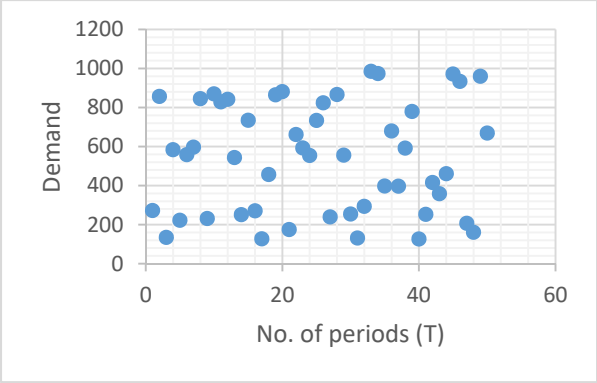
Figure 2.8: CPU time comparison between two algorithms for the SMBSP

Table 2.8: The result of the experiment for an inventory replenishment problem

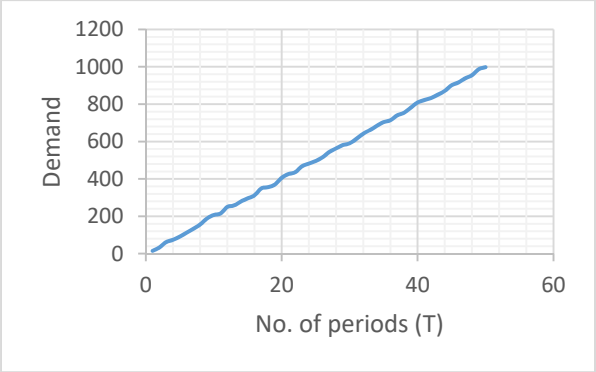
Demand Pattern	Instances	Period (T)	Average run time (μs)			SD (μs)			Number of times "If" statements (basic action) are executed			Number of times List operations (delete or insert) are performed		Number of times the value of a matrix cell is evaluated	
			Algorithm 1	Wagelmans et al. (1992) algorithm	Aggarwal and Park (1993) algorithm	Algorithm 1	Wagelmans et al. (1992) algorithm	Aggarwal and Park (1993) algorithm	Algorithm 1	Wagelmans et al. (1992) algorithm	Aggarwal and Park (1993) algorithm	Algorithm 1	Wagelmans et al. (1992) algorithm	Algorithm 1	Aggarwal and Park (1993) algorithm
Random	1	100	1200	1400	1700	422	516	483	222	300	320	113	137	112	217
	2	300	2800	4300	4700	422	483	675	663	900	932	331	400	330	625
	3	500	4400	7400	8500	699	516	527	1103	1500	1563	551	673	550	1018
	4	700	5500	9200	11100	707	632	738	1555	2100	2208	773	935	772	1429
	5	1000	7800	12800	14600	789	632	699	2219	3000	3156	1109	1342	1108	2032
	6	3000	14900	21700	25200	876	483	422	6668	9000	9285	3329	4029	3328	6001
	7	5000	21100	30300	34600	738	675	516	11116	15000	15456	5549	6715	5548	10011
Increasing linear trend	1	100	1400	1500	2400	516	527	516	232	300	330	116	150	116	215
	2	300	3700	4300	4900	675	483	738	711	900	937	373	455	373	617
	3	500	5900	6500	8400	738	707	699	1190	1500	1582	633	765	633	1043
	4	700	7500	8400	11400	527	699	516	1661	2100	2245	880	1062	880	1436
	5	1000	8600	11100	14700	516	738	483	2373	3000	3169	1256	1521	1256	2049
	6	3000	15500	21200	25500	527	789	527	7103	9000	9305	3751	4533	3751	6024
	7	5000	21700	30000	35300	675	943	675	11859	15000	15490	6251	7631	6251	10034
Seasonal effect	1	100	1300	1500	1700	483	527	483	253	300	335	130	181	134	227
	2	300	2700	4200	4300	483	422	483	754	900	952	388	545	392	629
	3	500	4200	6500	8100	422	707	876	1247	1500	1612	647	887	651	1056
	4	700	5200	8300	10700	632	675	483	1744	2100	2287	913	1245	917	1449
	5	1000	7300	11000	14000	675	667	816	2488	3000	3209	1300	1780	1304	2062
	6	3000	15200	21200	23900	632	632	876	7463	9000	9341	3828	5460	3830	6038
	7	5000	21400	29900	33600	516	876	699	12481	15000	15512	6429	9053	6435	10053
Seasonality, trend, & variation effect	1	100	1300	1500	1900	483	527	568	242	300	338	135	193	144	236
	2	300	2700	4100	4500	483	568	707	734	900	962	398	565	403	641
	3	500	4200	6500	7500	632	707	527	1235	1500	1635	652	893	668	1075
	4	700	5500	8500	10600	527	527	516	1732	2100	2315	929	1259	929	1461
	5	1000	7300	10900	12400	675	876	516	2464	3000	3222	1325	1792	1321	2078
	6	3000	14800	19700	22300	789	483	483	7458	9000	9356	3849	5479	3850	6053
	7	5000	20400	29200	31800	516	919	422	12356	15000	15539	6435	9071	6447	10074

Table 2.9: Result of the experiment for the SMBSP

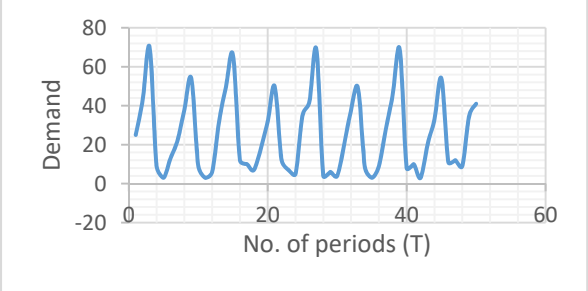
Instances	No. of Jobs (n)	Average run time (μs)		SD (μs)		Number of times List operations (delete or insert) are performed	
		Algorithm 1	Albers and Brucker (1993) algorithm	Algorithm 1	Albers and Brucker (1993) algorithm	Algorithm 1	Albers and Brucker (1993) algorithm
1	100	1200	1500	416	816	132	200
2	300	2600	3400	717	876	424	596
3	500	3900	5200	632	475	713	996
4	700	6100	8400	483	949	1053	1390
5	1000	8200	11300	522	522	1488	1995
6	3000	15400	21700	675	890	4391	5993
7	5000	22800	31600	516	675	7435	9986



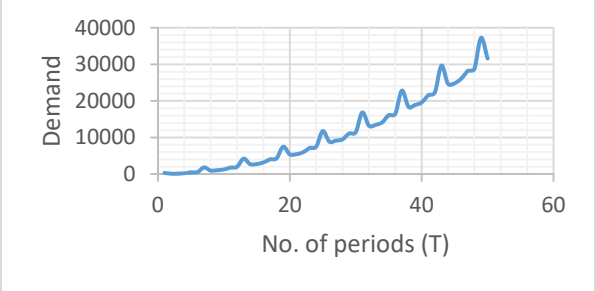
(a) Random demand



(b) Increasing linear trend



(c) Seasonality



(d) Increasing linear trend, seasonality, and variability

Figure 2.9: Demand data for the experiment

2.7 Conclusion:

This chapter presents a new linear-time algorithm for the ELSP and SMBSP, employing lists and stacks data structures. This approach is different from the famous and well-established linear-time algorithms by Wagelmans et al. (1992) (based on a geometric approach) and Aggarwal and Park (1993) (based on Monge arrays). The theoretical properties of Algorithm 1 are derived, and an experimental comparison to the algorithms developed by Aggarwal and Park (1993), Wagelmans et al. (1992), and Albers and Brucker (1993) is conducted. The results indicate that Algorithm 1 shows a performance improvement with respect to CPU time of a maximum of 40.54% and 51.40% and an average of 29.84% and 39.27% over the Wagelmans et al. (1992) and Aggarwal and Park (1993) algorithms, respectively. The developed algorithm is implemented for the SMBSP and shows an improvement of a maximum of 29.03% and an average of 25.75% over the Albers and Brucker (1993) algorithm. Moreover, Algorithm 1 executes the "If" statements (basic action) fewer times than Wagelmans et al. (1992) and Aggarwal and Park (1993) algorithms for all test data sets. The condition of the outer For loop in Algorithm 1 is checked exactly $(T - 1)$ times. The inner While loop is nested inside the inner For loop; if the inner For loop does not run, the inner While loop is not executed. The condition of the inner For loop in Line 7 of Algorithm 1 is checked at most $(2T - 4)$ times over all possible cases and runs at most $(T - 1)$ times. Most "If" statements are nested inside the inner For and inner While loops, which is why Algorithm 1 checks the "If" conditions fewer times than the comparable algorithms. Furthermore, Algorithm 1 performs fewer list operations than the algorithms by Wagelmans et al. (1992) and Albers and Brucker (1993). The number of matrix cells evaluated by Algorithm 1 is less than that in Aggarwal and Park (1993). By every metric of comparison, Algorithm 1 outperforms the other three algorithms. Algorithm 1, therefore, is faster.

REFERENCES

Aggarwal A, Park JK (1993) Improved algorithms for economic lot size problems. Oper. Res. 41(3):549-571. doi:10.1287/opre.41.3.549

Akbalik A, Rapine C (2012) Polynomial time algorithms for the constant capacitated single-item lot sizing problem with stepwise production cost. Operations Research Letters 40(5):390-397. doi:http://dx.doi.org/10.1016/j.orl.2012.05.003

Akbalik A, Rapine C (2013) The single item uncapacitated lot-sizing problem with time-dependent batch sizes: NP-hard and polynomial cases. European Journal of Operational Research 229(2):353-363. doi:http://dx.doi.org/10.1016/j.ejor.2013.02.052

Albers S, Brucker P (1993) The complexity of one-machine batching problems. Discrete Applied Mathematics 47(2):87-107. doi:http://dx.doi.org/10.1016/0166-218X(93)90085-3

Archetti C, Bertazzi L, Grazia Speranza M (2014) Polynomial cases of the economic lot sizing problem with cost discounts. European Journal of Operational Research 237(2):519-527. doi:http://dx.doi.org/10.1016/j.ejor.2014.02.044

Baki MF, Vickson RG (2003) One-operator, two-machine open shop and flow shop scheduling with setup times for machines and maximum lateness objective. INFOR : information systems and operational research 41(4):301-319.

Bitran GR, Yanasse HH (1982) Computational Complexity of the Capacitated Lot Size Problem. Management Science 28(10):1174-1186. doi:doi:10.1287/mnsc.28.10.1174

Chu C, Chu F, Zhong J, Yang S (2013) A polynomial algorithm for a lot-sizing problem with backlogging, outsourcing and limited inventory. Computers & Industrial Engineering 64(1):200-210. doi:https://doi.org/10.1016/j.cie.2012.08.007

Evans JR (1985) An efficient implementation of the Wagner-Whitin algorithm for dynamic lot-sizing. Journal of Operations Management 5(2):229-235. doi:http://dx.doi.org/10.1016/0272-6963(85)90009-9

Fazle Baki M, Chaouch BA, Abdul-Kader W (2014) A heuristic solution procedure for the dynamic lot sizing problem with remanufacturing and product recovery. Computers & Operations Research 43(Supplement C):225-236. doi:https://doi.org/10.1016/j.cor.2013.10.001

Federgruen A, Tzur M (1991) A Simple Forward Algorithm to Solve General Dynamic Lot Sizing Models with n Periods in 0(n log n) or 0(n) Time. Management Science 37(8):909-925. doi:10.2307/2632555

Florian M, Klein M (1971) Deterministic Production Planning with Concave Costs and Capacity Constraints. Management Science 18(1):12-20. doi:doi:10.1287/mnsc.18.1.12

Harris FW (1913) How Many Parts to Make at Once. The Magazine of Management 10(2):135-136,152.

Hellion B, Mangione F, Penz B (2012) A polynomial time algorithm to solve the single-item capacitated lot sizing problem with minimum order quantities and concave costs. European Journal of Operational Research 222(1):10-16. doi:http://dx.doi.org/10.1016/j.ejor.2012.04.024

Hsu VN (2000) Dynamic Economic Lot Size Model with Perishable Inventory. Management Science 46(8):1159-1169. doi:10.1287/mnsc.46.8.1159.12021

Hsu VN (2003) An Economic Lot Size Model for Perishable Products with Age-Dependent Inventory and Backorder Costs. IIE Transactions 35(8):775-780. doi:10.1080/07408170304352

Li X, Ishii H, Masuda T (2012) Single machine batch scheduling problem with fuzzy batch size. Computers & Industrial Engineering 62(3):688-692. doi:https://doi.org/10.1016/j.cie.2011.12.021

Mosheiov G, Oron D (2008) A single machine batch scheduling problem with bounded batch size. European Journal of Operational Research 187(3):1069-1079. doi:https://doi.org/10.1016/j.ejor.2006.01.052

Okhrin I, Richter K (2011) An O(T3) algorithm for the capacitated lot sizing problem with minimum order quantities. European Journal of Operational Research 211(3):507-514. doi:http://dx.doi.org/10.1016/j.ejor.2011.01.007

Retel Helmrich MJ, Jans R, van den Heuvel W, Wagelmans APM (2015) The economic lot-sizing problem with an emission capacity constraint. European Journal of Operational Research 241(1):50-62. doi:https://doi.org/10.1016/j.ejor.2014.06.030

Sargut FZ, Işık G (2017) Dynamic economic lot size model with perishable inventory and capacity constraints. Applied Mathematical Modelling 48(Supplement C):806-820. doi:https://doi.org/10.1016/j.apm.2017.02.024

Teksan ZM, Geunes J (2015) A polynomial time algorithm for convex cost lot-sizing problems. Operations Research Letters 43(4):359-364. doi:http://dx.doi.org/10.1016/j.orl.2015.03.009

van Hoesel S, Wagelmans A, Moerman B (1994) Using geometric techniques to improve dynamic programming algorithms for the economic lot-sizing problem and extensions. European Journal of Operational Research 75(2):312-331. doi:http://dx.doi.org/10.1016/0377-2217(94)90077-9

Wagelmans A, Hoesel SV, Antoon K (1992) Economic Lot Sizing: An O(n log n) Algorithm That Runs in Linear Time in the Wagner-Whitin Case. Operations Research 40:S145-S156. doi:10.2307/3840844

Wagner HM, Whitin TM (1958) Dynamic Version of the Economic Lot Size Model. Management Science 5(1):89-96. doi:10.1287/mnsc.5.1.89

Wang N, He Z, Sun J, Xie H, Shi W (2011) A Single-Item Uncapacitated Lot-Sizing Problem with Remanufacturing and Outsourcing. Procedia Engineering 15(0):5170-5178. doi:http://dx.doi.org/10.1016/j.proeng.2011.08.959

CHAPTER 3

A MODELING AND HYBRIDIZED DECOMPOSITION APPROACH FOR MULTI-LEVEL CAPACITATED LOT-SIZING PROBLEM WITH SETUP CARRYOVER, BACKLOGGING, AND EMISSION CONTROL

This chapter proposes a mixed integer linear programming model for the dynamic multi-level capacitated lot-sizing problem and the extension of this problem by allowing setup carryover, backlogging, and emission control. An item Dantzig-Wolfe decomposition technique is developed to decompose the problem into a number of uncapacitated dynamic single-item lot-sizing problems, which are solved by combining dynamic programming and a multi-step iterative capacity allocation heuristic approach. The capacity constraints are being taken into consideration implicitly through the dual multipliers, which are updated by a column generation procedure. Computational results show that the proposed optimization framework provides competitive solutions within a reasonable time frame.

3.1 Introduction:

There are a wide variety of models for production planning and inventory management. Lot-sizing problems involve determining the optimum production plan or inventory replenishment policy while minimizing the total cost of the system. Lot-sizing problems have attracted the attention of many researchers. Research is being undertaken to generalize the basic problem, which includes imposing limits on inventory and production capacity, as well as how to generalize across multiple product settings. The capacitated dynamic lot-sizing problem (CLSP) deals with the problem of determining time-phased

production quantities that meet given external demands and the capacity limits of the production system. The multi-level extension of the CLSP, known as Multi-Level Capacitated Lot-Sizing Problem (MLCLSP), deals with the production of multiple items when an interdependence among them at the different production levels is imposed by the product structure. The classical MLCLSP is introduced by Billington, McClain, and Thomas (1983), who describe a scenario in which the planning horizon is finite and divided into T discrete time periods (e.g., weeks). There are n items with period-specific external demands, which must be met without delay. The items are produced on m non-identical resources with limited period-specific capacities. The problem is to find an optimal production plan that minimizes production, setup, and inventory costs, and delivers optimal lot-sizes and production periods for each product. This problem forms the theoretical basis for material requirements planning (Buschkühl, Sahling, Helber, & Tempelmeier, 2010).

Setup operations are significant in some manufacturing industries and may strongly influence lot-sizing decisions. Setup operations prepare the processing units to manufacture production lots, consume production capacity (setup time) and incur setup costs. The classical CLSP assumes that setup of the resources for each item produced in each period is necessary. However, some researchers assume that setup state of a machine can be fully maintained over periods. In the literature (Briskorn, 2006) this is denoted as setup carryover. More specifically, setup carryover permits a setup state to be conserved between two consecutive periods. Haase (1998) points out that solutions change considerably when setup carry-over is considered.

Manufacturing industries are playing a key role in contributing to the prosperity and economic benefit of countries. As shown in Figure 3.1, these industries are responsible

for the emission of Greenhouse Gases (GHG) such as carbon dioxide (CO_2), nitrous oxide (N_2O) and methane (CH_4) often throughout the entire production process. Carbon is emitted directly from energy generation and the consumption of energy in setup, production, and inventory holding activities (X. Chen, Benjaafar, & Elomri, 2013). Recently there has been growing concern about the effect of these gases on climate change. Many countries are imposing various carbon regulatory mechanisms such as a carbon cap, carbon cap-and-trade, carbon cap-and-offset, and carbon tax to control the detrimental effects of carbon emissions on the environment. The governmental concern about emissions obliges the manufacturing industries to implement alternative environment-friendly production systems and invest in more energy efficient machines and facilities, and renewable energy sources, which are all costly practices for addressing the core problem. This has motivated many researchers to consider the environmental impact of emission by incorporating emission measures into the models for optimizing the production lot-size (Retel Helmrich, Jans, van den Heuvel, & Wagelmans, 2015).

In this chapter we present an item Dantzig Wolfe (DW) decomposition of the classical MLCLSP. We then extend the MLCLSP by allowing set-up carryover and backlogging. We also include emission capacity constraints and refer the problem as MLCLSP with Set-up Carryover, Backlogging and Emission control (MLCLSP-SCBE). We develop a Mixed Integer Linear Programming (MILP) model for the MLCLSP-SCBE and apply item DW decomposition of the proposed MILP formulation with an embedded Column Generation (CG) procedure. We propose a dynamic programming approach to solve each of the sub-problems and develop a Capacity Allocation (CA) heuristic to generate feasible solutions. An Integer Linear Programming (ILP) model is proposed to

determine the optimal setup carryover plan for a given production schedule. The solution approach is hybridized with an LP-based improvement procedure to refine the solution, thereby improving the solution quality given by the DW decomposition.

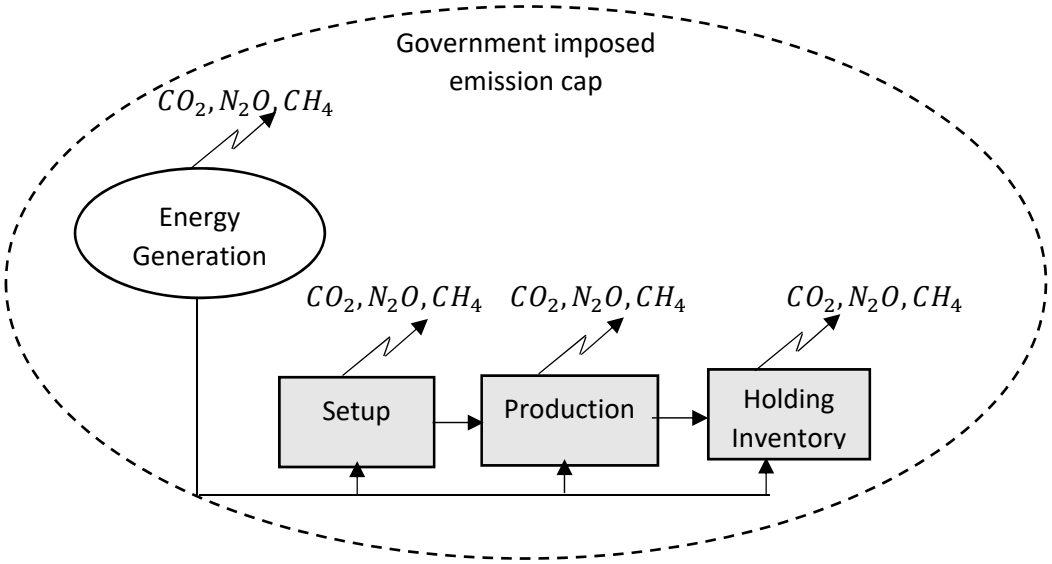


Figure 3.1: Greenhouse gas emission from the different production activities

The remainder of this chapter is organized as follows: In Section 3.2 the related literature is discussed. In Section 3.3 we present the problem statement along with the formulation of mathematical model. The proposed DW decomposition heuristic method is described in Section 3.4. Numerical results are discussed in Section 3.5. Finally, we conclude in Section 3.6.

3.2 Literature Review:

The MLCLSP has received much attention from researchers. Sahling et al. (2009) presents the MLCLSP as an extension of the single-level CLSP. An excellent review on MLCLSP formulations along with the solution approaches are presented by Buschkühl et al. (2010). Since the MLCLSP belongs to the class of NP-hard problems (Maes, McClain,

& Van Wassenhove, 1991), the application of heuristics and metaheuristics are the most common solution strategies for the MLCLSPs. Tempelmeier and Derstroff (1996) apply Lagrangean Relaxation (LR) to decompose the MLCLSP into several Single-Item Uncapacitated Lot-Sizing Problem (SIULSP) to obtain the lower bounds and propose a heuristic finite scheduling approach to find the upper bounds. Berretta and Rodrigues (2004) present a memetic algorithm for the MLCLSP with general product structures, setup costs, setup times. Later, Berretta et al. (2005) include non-zero lead time to the MLCLSP and approach the problem with a hybrid Simulated Annealing (SA) based Tabu Search (TS) method. Pitakaso et al. (2006) develop an ant based hybrid algorithm to solve the MLCLSP.

In many practical scenarios, the demand due date is not essentially met and backlogging can happen to avoid over time (Kimms, 1997). However, this may lead to the risk of stock out and loss of customer's goodwill. Moreover, quite typically penalty costs are usually accompanied with tardiness and backlogging. Toledo, de Oliveira, and Morelato França (2013) include backlogging in an MLCLSP and combine a multi-population-based metaheuristic with Fix-and-Optimize (FO) and mathematical programming techniques. Wu et al. (2011) propose two new mixed integer programming (MIP) models for MLCLSP problems with backlogging. They also develop hybrid exact methods and heuristics framework to solve the problem. Toledo et al. (2013) propose a hybrid mechanism, which combines a multi-population hierarchically-structured genetic algorithm and a FO heuristic method to solve the MLCLSP with backlogging. Zhao, Xie, & Xiao (2012) combine Variable Neighborhood Decomposition Search (VNDS) and accurate MIP to solve the MLCLSP. Later, Seeanner, Almada-Lobo & Meyr (2013)

hybridized the VNDS and the MIP-based FO approach as a new method for solving the MLCLSP.

Helber and Sahling (2010) introduce an iterative FO algorithm for the dynamic MLCLSP with positive lead times. They minimize the sum of setup, holding and overtime costs. Their approach solves a series of sub-problems where each sub-problem includes all the real-valued decision variables, but only a specific limited set of "free" binary variables. Later, Chen (2015) considers the same problem as Helber and Sahling (2010) and proposes an improved FO approach, which is more general and can be applied to other 0–1 MIP models.

Wu et al. (2013) propose an MIP formulation for modeling the MLCLSP with both backlogging and setup carryover. They present a progressive time-oriented decomposition heuristic framework that use Relax and Fix (RF) algorithm. Almeder, Klabjan, Traxler, and Almada-Lobo (2015) consider lead times and provide two formulations for the MLCLSP; one considering batch production (units produced in a batch can only be available when the processing of the whole batch is completed) and the other allowing for lot-streaming (allowing units to be transformed further on as soon as they are released). Boonmee and Sethanan (2016) study the MLCLSP for the poultry industry and develop an MIP model restricting the maximum lot-size for each time period. They apply Particle Swarm Optimization (PSO) to solve larger instances of the problem.

Gopalakrishnan, Miller, & Schmidt (1995) consider setup carryover to formulate the single-level CLSP, with an assumption of identical setup costs and times for all items. Later, Mohan Gopalakrishnan (2000) relaxes the assumption of identical setup costs and

times and extends their model to incorporate item dependent setup times and costs. Haase (1998) address a CLSP, which limits the setup carryover to at most one period. Sox & Gao (1999) present a set of MILPs for a multi-item CLSP that incorporates setup carryover without restricting the number of products produced in each period. They provide a Lagrangian decomposition heuristic that quickly generates near-optimal solutions and propose a dynamic programming approach to solve N independent single-item sub-problems. Later, Briskorn (2006) revisits the problem addressed by Sox & Gao (1999). He identifies a flaw in the dynamic programming approach of Sox and Gao (1999) and provides the necessary correction to solve the subproblems optimally. Karimi, Ghomi, & Wilson (2006) formulate a MILP for a multi-item CLSP with setup carry-over and backlogging and use TS to solve the problem.

Tempelmeier and Buschkühl (2009) consider setup carryover in an MLCLSP and develop a Lagrangian heuristic. Sahling et al. (2009) extends the work of Tempelmeier and Buschkühl (2009) by incorporating multi-periods setup carryovers and propose an iterative FO approach to solve a series of MILPs. The main idea of their proposed approach is to fix a large number of binary setup variables and optimize only a small subset of these variables, together with the complete set of the inventory and lot-size variables. Oztürk & Ornek (2010) present a formulation of MLCLSP with setup carryover and backlogging. Setup carryover is also considered by Caserta, Ramirez, and Voß (2010) for a MLCLSP. They formulate an MILP model and present a math-heuristic algorithm to solve the problem.

In recent decades, there has been an increasing awareness about the environmental damage caused by the manufacturing activities. Many researchers are interested in incorporating issues such as energy consumption and carbon emissions into the lot-sizing

models. Lot-sizing with emission constraints was introduced by Benjaafar, Li, & Daskin (2013). They consider the capacity of the total emissions over the entire planning horizon and investigate the impact of different regulatory policies such as carbon tax, carbon cap and trade, and carbon offsets. Retel Helmrich et al. (2015) show that lot-sizing with emission constraints is NP-hard and propose several solution methods. Absi, Dauzère-Pérès, Kedad-Sidhoum, Penz, & Rapine (2013) propose periodic, cumulative, global and rolling carbon emission constraints for a single item uncapacitated lot-sizing problem. These constraints impose a maximum value not on the total carbon emission, but on the average carbon emission per product. They show that the periodic case is polynomially solvable, while the cumulative, global and rolling cases are NP-hard. Later, they (2016) extend the analysis for the periodic carbon emission constraint to the realistic case of a fixed carbon emission, show that this problem is NP-hard, and propose a pseudo-polynomial algorithm to solve it. In general, Benjaafar, Li, and Daskin (2013), Retel Helmrich et al. (2015), and Absi, Dauzère-Pérès, Kedad-Sidhoum, Penz, and Rapine (2013) do not handle the multi-item CLSPs and the corresponding models only consider the emission capacity constraints. Production capacity due to resource constraints are not typically used to optimize the operational decisions for their models.

A great variety of heuristic algorithms have been developed to tackle the intractable nature of the CLSPs. One of the well-recognized approaches for solving the CLSPs is DW decomposition heuristic, which is used in lot-sizing problems for finding improved lower bounds (Duarte & de Carvalho, 2015; Jans & Degraeve, 2004). The basic idea of DW decomposition is to divide the lot-sizing problem into smaller subproblems that are much easier to solve and a coordinating master problem to obtain a good approximation of the

overall problem. Most of the literature considers single-level CLSP with multi-item, multi-period and setup time. However, there has been insufficient evidence of the implementation of DW decomposition for the MLCLSP.

DW decomposition is applied for CLSP for the first time by Manne (1958), in which lot-sizing problems are decomposed by item. The objective is to find a convex combination of given single-item schedules, which keeps the capacity constraints of the original CLSP and leads to minimal cost. Jans and Degraeve (2004) propose DW decomposition by period and show that the period decomposition method can provide at least the same or better lower bounds than decomposition by item. Degraeve and Jans (2007) later claims that the decomposition method proposed by Manne (1958) has an important structural deficiency; imposing integrality constraints on the variables in the master problem do not necessarily give an optimal integer solution as only the production plans, which satisfy the zero inventory property (if production takes place in a period t, the beginning inventory for that period must be zero) can be selected. Jans and Degraeve (2007) therefore proposed a new DW reformulation and a Branch-and-Price (B&P) algorithm. Pimentel et al. (2010) compare between item and period DW decomposition of a multi-item CLSP and apply the

B&P algorithm to solve the decomposition models. Caserta & Voß (2012) propose the DW decomposition approach in a meta-heuristic frame work for the multi-item, multi-period CLSP with setup times. Duarte & de Carvalho (2015) provide a DW decomposition of a known formulation for a discrete Lot-Sizing and Scheduling problem (DSLP) with setup costs and inventory holding. They develop a B&P and CG procedure to solve the problem optimality. Araujo et al. (2015) study the CLSP with setup time and propose a period DW decomposition for the problem. They develop a subgradient-based hybrid

scheme that combines LR and CG to find promising lower bounds. Fiorotto et al. (2015) develop two hybrid algorithms that combine LR and DW decomposition and apply them to obtain the stronger lower bounds for the CLSP with multiple items, setup time and unrelated parallel machines. Table 3.1 chronologically presents some of the studies conducted based on the solution approach proposed, type, and properties of the lot-sizing problem.

Table 3.1: Proposed heuristic approaches for solving the capacitated lot-sizing problems

Reference	Problem Solved	Properties of the Problem	Solution Approach
Manne (1958)	SL	OT	DW
Billington, McClain, and Thomas (1986)	ML	ST	LR and B&B
Tempelmeier and Derstroff (1996)	ML	ST	LDH
Sox & Gao (1999)	SL	ST, SC	LDH
Jans and Degraeve (2004)	SL	MI, ST	DW, CG and B&B
Tempelmeier and Buschkühl (2009)	ML	MI, ST, SC	LDH
Pimentel et. al. (2010)	SL	MI, ST	DW and B&P
Caserta & Voß (2012)	SL	SM, ST	DW and CG
Wu et. al. (2013)	ML	SC, BL	RF
Gören & Tunalı (2015)	SL	SC, ST	GA and FO
Fiorotto et al. (2015)	SL	MI, ST, PM	LR and DW
Araujo et al. (2015)	SL	MI, ST	DW, LR and CG
*Chowdhury, Baki, Azab	ML	MI,ST,SC,BL,EC	DW, CA and CG

Abbreviation:

Problem Solved: SL = Single-Level, ML = Multi-Level, MI = Multi-Item, ST = Setup Time, SC = Setup Carryover, SDST = Sequence Dependent Setup Time, SM = Single Machine, OT = Over Time, BL = Backlogging, EC = Emission Control

Solution Approach: B&B = Branch and Bound, LR = Lagrangian Relaxation, LDH = Lagrangian Decomposition Heuristic, DW= Dantzig–Wolfe, CG = Column Generation, B&P = Branch and Price, GA = Genetic Algorithm, RF = Relax and Fix, FO = Fix and Optimize, CA = Capacity Allocation heuristic.

*This chapter

3.3 Problem Formulation and Decomposition method for Classical MLCLSP and MLCLSP with Setup Carryover, Backlogging and Emission control (MLCLSP with SCBE)

3.3.1 Classical MLCLSP Formulation

Let us consider a multi-level capacitated lot-sizing (MLCLSP) problem with several end products, each with dynamic external period demands over a finite planning horizon. Each item is produced on a single resource with finite period capacity. A setup incurred may cause setup cost as well as a setup time. The problem is to find production quantities, setup decisions and inventory levels in each time period that meet the demand requirements and limited capacity resources, taking into consideration the BOM structure while simultaneously minimizing the production, inventory, and machine setup costs. This is known as the classical MLCLSP which is first introduced by Billington et al. (1983). The following assumptions are made for the formulation of the MLCLSP.

1. The planning horizon is divided into T periods (usually shifts or days).
2. There are m resources with period-specific capacities.
3. n items (including end items and subassemblies) with dynamic external period demands are arranged in a general product/ process structure with a unique assignment of each item to a single resource.
4. Production cost is time varying and setup cost is fixed over time;
5. Setup is sequence independent;
6. Full demand occurs at the beginning of each period;
7. Every item is assigned to a single machine;

8. Each machine can do multiple items;
9. Beginning and ending inventory is zero;
10. Shortage is not permitted.

The formulation of the model is given as follows:

Model MLCLSP:

$$Min \sum_{j=1}^{n} \sum_{t=1}^{T} (P_{jt} X_{jt} + h_j I_{jt} + c_j Y_{jt}) \quad (1)$$

Subject to:

$$I_{jt} = I_{j(t-1)} + X_{jt} - D_{jt} - \sum_{k \in \Gamma(j)} a_{jk} X_{kt} \qquad \forall j, t \quad (2)$$

$$X_{jt} \leq Y_{jt} * M \qquad \forall\, i, j \in \varphi(i), t \quad (3)$$

$$\sum_{j \in \varphi(i)} (p_j X_{jt} + s_j Y_{jt}) \leq R_{it} \quad \forall\, i,\ t \quad (4)$$

$$I_{jt}, X_{jt} \geq 0 \quad \forall\, i, j, t \geq 1 \quad (5)$$

$$Y_{jt} \in [0,1] \quad \forall\, i, j \in \varphi(i), t \quad (6)$$

Indices:

t Planning period ($t = 1,2,3, \dots, T$)

i Resource index ($i = 1,2,3, \dots, m$)

j item index ($j = 1,2,3, \dots, n$)

The decision variables are as follows:

I_{jt} Inventory level of item j at the end of period t

X_{jt} Production quantity of item j in period t

$$Y_{jt} = \begin{cases} 1 & \text{if there is a setup for item } j \text{ on machine } i \text{ in period } t \\ 0 & \text{otherwise} \end{cases}$$

The parameters used are as follows:

a_{jk} Quantity of item j required to produce one unit of item k

D_{jt} External demand of item j in period t

h_j Holding cost of item j

c_j Setup cost for item j

s_j Setup time for item j

M A large number

I_{j0} Initial inventory level of item j

R_{it} Available capacity of machine i in period t (in time units)

$\Gamma(j)$ Set of immediate successors of item j based on BOM

P_{jt} Production cost per unit of finished item j at period t

$\varphi(i)$ Set of items that can be assigned to machine i

ω Set of end items (items with external demand only; backlogging allowed on these)

$\mu(j)$ Set of immediate predecessor of item j

$\rho(j)$ Set of m/c eligible for job j

p_j Processing time required to produce one unit of item j

Objective function (1) represents the setup, holding and production cost. Constraints (2) represent the standard lot-sizing inventory balance capturing BOM. Constraints (3) forces a setup of machine i for item j in case of production of item j in period t; otherwise, the minimization objective function forces a zero value for Y_{jt} in case of zero production of item j in period t (X_{jt}). Limited resource capacity is reflected by

constraints (4). Constraints (5) and (6) provide the logical binary and non-negativity conditions of the decision variables.

3.3.2 DW Decomposition of the Classical MLCLSP:

DW decomposition is a special technique used to solve linear programming and integer programming models. DW decomposition redefines a new set of variables by replacing the original variables with a convex combination of the extreme points of a subsystem. This technique has been effectively implemented in different contexts. For more details on such technique see, Vanderbeck (2000), and Vanderbeck & Savelsbergh (2006). Degraeve & Jans (2007) have presented a DW approach for the CLSP, addressing an important structural deficiency of the standard DW approach for the CLSP proposed by Manne (1958). In this Section, borrowing ideas from Degraeve & Jans (2007), A DW decomposition for the classical MLCLSP and a dynamic programming approach for solving the subproblems are presented.

Let U_j be the set of all production schedules (extreme points). For a production schedule $u \in U_j$, let X_{jt}^u be the quantity of item j produced in period t in production plan u; I_{jt}^u be the level of inventory of item j at the end of period t in production plan u and Y_{jt}^u be the setup decisions for item j produced in period t in production plan u.

If we apply a DW decomposition to the original model, the reformulated problem is called the master problem. Its decision variables represent the weight of the extreme points of the subproblems. In this decomposition, the solutions of the subproblems are production plans for a single-item. For a given product, each production plan specifies the

production periods and the production quantity along with the inventory level and setup decisions.

The master problem and the derived subproblems are given as follows:

Master Problem (MP_1):

$$Min \sum_{u \in U_j} \sum_{j=1}^{n} \sum_{t=1}^{T} (P_{jt} X_{jt}^{u} + h_j I_{jt}^{u} + c_j Y_{jt}^{u}) \lambda_{ju} \quad (7)$$

Subject to:

$$\sum_{u \in U_j} \sum_{j \in \varphi(i)} (p_j X_{jt}^{u} + s_j Y_{jt}^{u}) \lambda_{ju} \leq R_{it} \; \forall \; i, \; t \quad (8)$$

$$\sum_{u \in U_j} \lambda_{ju} = 1 \; \forall \; j \quad (9)$$

$$\lambda_{ju} \geq 0 \; \forall \; j, u \in U_j \quad (10)$$

Let λ_{ju} be the new decision variable representing the weight of the production plan u for item j. Let w_{it} and v_j be the dual variables with respect to constraints (8) and (9) respectively. The objective function in (7) minimizes the overall costs. Constraints (8) are the production capacity constraints, which ensure the combination of production plans to meet the available capacity in each period. Constraints (9) are the convexity constraints, which force the choice of a combination of production plans. Constraints (10) express the non-negativity constraints for the decision variables.

The master problem (MP1) has fewer number of constraints compared with the original model (MLCLSP). The number of constraints of MP_1 is $(mT + n)$ as opposed to $(n + 2nT + mT)$ in the original model (MLCLSP). As far as decision variables are concerned, model MP_1 has greater number of decision variables than that of model

MLCLSP. Because of the nature of the method and the growing number of decision variables, a set of finite number of variables can be initially generated and then solved and improve sequentially using the classical CG approach. CG begins by defining a restricted master problem that has only a subset of columns or production plans. In each iteration, the columns that price out favorably are included in the MP_1. The algorithm ends when no more columns price out favourably, providing the optimal solution. The decomposed subproblems for each end item $j|j \in \omega$ is as follows.

Subproblem ($SP1_{End}$):

$$Min \sum_{i=1}^{m} \sum_{t=1}^{T} \left[(P_{jt} - w_{it} p_j) X_{jt} + (c_j - w_{it} s_j) Y_{jt} + h_j I_{jt} \right] - v_j \quad \forall j \qquad (11)$$

Subject to:

$$I_{jt} = I_{j(t-1)} + X_{jt} - D_{jt} \qquad \forall t \qquad (12)$$

$$X_{jt} \leq \mathrm{M} * Y_{jt} \ \forall t \qquad (13)$$

$$I_{jt}, X_{jt} \geq 0 \quad \forall\, i \in \rho(j), t \geq 1 \qquad (14)$$

$$Y_{jt} \in \{0,1\} \quad \forall\, i \in \rho(j), t \geq 1 \qquad (15)$$

$SP1_{End}\ (11-15)$ is a single-item uncapacitated lot-sizing problem, which determine the production schedule of the end items with strictly external demand (i.e., no successors). After all end items are scheduled, the next item, $k|k \in \mu(j)$, is scheduled. The decomposed subproblems for all k are as follows:

Subproblem $\left(SP1_{Component}\right)$:

Min (11)

Subject to:

$$I_{kt} = I_{k(t-1)} + X_{kt} - \sum_{k' \in \Gamma(k)} a_{kk'} X_{k't} \quad \forall t \tag{16}$$

and (13)-(15) for $j = k$.

The internal demand of any item (successor requirement) is placed on the right-hand side of the constraint (16) because $\sum_{k' \in \Gamma(k)} a_{kk'} X_{k't}\ \forall t$ are the dependent demands for item k, due to the production of its successors j that have already been scheduled. Treating the internal demands as constants, $SP1_{Component}$ is equivalent to $SP1_{End}$, and hence, item k's production schedule can now be determined. Thus, a production schedule for all the items can be found for a given set of dual variables if the procedure is followed item-by-item in succession to make sure that all requirements resulting from the production of the successor items are calculated before scheduling an immediate predecessor. Equation (16) uses a sequential bill of material approach to pass successors' production requirements between levels. Although it does not guarantee optimality, this procedure will ensure a feasible solution to the full set of inventory constraints.

During the CG process, the subproblems are solved to evaluate if there are any production plans that could improve the objective function (7). Since, the subproblems can be effectively solved using the dynamic programming approach (Absi, Kedad-Sidhoum, & Dauzère-Pérès, 2011), for each subproblem, we apply dynamic programming recursion (Section 3.3.3) separately for both the end items and the component items to obtain a production plan. A Capacity Allocation (CA) heuristic approach (Section 3.4.2) is applied to obtain a feasible solution if the subproblems produce an infeasible solution. Otherwise, the master problem (MP_1) may become infeasible because the capacity constraints (8) may not be satisfied. The setup decision variable ($Y_{jt} \forall j, t$) obtained from the CA heuristic is

used as a parameter and the LP-based improvement procedure is applied to obtain an optimal X_{jt} and I_{jt} for the given $Y_{jt} \forall j, t$. If there exists at least one production schedule (X_{jt}, I_{jt}, Y_{jt}) that makes the reduced cost negative, it is added to U_j and MP_1 is solved to provide new dual values. If no new column with a negative reduced cost can be found, the optimal solution of MP_1 gives a lower bound for the original problem. The detailed outline of the procedure is reported in Section 3.4.1.

3.3.3 Dynamic Programming Recursion for the SP1 (DPR1):

Subproblems (SP1) can be solved efficiently using a dynamic programming algorithm. It is obvious that the DP algorithm will generate an optimal solution for SP1 because each of the uncapacitated single-item subproblems has a WW cost structure. Given $1 \leq t' \leq t \leq T+1$, let us assume that production in period t' satisfies demands in periods t'through $t-1$. Let $SC_{t'}^{j}$, $PC_{t'}^{j}(t)$ and $HC_{t'}^{j}(t)$ be the total setup, production and holding to satisfy demands in periods t'through $(t-1)$ by the production of item j in period t'.

$$\left.\begin{array}{c} SC_{t'}^{j} = c_j - w_{it} s_j \\ PC_{t'}^{j}(t) = (P_{jt} - w_{it} p_j) \sum_{r=t'}^{t-1} D_{jr} \\ HC_{t'}^{j}(t) = h_j \sum_{r=t'}^{t-1} (r-t)\, D_{jr} \end{array}\right\} \quad (17)$$

For $1 \leq t \leq T+1$, let $f_j(t)$ be the optimal cost of satisfying demand from period 1 through $t-1$. Defining $f_j(0) = 0$, the dynamic programming recursion for the problem SP1 is as follows:

$$f_j(t) = \min_{1 \leq t' \leq t-1} \{SC_{t'}^{j} + PC_{t'}^{j}(t) + HC_{t'}^{j}(t) + f_j(t'-1)\} \quad (18)$$

Wagner and Whitin (1958) propose an $O(T^2)$ time algorithm to solve the dynamic programming recursion in Equation (18). Subsequently, many researchers have worked to improve the time complexity of the Wagner-Whitin algorithm. Wagelmans et al. (1992), Aggarwal and Park (1993), and Chowdhury, Baki, & Azab (2018) propose linear time algorithm to solve the dynamic programming recursion of Equation (18).

3.3.4 MLCLSP with Setup Carryover, Backlogging and Emission control (MLCLSP with SCBE) Formulation:

The following assumptions for the MLCLSP with SCBE are made:

- If an item produced at the end of period t is continued at the beginning of the next period $t + 1$, no additional setup is required;
- Setup state can be carried over from one period to the next at most once;
- At the beginning of the planning horizon, machines are not setup for any job;
- A setup state is not lost if there is no production on a machine within a period;
- Backordering is allowed only for the end items;
- No backlog at the beginning of the planning horizon;
- There are no independent demands for component items.

Moreover, we account for carbon emissions generated by different activities of the firm such as production (e.g., Greenhouse gas emissions due to burning fossil fuels for energy, as well as certain chemical reactions necessary to produce goods from raw materials.), holding (emissions due to energy spent on storage) and setup (emissions due

to machine setup). A carbon emission regulatory mechanism is considered in which the total emissions due to all activities over the planning horizon cannot exceed a carbon cap imposed by a regulator.

To include setup-carryover, backlogging and emission constraints into the model formulation, some new sets of variables must be introduced. They are as follows:

b_{jt} Quantity back ordered for item j in period t

$$\alpha_{jt} = \begin{cases} 1 & \text{if the setup state of machine } i|\, j \in \varphi(i) \text{ at the end of period } t \text{ and at the beginning of period } (t+1) \text{ is item } j \\ 0 & \text{otherwise} \end{cases}$$

E_t Emission due to production, inventory and setup in period t

The following additional parameters are used for MLCLSP with SCBE.

$\hat{s}_j$ Carbon emission related to the setup of item j

$\hat{p}_j$ Total carbon emission related to the production of item j

$\hat{h}_j$ Carbon emission related to holding inventory of item j

C_{cap} Total allowable carbon emission cap

Model MLCLSP_SCBE:

$$Min \sum_{j=1}^{n} \sum_{t=1}^{T} (P_{jt}X_{jt} + h_j I_{jt} + c_j Y_{jt} + \beta_j b_{jt}) \quad (19)$$

Subject to:

$$I_{jt} = I_{j(t-1)} + X_{jt} + b_{jt} - b_{j(t-1)} - D_{jt} \forall j, t \,|j \in \omega \quad (20)$$

$$I_{jt} = I_{j(t-1)} + X_{jt} - \sum_{k\in\Gamma(j)} a_{jk}X_{kt} \quad \forall j,t \,|j \notin \omega \tag{21}$$

$$X_{jt} \leq \mathrm{M}(Y_{jt} + \alpha_{j(t-1)})\ \forall\ i,j \in \varphi(i),t \tag{22}$$

$$\sum_{j\in\varphi(i)}(p_j X_{jt} + s_j Y_{jt}) \leq R_{it}\ \forall\ i,\ t \tag{23}$$

$$Y_{jt} + \alpha_{j(t-1)} \ \leq 1\ \forall\ i,j \in \varphi(i),t \tag{24}$$

$$\sum_{j\in\varphi(i)} \alpha_{jt} = 1\ \forall\ t \geq 1,i \tag{25}$$

$$E_t = \sum_{j=1}^{n}\left(\hat{p}_j X_{jt} + \hat{h}_j I_{jt} + \hat{s}_j Y_{jt}\right) \quad \forall\ t \tag{26}$$

$$\sum_{t=1}^{T} E_t \leq C_{cap} \tag{27}$$

$$I_{jt}, X_{jt}, b_{jt} \geq 0 \quad \forall\ i,j,t \tag{28}$$

$$Y_{jt,}\ \alpha_{jt} \in \{0,1\} \quad \forall\ i,j \in \varphi(i),t \tag{29}$$

The complete MIP model is presented as the minimization of the objective function (19), subject to constraints (20)-(29). The objective function minimizes the total ordering, holding, setup and backlogging cost. Constraints (20) and (21) represent the inventory balance for those products that need to satisfy external and internal demands, respectively. Constraints (22) ensure that the production of item j takes place in period t only if there is a setup of the machine i for item j during that period (Y_{jt}= 1), or if the resource is already in the correct setup state at the beginning of that period ($\alpha_{j(t-1)}$= 1). Constraints (23) indicate that production cannot exceed the available capacity. Constraints (24) prevent recurrence of setup of item j in period t on machine i if the setup state of item j on machine i is carried over from the previous period. Constraints (25) state that a machine can carry only one setup state into the subsequent period. Constraint (26) computes the carbon emission due to production, inventory and setup for each period. Constraint (27) is

the total emissions capacity constraint, which states that the total emissions should not exceed the total available emission limit. Constraints (28) and (29) are, respectively, nonnegativity and integrality constraints on the variables.

3.3.5 DW decomposition for the MLCLSP with SCBE:

MLCLSP with SCBE can be decomposed into several single-item uncapacitated subproblems with backlogging and setup carryover along with a master problem with the production capacity, emission capacity and setup carryover constraints.

Let us introduce a more compact notation for the variables: $X^j = (X_{j1}, X_{j2}, \dots, X_{jT})$, $I^j = (I_{j1}, I_{j2}, \dots, I_{jT})$, $b^j = (b_{j1}, b_{j2}, \dots, b_{jT})$, $Y^j = (Y_{j1}, Y_{j2}, \dots, Y_{jT})$ $\alpha^j = (\alpha_{j1}, \alpha_{j2}, \dots, \alpha_{jT})$. Further we define X as the single-item lot-size polytope for each item j as follows:

$$X = \{X^j, I^j, b^j, Y^j, \alpha^j\} \; \forall j = 1,2, \dots, n$$

For a production plan u, let b_{jt}^u be constants of the quantity back ordered for item j in period t and let α_{jt}^u be constants showing whether the setup state of a machine for item j is carried over from period t to $(t+1)$. The subproblems and the restricted master problem are given as follows:

$$MP_2: \quad Min \sum_{u \in U_j} \sum_{j=1}^{n} \sum_{t=1}^{T} (P_{jt} X_{jt}^u + h_j I_{jt}^u + c_j Y_{jt}^u + \beta_j b_{jt}^u) \lambda_{ju} \tag{30}$$

Subject to:

$$\sum_{u \in U_j} \sum_{j \in \varphi(i)} (p_j X_{jt}^u + s_j Y_{jt}^u) \lambda_{ju} \leq R_{it} \; \forall \; i, \; t \tag{31}$$

$$\sum_{u \in U_j} \sum_{t=1}^{T} \sum_{j=1}^{n} \left(\hat{p}_j X_{jt}^u + \hat{h}_j I_{jt}^u + \hat{s}_j Y_{jt}^u\right) \lambda_{ju} \leq \; C_{cap} \tag{32}$$

$$\sum_{u\in U_j}\sum_{j\in\varphi(i)}\alpha_{jt}^{u}\lambda_{ju}=1\ \forall\ t\geq 0,i \tag{33}$$

$$\sum_{u\in U_j}\lambda_{ju}=1\ \forall\ j \tag{34}$$

$$\lambda_{ju}\geq 0\ \forall\ j,u\in U_j \tag{35}$$

Let λ_{ju} be the new decision variable representing the weight of the production plan u for item j. The objective function (30) minimizes the total cost of the production plans chosen for each item. Let w_{it}, γ, y_{it} and v_j be the dual variables with respect to (31), (32), (33) and (34) respectively. Constraints (31) are the production capacity constraints and (32) are the emission capacity constraints. The setup carryover assignment constraints are presented in (33). Constraints (34) are the convexity constraints which force the choice of a combination of production plans. Constraints (35) express the non-negativity constraints.

In the subproblem $SP2_{End}$, the objective function (36) minimizes the reduced cost. The subproblems contain the inventory balance constraints (37), machine setup (38), setup carryover constraints (39), the non-negativity (40), and integrality conditions (41). In this decomposition, the solutions of the subproblems are production plans. For a given product, each production plan indicates the production periods and the production quantity along with the inventory level, backlogging and setup carryover decisions. The decomposed subproblems for each end item $j|j\in\omega$ are as follows.

Subproblem $(SP2_{End})$:

$$Min\ \sum_{i=1}^{m}\sum_{t=1}^{T}\left[(P_{jt}-w_{it}p_j-\gamma\hat{p}_j)X_{jt}+(c_j-w_{it}s_j-\gamma\hat{s}_j)Y_{jt}-\alpha_{jt}y_{it}\right]+$$

$$\sum_{t=1}^{T}\left[(h_j-\gamma\hat{h}_j)I_{jt}+\beta_j b_{jt}\right]-v_j\quad\forall j \tag{36}$$

Subject to:

$$I_{jt} = I_{j(t-1)} + X_{jt} + b_{jt} - b_{j(t-1)} - D_{jt} \qquad \forall t \tag{37}$$

$$X_{jt} \leq \mathrm{M}(Y_{jt} + \alpha_{j(t-1)}) \; \forall \; i \in \rho(j), t \tag{38}$$

$$Y_{jt} + \alpha_{j(t-1)} \; \leq 1 \; \forall \; i \in \rho(j), t \tag{39}$$

$$I_{jt}, X_{jt}, b_{jt} \geq 0 \qquad \forall \; i \in \rho(j), t \geq 1 \tag{40}$$

$$Y_{jt}, \alpha_{jt} \in \{0,1\} \; \forall \; i \in \rho(j), t \geq 1 \tag{41}$$

$SP2_{End}$ (36-41) is a single-item uncapacitated lot-sizing problem and is solved by Dynamic Programming Recursion (DPR2) to determine the production schedule of the end items with strictly external demand (i.e., no successors). After all end items are scheduled, the next item, $k|k \in \mu(j)$, is scheduled. The decomposed subproblems for all k are as follows:

Subproblem $\left(SP2_{Component}\right)$:

$$Min \; \sum_{i=1}^{m} \sum_{t=1}^{T} [(P_{kt} - w_{it} p_k - \gamma \hat{p}_k) X_{kt} + (c_k - w_{it} s_k - \gamma \hat{s}_k) Y_{kt} - \alpha_{kt} y_{it}] +$$

$$\sum_{t=1}^{T} \left[(h_k - \gamma \hat{h}_k) I_{kt}\right] - v_k \quad \forall k \in \mu(j) \tag{42}$$

Subject to:

$$I_{kt} = I_{k(t-1)} + X_{kt} - \sum_{k' \in \Gamma(k)} a_{kk'} X_{k't} \quad \forall t \tag{43}$$

and (38)-(41) for $j = k$.

The internal demand of any item (successor requirement) is placed on the right-hand side of the constraint (43) because $\sum_{k' \in \Gamma(k)} a_{kk'} X_{k't} \; \forall t$ are the dependent demands

for item k due to the production of its successors j that have already been scheduled. Treating the internal demands as constants, $SP2_{Component}$ is equivalent to $SP2_{End}$, and hence, item k's production schedule can now be determined. Thus, a production schedule for all the items can be found for a given set of dual variables if the procedure is followed item-by-item in succession to make sure that all requirements resulting from the production of the successor items are calculated before scheduling an immediate predecessor. Equation (43) uses a sequential bill of material approach to pass successors' production requirements between levels. Although it does not guarantee optimality, this procedure will ensure a feasible solution to the full set of the inventory constraint.

The CG begins by creating an initial set of feasible columns for the master problem by fixing all the dual variables at a value of zero. The initial set of columns are obtained from the uncapacitated single-item subproblems. In this chapter, the Dynamic Programming Recursion 2 (DPR2) is used to solve the subproblems. However, it is possible that the production requirements of the items in a period may be greater than the available capacity. According to the theory of decomposition algorithms, updating the dual variables w_{it}, γ, y_{it} and v_j should take these infeasibilities into account; otherwise, the master problem (MP_2) becomes infeasible because the constraints (31)-(35) may not be satisfied. If demand for one item is greater than the capacity in a period, a split lot is required. Ramsay (1981) shows that a feasible solution is often not attainable because an uncapacitated lot-sizing problem does not split the lot-sizes between periods. To avoid infeasibility, we propose a CA heuristic (Section 3.4.2) to obtain a feasible setup plan. Since each of the SIULSP is solved individually, the setup carryover decisions per resource are not coordinated. At most one job can be carried over from one period to the next and if

an item is carried over from period t to $(t+1)$, this item must have been produced first in period $(t+1)$. Therefore, it is necessary to generate a feasible solution by incorporating setup carryover constraints to the solution of the single-item subproblems. In addition to that an Integer Linear Programming (ILP) model (Section 3.3.7) is developed to determine the setup carryover decision variables optimally with the objective of maximizing the savings vis-a-vis setup costs. An LP-based improvement procedure is applied to obtain an optimum production schedule for a given set of setup plans and setup carryover decisions. If there is a production schedule u that makes the reduced cost negative, it is added to U_j. Then the master problem is solved to provide new dual variables. If no new column with a negative reduced cost can be found, the optimal solution of the MP_2 returns a production plan for the original problem.

3.3.6 Dynamic Programming recursion (DPR2) for single-item subproblem withsetup carryover:

The DPR2 formulation uses a network representation of the single-item lot-sizing problems that integrates the setup status of machines into the state space. The MLCLSP with SCBE is shown in the graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ in Figure 3.2. Each node of the network can be represented by $\mathcal{N} = \{(t_1, t_2) | t_1 \geq t_2\}$, where index t_1 is the period of production and t_2 is the time period to start meeting the demand from. The arc $\mathcal{A} = \{(t_1, t_2), (t_3, t_4) | t_2 \leq t_1 < t_4 \leq t_3\}$ indicates the production of item j in period t_1 to satisfy demands in periods t_2 through $(t_4 - 1)$ and t_3 is the next period of production.

Properties 1 and 2 hold.

Property 1: There are nodes (t_1, t_2) with $t_2 \leq t_1 \; \forall \; t_1 = \; t_2, \ldots T-1$ and $t_2 = 1, \ldots T+1$.

Proof of property 1: If $t_2 > t_1$, which means that production at t_1 meets demand starting from period $t_2 | t_2 > t_1$ and the demand of periods $t_1, \ldots, (t_2 - 1)$ met by production at period $t_1' < t_1$. So, zero inventory property is violated at t_1. In this case a better production schedule is obtained by shifting the demand of periods $t_1, \ldots, (t_2 - 1)$ from period t_1' to t_1. This modified production schedule saves cost of holding demand of periods $t_1, \ldots, (t_2 - 1)$ without increasing any setup cost or any other costs. Therefore, $t_2 \leq t_1$ ■

Property 2: Now let us consider nodes (t_1, t_2) and (t_3, t_4), where $t_2 \leq t_1$ and $t_4 \leq t_3$. There is an arc (t_1, t_2) to (t_3, t_4) if and only if $t_1 < t_4$.

Proof of property 2: The arc from (t_1, t_2) to (t_3, t_4) means that production in t_1 is followed by production in t_3. Production in t_1 meets the demand of periods $t_2, \ldots, \; t_1, \ldots, (\; t_4 - 1)$ and production t_3 meets the demand of periods $t_4, \ldots, \; t_3$ and more. Therefore $t_1 < t_4$ ■

Observation 1: The number of arcs that can be eliminated from any node (t_1, t_2) to $(t_3, t_4) | t_2 \leq t_1 < t_4 \leq t_3$ is $\sum_{r=1}^{t_1} (T - t_r + 1)$.

Given $1 \leq \tau \leq \tau' \leq t < t' \leq t'' \leq T$, let us assume that production in period t satisfies demands in periods $t' \text{through } t''$ and it also satisfies the backlogged quantities from periods τ through τ'. Let SC_t^j , $PC_t^j(t', t'')$ and $HC_t^j(t', t'')$ be the total setup, production and holding to satisfy demands in periods $t' \text{through } (t'' - 1)$ and $BC_t^j(\tau, \tau')$

be the total backlogging cost to satisfy demands in periods τ through $(\tau' - 1)$ by the production of item j in period t. These cost functions can be defined as follows:

$$\left.\begin{array}{c} SC_t^j = c_j - w_{it}s_j - \gamma\hat{s}_j \\ PC_t^j(t', t'') = (P_{jt} - w_{it}p_j - \gamma\hat{p}_j)\sum_{r=t'}^{t''-1} D_{jr} \\ HC_t^j(t', t'') = (h_j - \gamma\hat{h}_j)\sum_{r=t'}^{t''-1}(r-t)\,D_{jr} \\ BC_t^j(\tau, \tau') = \beta_j \sum_{r=\tau}^{\tau'-1}(t-r)D_{jr} \end{array}\right\} \quad (44)$$

For $1 \le t_2 \le t_1 < t_4 \le t_3 \le T$, let $f_j\{(t_1, t_2), (t_3, t_4)\}$ be the total cost to satisfy demands in periods t_2 through $(t_4 - 1)$ by the production of item j in period t_1 and t_3 is the next production period.

$$f_j\{(t_1, t_2), (t_3, t_4)\}$$

$$= \begin{cases} SC_{t_1}^j + PC_{t_1}^j(t_2, t_4) + HC_{t_1}^j(t_2, t_4) \;\; if\; t_1 = t_2, \alpha_{jt_1} = 0 & (45) \\ SC_{t_1}^j + PC_{t_1}^j(t_2, t_2+1) + y_{it_1} + PC_{t_1+1}^j(t_2+1, t_4) + HC_{t_1+1}^j(t_2+1, t_4) & \\ \qquad if\; t_1 = t_2, t_2 < t_4 - 1 \;and\; \alpha_{jt_1} = 1, i \in \rho(j) & (46) \\ SC_{t_1}^j + PC_{t_1}^j(t_2, t_4) + HC_{t_1}^j(t_1, t_4) + BC_{t_1}^j(t_2, t_1) \quad if\; t_2 < t_1 < t_4 \;and\; \alpha_{jt_1} = 0 & (47) \\ SC_{t_1}^j + PC_{t_1}^j(t_2, t_1+1) + BC_{t_1}^j(t_2, t_1) + y_{it_1} + PC_{t_1+1}^j(t_1+1, t_4) + HC_{t_1+1}^j(t_1+1, & \\ \qquad if\; t_2 < t_1 < t_4 - 1 \;and\; \alpha_{jt_1} = 1, i \in \rho(j) & (48) \end{cases}$$

$t_1 = t_2$ in expression (45) represents a setup in period t_1 followed by the production for the demands of periods t_2 through $(t_4 - 1)$. This schedule does not have any setup carryover from period t_1 to $(t_1 + 1)$ and hence, $\alpha_{jt_1} = 0$. Expression (46) includes a schedule where there is a setup and production in period t_1 equal to the demand of period t_2, followed by carryover ($\alpha_{jt_1} = 1$) onto period $(t_2 + 1)$ and production in period $(t_2 + 1)$ equal to the demands of period $(t_2 + 1)$ through $(t_4 - 1)$. The case of $t_2 < t_1 < t_4$ and no carryover ($\alpha_{jt_1} = 0$) is addressed in expression (47) where the setup is done in period t_1 and the production in period t_1 amounts to the demands of periods t_2 though

$(t_4 - 1)$ considering the backlogged quantities of periods t_2 though $(t_1 - 1)$. Expression (48) indicates a schedule for production and setup in $t_1 | t_2 < t_1 < t_4 - 1$, production of demands of periods t_2 through t_1, along with the backlogged quantities of periods t_2 though $(t_1 - 1)$, setup carryover to the period $(t_1 + 1)$ and production in period $(t_1 + 1)$ equal to the demands of periods $(t_1 + 1)$ through $(t_4 - 1)$.

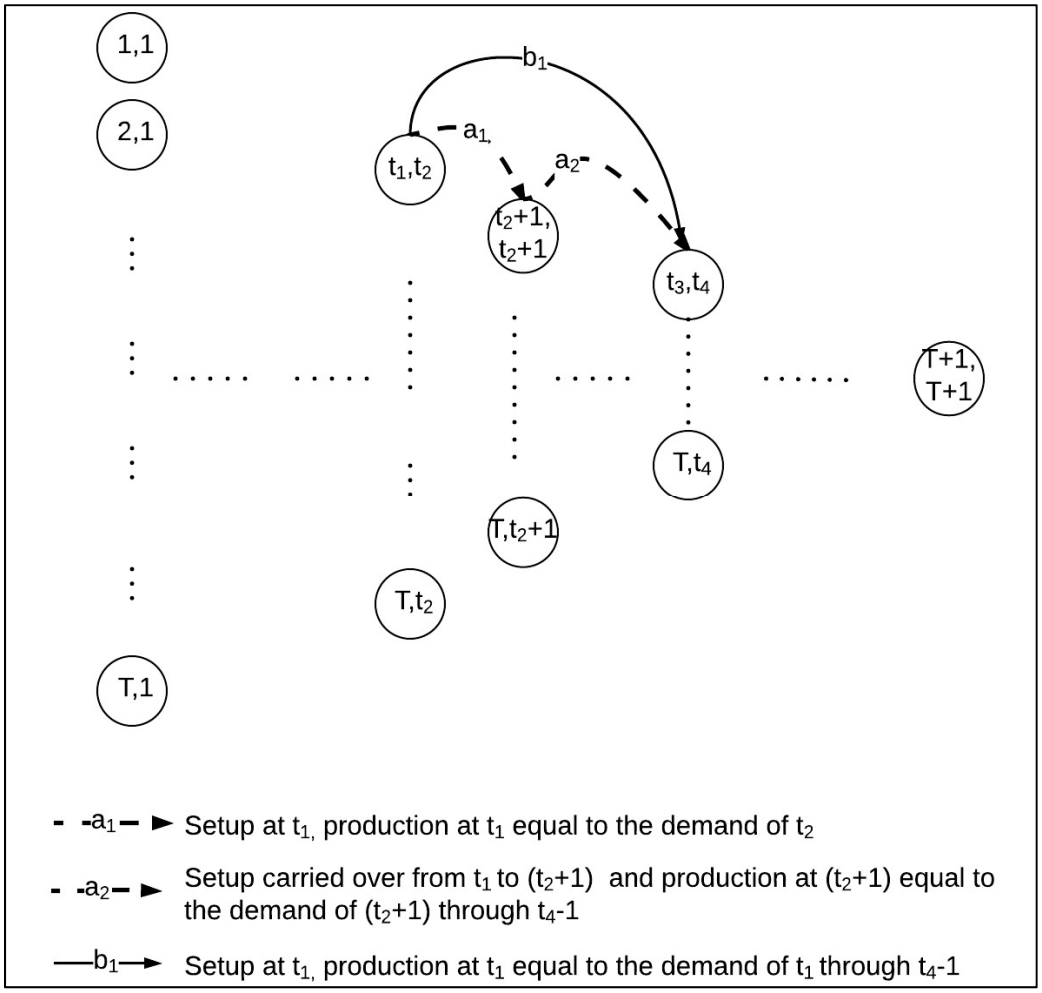


Figure 3.2: Shortest path network for the Subproblem

For $1 \leq k \leq T + 1$, Let $V_j(k)$ be the minimum cost of satisfying demand in periods 1 through $(k - 1)$ for item j Defining $V_j(1) = 0 \forall j$, we have the following DP recursion:

$$V_j(k) = \min_{1 \le k' \le t' < k \le t''} \{ V(k') + f_j\{(t', k'), (t'', k)\}\} \quad (49)$$

To analyze the computational complexity of recursion (49), it takes $O(T)$ time to obtain SC_t^j and $O(T^2)$ time to obtain $PC_t^j(t', t''), HC_t^j(t', t'')$, and $BC_t^j(\tau, \tau')$ for all $1 \le \tau \le \tau' \le t < t' \le t'' \le T$ from Equation (44). It is noted that after an $O(T^2)$ time preprocessing step, each $f_j\{(t_1, t_2), (t_3, t_4)\}$ where $1 \le t_2 \le t_1 < t_4 \le t_3 \le T$ can be evaluated in constant time via Equation (45) through (48). Once these values are available, $V_j(k) \forall 1 \le k \le T + 1$ can be obtained in $O(T^3)$ time.

3.3.7 Setup Carry over Assignment:

The problem of setup carryover assignment can be described as follows: If an item j is produced both in period t and $(t + 1)$ and a setup is performed in both periods, the second setup can be replaced by a setup carryover if the item is produced at the end of period t and at the beginning of period $(t + 1)$. This last condition can be fulfilled by only one item that is produced in both period t and $(t + 1)$. This saves both setup time and setup costs and such savings are attainable by only one item that is produced in both period t and $(t + 1)$.

An ILP model can be formulated for each machine to determine the setup carryover assignment variable. The objective of the problem is to maximize savings in setup cost. Suppose we are given $S(i, t) \forall i, t$, where $S(i, t)$ is the set of items produced in machine $i \forall i = 1, \ldots, m$ in period $t \forall\, t = 1 \ldots T$. Let us assume another set $S'(i, t) | S'(i, t) = S(i, t) \cap S(i, t + 1)\ \forall i = 1, \ldots, m$ and $t = 1 \ldots T - 1$. Each element of $S'(i, t)$ represents an item that can be carried over from period t to $(t + 1)$ to avoid the machine setup for

that item in period $(t+1)$. Since for a particular machine i, only one item can be carried over to the next period, we have to pick exactly one element from $S'(i,t)$. Let us introduce the parameters for the problem as follows:

c_j Setup cost saving associated with element $j | j \in S'(i,t)\ \forall t$

$$q_{jt} = \begin{cases} 1 & \text{if item } j \in S'(i,t) \\ 0 & \text{otherwise} \end{cases}$$

$$r_{jt} = \begin{cases} 1 & \text{if } q_{jt} = q_{j(t+1)} = 1 \text{ and } if\ |S'(i,t+1)| > 1 \\ 0 & \text{otherwise} \end{cases}$$

Decision variable:

$$z_{jt} = \begin{cases} 1 & \text{If item } j \in S'(i,t) \text{ is produced at the end of period } t \text{ and in the beginning of period } (t+1) \\ 0 & \text{otherwise} \end{cases}$$

Model SC:

$$Max\ \sum_{j \in \varphi(i)} \sum_{t=1}^{T-1} c_j\, z_{jt} \qquad \forall i \tag{50}$$

Subject to,

$$z_{jt} \leq q_{jt}\ \forall j, t < T \tag{51}$$

$$\sum_{j \in S(i,t) | q_{jt}=1} z_{jt} \leq 1 \quad \forall t \leq T \tag{52}$$

$$z_{jt} + z_{j(t+1)} \leq 1 \quad \forall j, t < T-1 | r_{jt} = 1 \tag{53}$$

$$z_{jt} \in \{0,1\}\ \forall j, t \tag{54}$$

The objective function to maximize the setup cost savings for all $i = 1..m$ is expressed in equation (50). Constraints (51) ensure that an item, which is produced in two consecutive periods, should be carried over to the next period. Constraints (52) state that at most one item can be carried over to the next period. But for some t, if $q_{jt} = 0 \forall j \in$

$S(t), \sum_{j \in S(i,t)} z_{jt} = 0$. Constraints (53) prevents the same item from being selected to carry over in two consecutive periods if $r(j,t) = 1$, which implies the condition that if item j is carried over from period t to $(t+1)$ then j cannot be carried over from $(t+1)$ to $(t+2)$. Finally the type of variables are defined in constraints (54). We determine the setup carryover variable α_{jt} by applying Procedure 1.

Procedure 1:

Input: z_{jt} , $S(i,t)$
Output: α_{jt}
Initialization: $\alpha_{jt} = z_{jt} \ \forall j,t$
Case 1: If $|S(i,t)| = 0$ then $\alpha_{jt} = \alpha_{j(t-1)}$
Case 2: if $|S(i,t)| = 1$ then $\alpha_{jt} = 1 | j \in S(i,t)$
Case 3: let ϵ =random number between 1 and $n \mid \epsilon \in S(i,t)$
if $|S(i,t)| > 1$ and $\sum_{j \in \varphi(i)} \alpha_{jt} = 0$ then $\alpha_{\epsilon t} = 1$

3.4 Proposed DW decomposition Heuristic Method

3.4.1 Outline of the solution procedure:

Model MLCLSP (**MLCLSP_SCBE**):

Step 1: Generate an initial set of solutions by applying the following procedure:

Step 1.1: From Equation (17) (**Equation (44)**) calculate $SC_{t'}^{j}, PC_{t'}^{j}(t)$, and $HC_{t'}^{j}(t)$ ($SC_{t}^{j}, PC_{t}^{j}(t',t''), HC_{t}^{j}(t',t'')$, and $BC_{t}^{j}(\tau,\tau')$) by fixing the dual variables w_{it} and v_j ($\boldsymbol{w_{it}, y_{it}, \gamma}$**, and** $\boldsymbol{v_j}$) a value of zero for the end items $j | j \in \omega$.

Step 1.2: Use $SC_{t'}^{j}, PC_{t'}^{j}(t)$, and $HC_{t'}^{j}(t)$ ($\boldsymbol{SC_t^j}$, $\boldsymbol{PC_t^j(t', t'')}, \boldsymbol{HC_t^j(t', t'')}$, **and** $\boldsymbol{BC_t^j(\tau, \tau')}$) as the input for DPR1 (**DPR2**) and obtain the optimal production quantity X_{jt} and setup decision Y_{jt} for item j in period t.

Step 1.3: Derive demand for the components $k|k \in \lambda$ as follows:

$$D_{kt} = \sum_{k' \in \Gamma(k)} a_{kk'} X_{k't} \; \forall t$$

Step 1.4: Repeat Steps 1.1 and 1.2 for the components. The planned production is exploded down to the immediate predecessor level.

Step 1.5: Apply a Capacity Allocation (CA) heuristic to make X_{jt} and Y_{jt} feasible (Section 3.4.2).

Step 1.6: For **MLCLSP_SCBE,** solve the ILP for maximizing setup cost savings (Equation (50)-(54)) and obtain the value of the setup carryover decision variable $\alpha_{jt} \forall j, t$ by applying Procedure 1 (Section 3.3.7).

Step 1.7: Use the Y_{jt} values from step 1.5 (**and $\boldsymbol{\alpha_{jt}}$ from step 1.6**) as parameters and solve model MLCLSP (**MLCLSP_SCBE**) to obtain an optimal value for X_{jt}, I_{jt} (**and $\boldsymbol{b_{jt}}$**).

Step 2: Solve the LP relaxation of the $MP_1 (MP_2)$and obtain the dual values of constraints (8) and (9) (**constraints (31) through (34)**).

Step 3: Solve the subproblems using the following approach:

Step 3.1: Use the dual values obtained from Step 2 and calculate $SC_{t'}^{j}, PC_{t'}^{j}(t)$, and $HC_{t'}^{j}(t)$ ($\boldsymbol{SC_t^j}$, $\boldsymbol{PC_t^j(t', t'')}$, $\boldsymbol{HC_t^j(t', t'')}$, **and** $\boldsymbol{BC_t^j(\tau, \tau')}$) by using Equation (17) (**Equation (44)**).

Step 3.2: Repeat Steps 1.2 through 1.7.

Step 4: If there exists at least one new column with negative reduced cost, add such columns to $MP_1(\boldsymbol{MP_2})$ and start from Step 2 again. Otherwise, stop.

3.4.2 Description of the Capacity Allocation (CA) Heuristic:

The pseudocode for the CA heuristic is given in Section 3.4.2, where the following symbols are used:

l	Index for levels of product hierarchy (from 0 for the end item to L).
$\pi(l)$	Set of items positioned in level l of the product hierarchy.
Q_{jt}	Production quantity for item j in period t obtained from WW solution (capacity constraint relaxed).
X'_{jt}	Production quantity for item j in period t obtained from CA heuristic.
Z'_{jt}	Allocated capacity for item j in period t in time units.
Y'_{jt}	Setup decision for item j in period t obtained from CA heuristic.
I'_{jt}	Inventory level of item j in period t obtained from CA heuristic.
$Req_{Cap(i,t)}$	Required capacity of machine i in period t in time units.
$Available_{Cap(i,t)}$	Available capacity of machine i in period t in time units.

t'	Last period before the next period of production obtained from the WW solution.
$(RQ)_j$	Remaining quantity of item j from the WW solution after the production quantity is adjusted in any period.
$(RD)_{j,t}$	Remaining demand of item j in period t that cannot be satisfied due to the limit of the capacity of resource $i \mid i \in \rho(j)$.
$Unused_{Cap_{(i,t)}}$	Unutilized capacity of machine i in period t.
$Allowable_{j,t}$	Allowable quantity of item j that can be allocated in period t.

The CA heuristic works as follows: The algorithm starts with $t = 1$ and $l = 0$. Let us consider an item $j \mid j \in \pi(l)$ and machine i that is responsible to produce j is currently overloaded in period t. This overload is decreased by shifting the production quantity of an item $j \mid j \in \varphi(i)$ into an earlier period or later period. The production quantity of item j is reduced according to the ratio of the allowable capacity and the required capacity of machine i in period t as shown in Equation (55). The production quantity of item j in period t is assigned using Equation (56).

$$Z'_{jt} = (Q_{jt} \times p_j + s_j) \times \frac{Available_{Cap(i,t)}}{Req_{Cap(i,t)}} \tag{55}$$

$$X'_{jt} = \max\left(\frac{Z'_{jt} - s_j}{p_j}, D_{jt} - I'_{j(t-1)}\right) \tag{56}$$

While decreasing the production quantity of any item, one has to remember that a reduction in the production quantity should not lead to backorders for this item resulting from successor item demands. That is why it is necessary to adjust the production quantity of the successor item. If there is no further item causing an overload of the resource in

question in the current level, then we will adjust the quantity of the successor items of the product hierarchy. For all direct and indirect successors j' of item j, the maximum quantity that can be decreased is determined according to Equation (57).

$$X'_{j't} = \max\left(D_{j't} - I'_{j'(t-1)} , \min_{j\in\mu(j')} \frac{X'_{jt}}{a_{jj'}} \right) \quad (57)$$

In the case where the sum of demands of all the items j produced in machine i in period t exceeds the available capacity of machine i in period t, we shift the production ($(RD)_{j,t} = D_{jt} - I_{j(t-1)} - X'_{jt}$) backward into period $\tau|\tau < t$ and $Unused_{Cap_{(i,\tau)}} > 0$. Shifting production to the earlier period is possible because the feasibility of the resulting problem instances with respect to the capacity constraints is maintained by ensuring that the cumulative capacity for every period is larger than (or equal to) the cumulative requirement. Because of this shifting to earlier period, the production quantity of item j in period τ increases. To accommodate the derived demand of the predecessor items j' of j, the production quantity of all $j'|j' \in \mu(j)$ is adjusted as follows: $X'_{j'\tau} = \max(X'_{j'\tau} , D_{j'\tau})$

If, for all $j|j \in \pi(l)$ and for all $i|i \in \rho(j)$, the available capacity of machine i in period t is allocated among all $j|j \in \varphi(i)$, then we move into the next level of the product hierarchy. When the production quantity of all items j is allocated according to the available capacity of machine i in period t, shift forward the remaining quantity $(RQ)_j$ to period $t'|t' > t$ and assign the production of item j in period t' as follows: $X'_{jt'} = min(Allowable_{jt'}, D_{jt'}, (RQ)_j)$. Update $(RQ)_j$. Next, shift the rest of the quantity backward for all $t' = t' - 1, .. t + 1$.

3.4.3 Pseudocode for the CA Heuristic:

Input: $Q_{jt}\ \forall j,t$
Output: $X'_{jt}, I'_{jt},\ Y'_{jt} \forall j,t$
$t = 1$
$While\ (t \leq T) do$
$Forall\ (l\ in\ 0..L)\ do$
$Forall\ (j\ \in \pi(l), i \in \rho(j))\ do$
$Req_{Cap(i,t)} = \sum_{k\in\varphi(i)}(X'_{kt}p_k + Y'_{kt}s_k)$ (58)
$Unused_{Cap_{(i,t)}} = \max(0, Available_{Cap_{(i,t)}} - Req_{Cap(i,t)})$
$While\ \left(Req_{Cap_{(i,t)}} >\ Available_{Cap_{(i,t)}}\right) do$
$Ratio = Available_{Cap_{(i,t)}}/Req_{Cap_{(i,t)}}$
$Forall\ (k\ \in \varphi(i))\ do$
$Z'_{kt} = (Q_{kt} \times p_k + s_k \times Y'_{kt}) \times Ratio$
$If\ \sum_{k'\in\varphi(i)}(D_{k't} \times p_{k'}) \leq Available_{Cap_{(i,t)}}\ then$
$X'_{kt} = \max\left(\left\lfloor \frac{Z'_{kt} - s_k}{p_k} \right\rfloor, D_{kt} - I'_{k(t-1)}\right)$
$Else$
$X'_{kt} = \left\lfloor \frac{Z'_{kt} - s_k}{p_k} \right\rfloor$
$(RD)_{kt} = D_{kt} - I'_{k(t-1)} - X'_{kt}$
Allocate unsatisfied demand to prior periods and update the production quantities of the predecessor items using **Procedure 2**
$End - If$
$Update\ Y'_{kt}\ and\ I'_{kt}$
$Forall\ \left(k' \in\ \mu(k)\right)\ D_{k't} = X'_{kt} \times a_{k'k}$
$End - do$
$Compute\ Req_{Cap(i,t)}$ using Equation (58)
Update production quantities of the successor items using **Procedure 3**
$Let, t' = last\ period\ before\ the\ next\ production\ and\ t' > t$
Allocate capacity from period t' backwards to period $(t+1)$ using **Procedure 4**
$Update\ Y'_{jt''}\ and\ I'_{jt''} \forall t < t'' \leq t'$
$t = t + 1$
$End - do$

Procedure 2:

Input: $(RD)_{kt}$ and $Unused_{Cap_{(i,\tau)}} \forall i \in \rho(k), \tau \leq t-1$
Output: $X'_{k\tau} \forall \tau \leq t-1$
$\tau = t-1$
$While\ (RD)_{kt} > 0\ then$
  $Forall\ \left(i' \in \rho(k) |\ Unused_{Cap_{(i',\tau)}} > 0\right)\ do$
    $X'_{k\tau} = X'_{k\tau} + \min(Unused_{Cap_{(i',\tau)}}, (RD)_{kt})$
    $(RD)_{kt} = \max(0, (RD)_{kt} - Unused_{Cap_{(i',\tau)}})$
    $l' = l+1$
    $While\ (l' \leq L)\ do$
      $Forall\ (j' \in \pi(l') | j' \in \mu(k))\ do$
        $D_{j'\tau} = X'_{k\tau} \times a_{j'k}$
        $X'_{j'\tau} = \max(X'_{j'\tau}, D_{j'\tau})$
      $End-do$
      $l' = l'+1$
    $End-do$
  $End-do$
  $\tau = \tau - 1$
$End-do$

Procedure 3:

Input: $X'_{jt} \forall j \in \pi(l)$
Output: $X'_{jt} \forall j \in \pi(l'), l' \leq l-1$
$l' = l-1$
$While\ (l' \geq 0)\ do$
  $Forall\ \left(j' \in \pi(l')\right) do$
    $X'_{j't} = \max\left(D_{j't} - I'_{j'(t-1)},\ \min_{k \in \mu(j')} \frac{X'_{kt}}{a_{kj'}}\right)$
  $End-do$
  $l' = l'-1$
$End-do$

Procedure 4:

Input: Q_{jt} , X'_{jt} $\forall j$
Output: $X'_{jt''}$ $\forall j$, $t < t'' \leq t'$
$Forall\ (i\ in\ 1..m, j \in \varphi(i))\ do$
    $Let, (RQ)_j = \max\{0, Q_{jt} - X'_{jt}\}$
    $While\ (t' > t + 1) do$
        $Forall\ \left(i' \in \rho(j)\right)\ do$
            $Unused_{Cap_{(i',t')}} = \max\{0, (Available_{Cap_{(i',t')}} - Req_{Cap_{(i',t')}})\}$
            $Allowable_{j,t'} = \dfrac{Unused_{Cap_{(i',t')}} - s_j}{p_j}$
            $X'_{jt'} = \min\{Allowable_{j,t'}, (D_{jt'} + (RD)_{jt}), (RQ)_j\}$
            $(RQ)_j = (RQ)_j - X'_{jt'}$
            $If\ X'_{jt'} = Allowable_{j,t'}\ then\ (RD)_{jt} = D_{j,t'} - X'_{jt'}$
            $Update\ Y'_{jt'}\ and\ I'_{jt'}$
            $Forall\ \left(k\ in\ \mu(j)\right) D_{kt'} = \sum_{k' \in \Gamma(k)} X'_{k't'} \times a_{k'k}$
        $End - do$
        $t' = t' - 1$
    $End - do$
    $If\ (RQ)_j > 0\ then$
        $X'_{jt'} = X'_{jt'} + (RQ)_j$
        $Forall\ \left(k\ in\ \mu(j)\right) D_{kt'} = \sum_{k' \in \Gamma(k)} X'_{k't'} \times a_{k'k}$
    $End - if$
$End - do$

3.4.4 Illustrative Example for CA heuristic:

Let us consider an instance of 4 periods and there are two end items with demand D_{1t}=(20, 25, 30, 30) and D_{4t}=(25, 20, 30, 35) $\forall t = 1..4$ to satisfy and each of the end items has two components. The product breakdown structure (See Figure 3.3) and other parameters (Table 3.2) are given below:

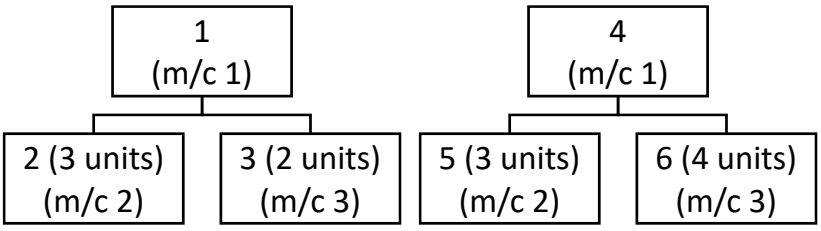


Figure 3.3: Product hierarchy structure for the example problem

Table 3.2: Parameters for the example problem of size ($T \times n \times m = 4 \times 6 \times 3$)

Job (j)	1	2	3	4	5	6
Processing time (p_j)	2	3	2	3	1	2
Setup time (s_j)	15	20	25	30	20	20
Holding cost(h_j)	0.5	0.6	0.2	0.3	0.6	0.2

Step 1: WW for end-items:

Each subproblem is an SIULSP. Let, X_{jt} and Z'_{jt} be the production quantity and allocated capacity for item j in period t respectively. The WW solution and the required capacity for item 1 and 4 at each period is given in Table 3.3.

Table 3.3: WW solution for end items 1 and 4

Period (t)	1	2	3	4
X_{1t}	45	0	60	0
Z'_{1t}	105	0	135	0
X_{4t}	110	0	0	0
Z'_{4t}	360	0	0	0

Step 2: Derive demands for components is given in Table 3.4.

Table 3.4: Derive demands for components

Period (t)	1	2	3	4
D_{2t}	135	0	180	0
D_{3t}	90	0	120	0
D_{5t}	330	0	0	0
D_{6t}	440	0	0	0

Step 3: WW for components

The WW solution for items 2,3,5 and 6 is given in Table 3.5.

Table 3.5: WW solution for items 2, 3, 5, and 6.

Period (t)	1	2	3	4
X_{2t}	135	0	180	0
X_{3t}	210	0	0	0
X_{5t}	330	0	0	0
X_{6t}	440	0	0	0

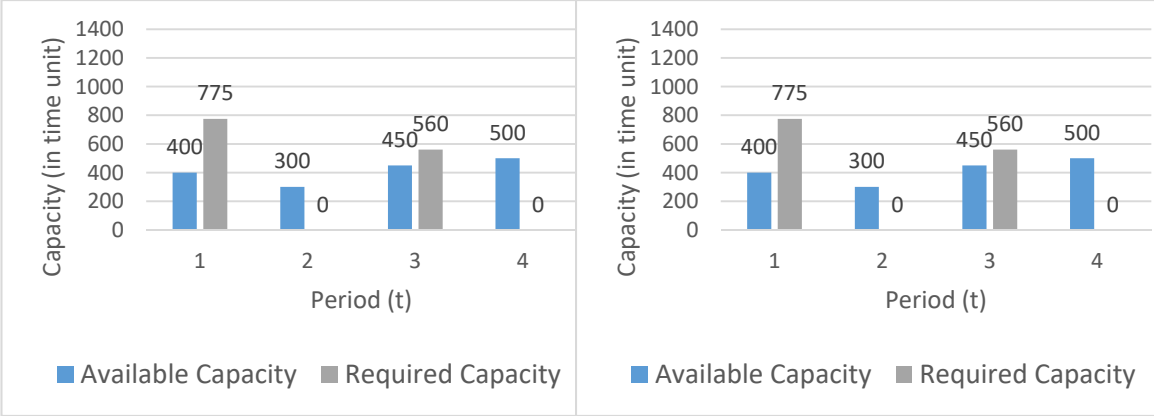
Step 4: Feasibility Procedure:

Step 4.1: Capacity allocation of the WW solution is shown in figure 3.4. Let, $t = 1, l = 0$. Item 1 and 4 are at level 0 and both of these items are processed by machine 1. The required capacity of machine 1 in period 1 exceeds the available capacity. That is why the production quantity of items 1 and 4 in period 1 is shifted to the later periods.

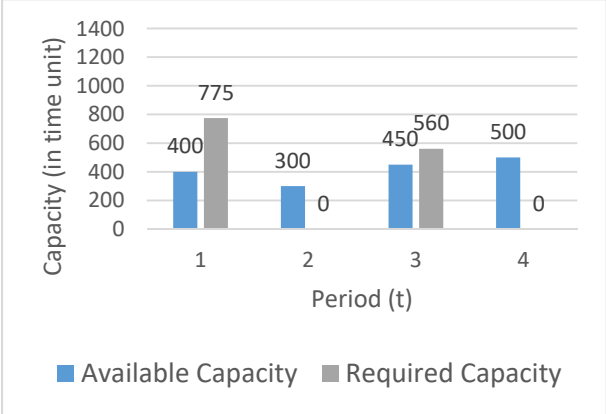
The available capacity of machine 1 in period 1 is allocated for items 1 and 4 as follows: $Z'_{11} = \left(105 \times \left(\frac{300}{465}\right)\right) = 67.74$ and $Z'_{41} = \left(360 \times \left(\frac{300}{465}\right)\right) = 232.258$. As a result, the production quantity for item 1 and 4 in period 1 is decreased as follows:

$X'_{11} = \lfloor (67.74 - 15)/2 \rfloor = 26$. and $X'_{41} = \lfloor (232.258 - 30)/3 \rfloor = 67$.

Derived demand and required capacity for items 2, 3, 5 and 6 in period 1 are 78, 52, 201 and 268 respectively.



(a) Machine 1 (b) Machine 2



(c) Machine 3

Figure 3.4: Capacity Allocation of WW solution

Step 4.2: Let $l = 1$. Items 2, 3, 5 and 6 are produced in the next level. The required capacity of machine 2 and 3 in period 1 is computed as follows:

item (j)	X'_{j1}	Z'_{j1}	Machine
2	$\max(78,135) = 135$	425	2
3	$\max(52,210) = 210$	445	3
5	$\max(201,,330) = 330$	350	2
6	$\max(268,440) = 440$	900	3

Thus, the required capacity of machine 2 and 3 in period 1 is 775 and 1345 time units respectively.

Step 4.3: The available capacity of machine 2 in period 1 is allocated for items 2 and 5 as follows:

$Z'_{21} = \left(425 \times \left(\frac{400}{775}\right)\right) = 219.35$ and $Z'_{51} = \left(350 \times \left(\frac{400}{775}\right)\right) = 180.65$. As a result, the production quantity for items 2 and 5 in period 1 decreases as follows: $X'_{21} = \lfloor (219.35 - 20)/3 \rfloor = 66$ and $X'_{51} = \lfloor (180.65 - 20)/1 \rfloor = 160$. Similarly, the production quantity for item 3 and 6 in period 1 is decreased to $X'_{31} = 70$ and $X'_{61} = 157$.

Required capacity of machine 2 = $66 \times 3 + 20 \times 1 + 160 \times 1 + 20 \times 1 = 398 < 400$.

Required capacity of machine 3 = $70 \times 2 + 25 \times 1 + 157 \times 2 + 20 \times 1 = 499 < 500$.

If required capacity exceeds the available capacity then start from step 4.2.

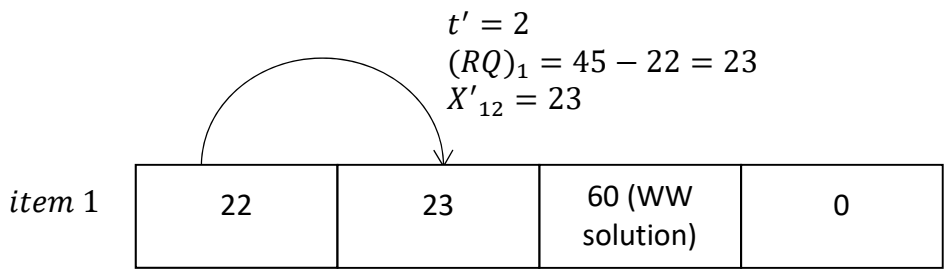
Step 4.4: Compute production quantity of the successor items: $X'_{11} = \max\left(20, \min\left(\frac{66}{3}, \frac{70}{2}\right)\right) = 22$ and $X'_{41} = \max\left(25, \min\left(\frac{160}{3}, \frac{157}{4}\right)\right) = 39$

Step 4.5: Update the production quantity of predecessors.

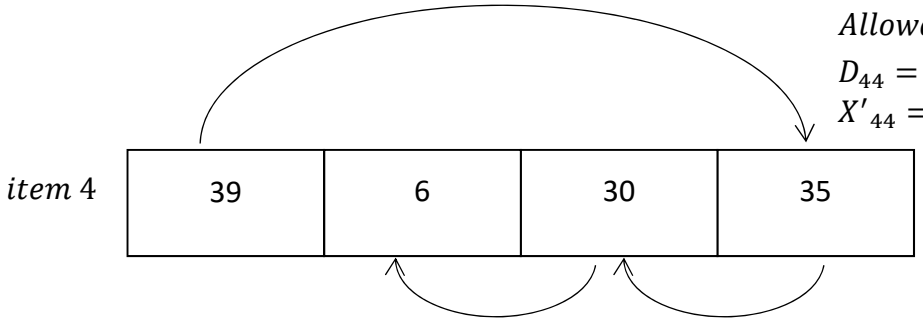
$X'_{21} = \max(66, 22 \times 3) = 66$, $X'_{31} = \max(70, 22 \times 2) = 70$,

$X'_{51} = \max(160, 39 \times 3) = 160$, $X'_{61} = \max(157, 39 \times 4) = 157$.

Step 4.6: Shift the production quantity for each item to the period (t') before the next production period obtained from WW schedule and then shift the excess production forward. For any item j, if $X_{jt} = 0 \forall t > 1$ then assign $t' = T$.



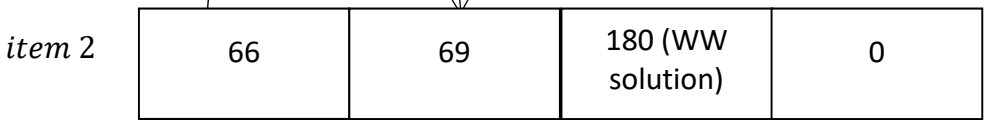
$t' = 4$
$(RQ)_4 = 110 - 39 = 71$
$Allowable_{4,4} = \frac{360-30}{3} = 110,$
$D_{44} = 35, (RD)_{2,4} = 0$
$X'_{44} = \min(110, 71, 35) = 35$



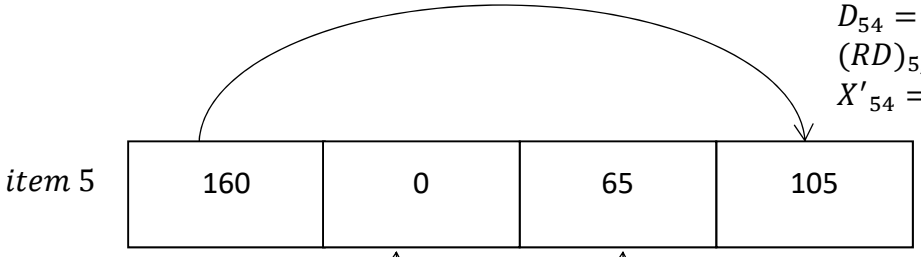
$(RQ)_4 = 36 - 30 = 6$
$X'_{42} = 6$

$t' = 3$
$(RQ)_4 = 71 - 35 = 36$
$Allowable_{4,3} = \frac{500-30}{3} = 156,$
$D_{43} = 30$
$(RD)_{4,3} = \min(0, D_{43} - Allowable_{4,3}) = 0$
$X'_{43} = \min(156, 36, 30) = 30$

$t' = 2$
$(RQ)_2 = 135 - 66 = 69$
$X'_{22} = 69$



$t' = 4$
$(RQ)_5 = 330 - 160 = 170$
$Allowable_{5,4} = \frac{500-20}{1} = 480,$
$D_{54} = 35 \times 3 = 105,$
$(RD)_{5,4} = \min(0, 105 - 480) = 0$
$X'_{54} = \min(480, 170, 105) = 105$



$(RQ)_4 = 65 - 65 = 0$
$X'_{52} = 0$

$t' = 3$
$(RQ)_5 = 170 - 105 = 65$
$Allowable_{5,3} = \frac{450-20}{1} = 430,$
$D_{53} = 30 \times 3 = 90, (RD)_{5,3} = 0$
$X'_{53} = \min(430, 65, 90) = 65$

Similarly the capacity allocation for item 3 and 6 is as follows:

$item\ 3$	70	20	120	0
$item\ 6$	157	36	107	140

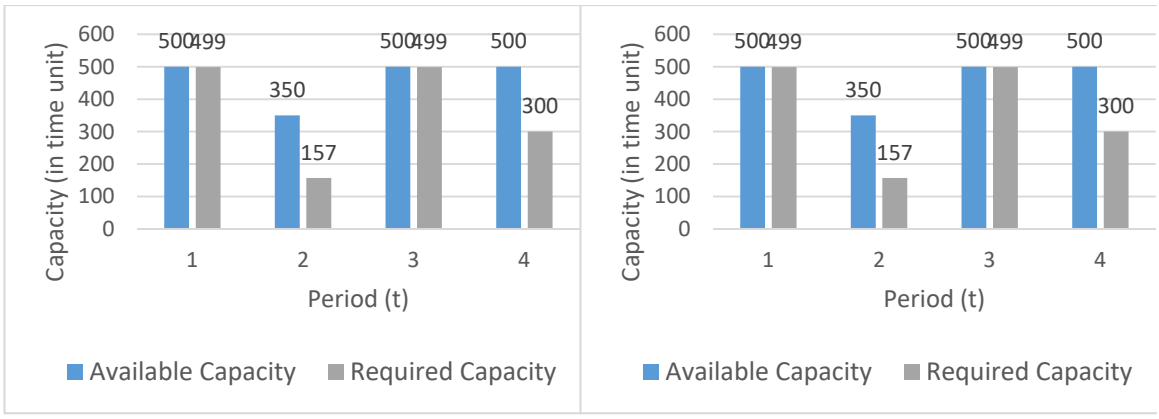
Step 4.7: $t = t + 1$ and repeat step 4.1 to 4.6 until $t = T$. The feasible solution after the capacity allocation is completed is shown in Table 3.6 and the capacity allocation of a feasible solution is shown in Figure 3.5.

Table 3.6: A feasible solution after the CA heuristic is completed

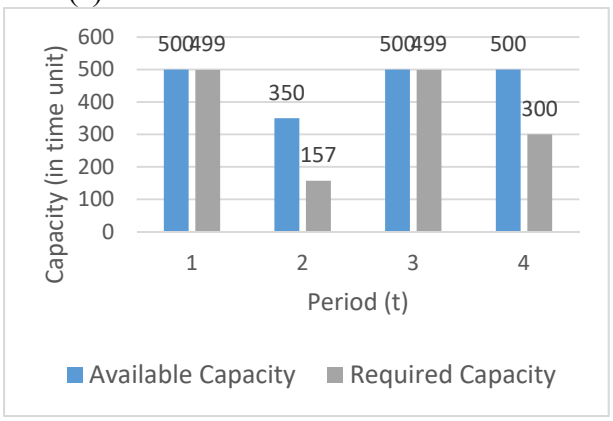
Job(j)	Period(t)			
	1	2	3	4
1	22	23	30	30
2	66	69	115	65
3	70	20	120	0
4	39	6	30	35
5	160	0	65	105
6	157	36	107	140

Step 5: Assign setup decision variables. For the example problem, $Y'_{jt} = 1$ forall j and t except $Y'_{3,4} = Y'_{5,2} = 0$.

Step 6: improvement procedure: Solve original problem as LP given the setup variables. The setup decisions (Y'_{jt}) provided by the CA heuristic is used as a parameter in the relaxed LP model for local search. As a result, the refined solution becomes optimum for a particular setup decision. Furthermore, if the setup decisions are correct, then the solution obtained using the local search method provide the optimum solution. The production schedule after local search is shown in Table 3.7.



(a) Machine 1 (b) Machine 2



Machine 3

Figure 3.5: Capacity Allocation of a feasible solution

Table 3.7: Production schedule after improvement procedure

Job(j)	Period(t)			
	1	2	3	4
1	20	25	30	30
2	60	75	90	90
3	40	59	111	0
4	46	0	29	35
5	138	0	87	105
6	184	36	116	140

3.5 Computational Study

The performance of the proposed DW decomposition and the CG procedure with the CA heuristic is tested using a large number of experimental test cases. We first consider a subset of the test instances introduced by Tempelmeier and Derstroff (1996), namely the 600 problem instances of class B with a noncyclic resource graph of general and an assembly product structure (as shown in Figure 3.6). All the test cases are comprised of ten items, three resources and four time periods. The 600 instances were generated combining:

1. One general and one assembly product structure
2. Three demand structures with varying coefficients of variance (CV = 0.1, 0.4, 0.7)
3. Five setup cost structures resulting in different profiles of average Time Between Orders (TBO = the average length of a production cycle) The numbers divided by slashes means TBO values for the higher, middle or the lower levels of the product hierarchy. Setup cost is computed using the following formula: $Setup\ cost = 0.5 \times holding\ cost \times average\ demand \times (TBO)^2$
4. Five capacity utilization profiles (90%, 70%, 50%, 90%/70%/50%, 40%/70%/90%). Available capacity per period is computed by dividing the mean demand by the target capacity utilization.
5. Two set up time profiles (see Table 3.8)
6. Two resource assignment profiles (see Table 3.9)

The mathematical model and the heuristic is coded using Fico's Mosel (Xpress) algebraic modeling language. All the test instances are run on a PC with an Intel Core i7 1.8 GHz processor, 8 GB of RAM and an L2 cache of 512KB.

Table 3.8: Setup time profiles for problem class B (Tempelmeier & Derstroff, 1996)

Setup time profile	Setup Time		
	5	10	15
1	7, 8, 9, 10	1, 2, 5, 6	3, 4
2	3, 4	1, 2, 5, 6	7, 8, 9, 10

Table 3.9: Resource assignment for problem class B (Tempelmeier & Derstroff, 1996)

Resource	General Product Structure	Assembly Product Structure
A	1..4	1
B	5..7	2..4
C	8..10	5..10

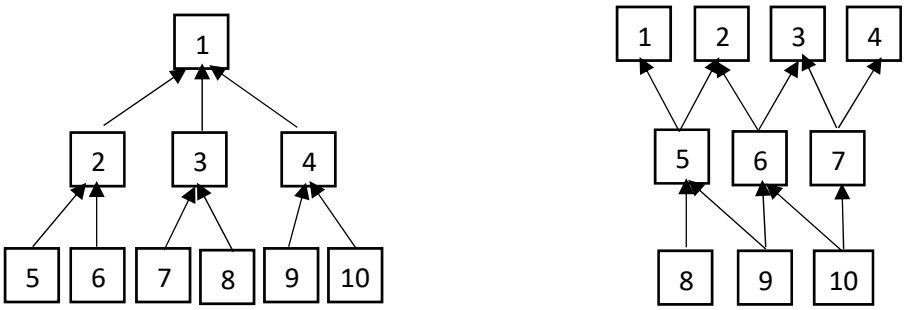
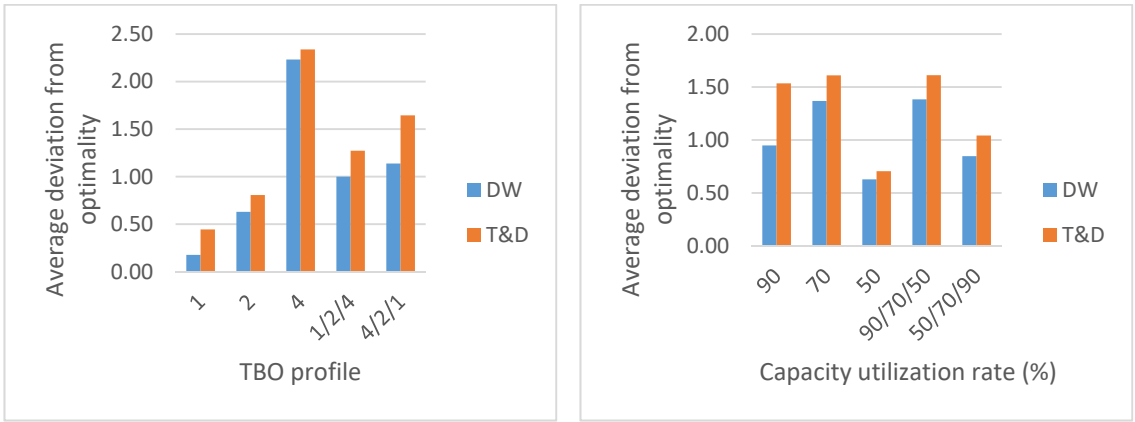


Figure 3.6: General and Assemble Product Structure for problem class B (Tempelmeier & Derstroff, 1996)

A comparison between the solution quality of the proposed approach, which uses the DW decomposition and CG combined with a CA heuristic and that of the Tempelmeier and Derstroff (1996) approach is shown in Table 3.10. As seen in Table 3.10, the average deviations from optimality by using the proposed heuristic method are much smaller than those reported in Tempelmeier and Derstroff (1996). Overall average optimality gap improves by 20% as compared to Tempelmeier and Derstroff (1996). Figure 3.7 shows the

average deviations from optimality (a) per TBO profile and (b) per capacity profile. Table 3.10 and Figure 3.7 confirm the competitiveness of the proposed heuristic method.

We apply the proposed heuristic in order to solve the MLCLSP with SCBE. Unlike the original data specification, we apply only the assembly product structure, three demand structures with varying coefficient of variance, five TBO profiles, five capacity utilization profiles, one setup time profiles (setup profile 1 from Table 3.8), one resource assignment profile, and three emission capacity profile (1500 t/MWh, 2000 t/MWh, and 2500 t/MWh). In Table 3.11, the percentage deviations of the heuristic solution values from the exact values are presented, broken down according to utilization profile, emission capacity profile, TBO profile and coefficient of variation of the demand series. The average computation time per problem instance is about 0.789 seconds for MILP and 0.928 seconds for the heuristic. The overall mean deviation from optimality for the 225 test instances are 1.75 and the mean variance is 0.63.



(a) (b)

Figure 3.7: Average deviations from optimality per (a) TBO profile (b) capacity profile

Table 3.10: Average deviation of the proposed heuristic solutions and comparison with the result given by Tempelmeier and Derstroff (1996)

TBO Profile	CV	Utilization rate(%)											
		90		70		50		90/70/50		50/70/90		mean	
		DW	T&D	DW	T&D	DW	T&D	DW	T&D	DW	T&D	DW	T&D
1	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.4	0.00	0.74	0.10	0.01	0.01	0	0.53	0.79	0.29	0.33	0.19	0.37
	0.7	0.12	2.24	0.22	0.24	0.12	0.25	0.52	1.24	0.75	0.83	0.35	0.96
	mean	0.04	0.99	0.11	0.08	0.04	0.08	0.35	0.68	0.35	0.39	0.18	0.44
2	0.1	0.05	0.13	1.09	1.1	0.02	0	0.44	0.48	0.59	0.73	0.44	0.49
	0.4	0.41	1.39	0.78	0.8	0.38	0.49	0.18	1.13	0.24	0.19	0.40	0.80
	0.7	1.45	1.35	0.70	0.78	0.4	0.51	1.42	1.51	1.29	1.49	1.05	1.13
	mean	0.64	0.96	0.86	0.89	0.27	0.33	0.68	1.04	0.71	0.81	0.63	0.81
4	0.1	0.23	0.28	4.52	4.88	0.18	0.08	2.45	2.55	0.85	0.91	1.65	1.74
	0.4	2.53	2.83	4.55	4.53	2.62	2.59	3.12	3.27	1.05	1.13	2.77	2.87
	0.7	3.42	3.54	1.80	1.99	0.44	0.57	3.18	3.24	2.54	2.68	2.28	2.40
	mean	2.06	2.22	3.62	3.8	1.08	1.08	2.92	3.02	1.48	1.57	2.23	2.34
1/2/4	0.1	0.16	0.18	0.25	0.86	0.53	0.84	0.48	1.18	0.14	0.14	0.31	0.64
	0.4	2.63	3.05	0.22	0.17	0.75	0.91	1.52	1.63	0.22	0.36	1.07	1.22
	0.7	2.23	4.4	1.05	0.58	1.19	1.28	1.18	1.26	2.45	2.53	1.62	2.01
	mean	1.67	2.54	0.51	0.54	0.82	1.01	1.06	1.26	0.94	1.01	1.00	1.27
4/2/1	0.1	0.03	0.58	1.62	2.31	0.02	0	0.85	0.82	0.00	0.39	0.50	0.82
	0.4	0.22	1.46	1.05	1.19	1.34	1.55	1.84	2.15	0.05	0.12	0.90	1.29
	0.7	0.75	0.85	2.58	4.71	1.42	1.53	3.05	3.21	2.24	3.77	2.01	2.81
	mean	0.33	0.96	1.75	2.74	0.93	1.03	1.91	2.06	0.76	1.43	1.14	1.64
Overall mean (600 problem instances)												1.04	1.30

*T&D = Tempelmeier and Derstroff (1996)

Table 3.11: Percentage Deviations from Optimality for MLCLSP with SCBE

		Utilization rate (%)															
		90			70			50			90/70/50			50/70/90			
Emission cap (t/MWh)		1500	2000	2500	1500	2000	2500	1500	2000	2500	1500	2000	2500	1500	2000	2500	mean
TBO Profile	CV																
1	0.1	0.80	0.14	0.19	0.52	0.56	0.19	1.34	0.33	0.19	1.34	0.69	0.19	0.73	0.14	0.19	0.50
	0.4	0.57	0.86	0.20	0.71	0.13	0.13	0.41	0.13	0.19	0.57	0.93	0.19	0.57	0.92	0.65	0.48
	0.7	0.34	0.45	0.64	0.95	0.46	0.53	1.19	0.48	0.65	0.99	0.45	0.19	0.59	0.45	0.19	0.57
	mean	0.57	0.48	0.34	0.73	0.38	0.28	0.98	0.32	0.34	0.97	0.69	0.19	0.63	0.50	0.34	0.52
	variance	0.49	1.00	0.04	0.06	1.05	0.40	1.06	1.47	0.53	0.13	0.68	0.10	0.11	0.40	0.07	0.08
2	0.1	1.80	0.52	0.76	1.95	2.37	1.78	1.90	2.05	1.71	1.90	0.52	1.26	1.44	0.52	0.69	1.41
	0.4	0.79	1.34	1.08	1.75	0.50	1.66	0.79	0.57	0.97	1.56	1.26	1.15	0.79	1.26	1.10	1.10
	0.7	2.13	2.51	1.09	2.25	2.17	2.81	2.85	2.97	2.43	2.28	2.17	1.75	1.13	1.78	1.20	2.10
	mean	1.57	1.46	0.98	1.98	1.68	2.08	1.85	1.86	1.70	1.91	1.32	1.39	1.12	1.18	1.00	1.54
	variance	0.49	1.00	0.04	0.06	1.05	0.40	1.06	1.47	0.53	0.13	0.68	0.10	0.11	0.40	0.07	0.51
4	0.1	1.00	0.79	1.81	2.75	4.04	2.52	1.59	3.62	1.89	3.13	3.32	1.04	1.94	1.71	4.56	2.38
	0.4	1.39	3.19	1.45	2.87	3.96	1.69	2.36	2.52	3.39	2.64	3.86	1.63	1.63	3.33	0.95	2.46
	0.7	1.16	2.54	2.04	3.39	4.23	2.30	3.80	0.98	4.49	3.37	4.28	1.57	2.93	4.08	4.93	3.07
	mean	1.18	2.17	1.77	3.00	4.08	2.17	2.58	2.37	3.26	3.04	3.82	1.41	2.17	3.04	3.48	2.64
	variance	0.04	1.54	0.09	0.12	0.02	0.18	1.26	1.76	1.70	0.14	0.23	0.11	0.46	1.47	4.83	0.93
1/2/4	0.1	1.02	0.99	3.14	1.37	9.79	3.64	2.30	1.10	3.45	4.87	1.87	2.26	3.63	1.48	2.95	2.92
	0.4	0.09	1.32	5.24	4.91	8.67	2.26	1.98	0.57	4.04	3.76	1.77	2.12	2.85	3.19	2.12	2.99
	0.7	0.66	3.72	5.28	1.50	9.50	3.46	5.50	1.23	5.07	1.54	4.71	2.66	4.42	1.17	0.99	3.43
	mean	0.59	2.01	4.55	2.59	9.32	3.12	3.26	0.97	4.19	3.39	2.79	2.34	3.63	1.94	2.02	3.11
	variance	0.22	2.22	1.50	4.03	0.34	0.56	3.79	0.12	0.67	2.87	2.79	0.08	0.62	1.18	0.97	1.46
4/2/1	0.1	0.87	0.84	1.32	0.92	1.22	0.50	1.03	1.65	2.37	1.71	0.37	0.50	1.01	0.37	0.50	1.01
	0.4	0.72	1.15	0.51	1.14	0.88	0.50	1.20	0.37	1.19	0.71	1.12	0.50	0.71	1.20	0.93	0.86
	0.7	1.44	0.43	0.50	1.49	1.08	1.36	1.81	1.13	1.46	1.45	1.08	0.49	0.64	0.68	0.50	1.04
	mean	1.01	0.81	0.77	1.18	1.06	0.79	1.35	1.05	1.67	1.29	0.86	0.50	0.78	0.75	0.64	0.97
	variance	0.14	0.13	0.22	0.08	0.03	0.25	0.17	0.41	0.38	0.27	0.18	0.00	0.04	0.18	0.06	0.17
Overall mean (225 test instances) =1.75 Overall mean variance = 0.63																	

To further investigate the proposed heuristic, we generate 96 test instances with increased size of assembly product structure. The new test instances are divided into four sets with the dimensions given in Table 3.12. These sets are combined with two levels of

capacity utilization rate for both production and emission limits (90% and 70%). For each combination, six instances were generated using two TBO profiles (1 and 2) and three coefficients of variance (0.1, 0.4, 0.7), resulting in a total of 24 instances for each set. The computational results are shown in Table 13, where each row contains aggregate results for the 6 instances in each combination described above. For problem sets A and B, the average % of gap column in Table 3.13 indicates the difference of the objective values resulting from the proposed heuristic method relative to the optimal solution. For problem sets C and D, the average percentage of gap is computed from the difference of the heuristic solution and the lower bound resulting from relaxing constraints (31) and (32). A lower percentage shows better performance for the solution methods.

Table 3.12: Dimensions of the new test problems

Problem Set	No. of Products	No. of Resources	No. of Periods	No. of Instances
A	15	6	4	24
B	10	3	10	24
C	15	6	10	24
D	10	3	20	24

The dimension of the test problems moderately increased but in many cases XPRESS solver is not able to compute the optimum solution within a time limit of one hour on a PC with an Intel Core i7 1.8 GHz processor, 8 GB of RAM and L2 cache of 512KB. For problem set A and B, the average percentage of gap is 0.845% and 1.09% respectively. For problem set C and D, Xpress solver could not solve a single instance. The average percentage of gap for problem set C and D is 5.88% and 4.58% respectively. The average percentage of gap is higher for problem set C and D because the lower bound of the model MLCLSP_SCBE is compared with the heuristic solution. The proposed

framework solves all the instances taken into account in less computational time and with a very small percentage of gap when compared to the MILP.

Table 3.13: Extended computational results

Problem set	Utilization rate (%)		# of Instances solved		Computational Time (Seconds)		Average % of gap
	Production capacity	Emission capacity	MILP	DW heuristic	MILP	DW heuristic	
A	90	90	3	6	1.19	0.86	1.98
		70	3	6	1.14	0.84	0.18
	70	90	6	6	2.94	0.75	0.61
		70	6	6	3.01	0.78	0.61
Overall mean for Problem set A (24 instances)							0.84
B	90	90	5	6	1.21	1.25	1.42
		70	6	6	13.08	0.98	0.68
	70	90	5	6	1.51	1.17	1.37
		70	6	6	53.28	0.93	0.88
Overall mean for Problem set B (24 instances)							1.09
C	90	90	0	6	-	2.04	5.72
		70	0	6	-	1.96	5.92
	70	90	0	6	-	1.96	5.87
		70	0	6	-	1.73	5.99
Overall mean for Problem set C (24 instances)							5.88
D	90	90	0	6	-	1.19	5.46
		70	0	6	-	1.44	5.19
	70	90	0	6	-	1.59	3.45
		70	0	6	-	1.54	4.20
Overall mean for Problem set D (24 instances)							4.58

3.6 Conclusion:

This chapter proposes an MILP model for the extension of the classical MLCLSP by incorporating setup carryover, backlogging, and emission control (MLCLSP_SCBE). An item DW decomposition technique is developed to decompose both the classical MLCLSP and MLCLSP_SCBE into a number of uncapacitated dynamic single-item lot-sizing problems, which are solved by combining dynamic programming and a multi-step iterative capacity allocation heuristic approach. An ILP model is developed to determine

the setup carryover variable to optimality for a given production schedule. An LP based post-improvement procedure is implemented to refine the solution. The capacity constraints are being taken into consideration implicitly through the dual multipliers, which are updated using a column generation procedure. The performance of the heuristic for classical MLCLSP is tested by comparing the average percentage of deviation from optimality with that of Tempelmeier and Derstroff (1996). Overall, the average optimality gap is improved by 20% as compared to Tempelmeier and Derstroff (1996). The quality of the heuristic for MLCLSP_SCBE is tested based on 225 small instances taken from the literature. Four new data sets containing a total of 96 problem instances with increased size is generated. Computational results show that the proposed optimization framework provides competitive solutions within a reasonable time frame.

Acknowledgement:

The research of M. F. Baki and A. Azab is partially supported by the Natural Sciences and Engineering Research Council's (NSERC) Discovery Grants. M. F. Baki's research is also partially funded by the Research and Teaching Innovation Fund (RTIF), Odette School of Business, University of Windsor. This research has also been funded by Dr. A. Azab's internal faculty funds.

REFERENCES

Absi, N., Dauzère-Pérès, S., Kedad-Sidhoum, S., Penz, B., & Rapine, C. (2013). Lot sizing with carbon emission constraints. *European Journal of Operational Research, 227*(1), 55-61. doi: http://dx.doi.org/10.1016/j.ejor.2012.11.044

Absi, N., Dauzère-Pérès, S., Kedad-Sidhoum, S., Penz, B., & Rapine, C. (2016). The single-item green lot-sizing problem with fixed carbon emissions. *European Journal of Operational Research, 248*(3), 849-855. doi: http://dx.doi.org/10.1016/j.ejor.2015.07.052

Absi, N., Kedad-Sidhoum, S., & Dauzère-Pérès, S. (2011). Uncapacitated lot-sizing problem with production time windows, early productions, backlogs and lost sales. *International Journal of Production Research, 49*(9), 2551-2566. doi: 10.1080/00207543.2010.532920

Aggarwal, A., & Park, J. K. (1993). Improved algorithms for economic lot size problems. *Oper. Res., 41*(3), 549-571. doi: 10.1287/opre.41.3.549

Almeder, C., Klabjan, D., Traxler, R., & Almada-Lobo, B. (2015). Lead time considerations for the multi-level capacitated lot-sizing problem. *European Journal of Operational Research, 241*(3), 727-738. doi: https://doi.org/10.1016/j.ejor.2014.09.030

Araujo, S. A. d., Reyck, B. D., Degraeve, Z., Fragkos, I., & Jans, R. (2015). Period Decompositions for the Capacitated Lot Sizing Problem with Setup Times. *INFORMS Journal on Computing, 27*(3), 431-448. doi: 10.1287/ijoc.2014.0636

Benjaafar, S., Li, Y., & Daskin, M. (2013). Carbon Footprint and the Management of Supply Chains: Insights From Simple Models. *IEEE Transactions on Automation Science and Engineering, 10*(1), 99-116. doi: 10.1109/TASE.2012.2203304

BERRETTA, R., FRANÇA, P. M., & ARMENTANO, V. A. (2005). METAHEURISTIC APPROACHES FOR THE MULTILEVEL RESOURCE-CONSTRAINED LOT-SIZING PROBLEM WITH SETUP AND LEAD TIMES. *Asia-Pacific Journal of Operational Research, 22*(02), 261-286. doi: 10.1142/s0217595905000510

Berretta, R., & Rodrigues, L. F. (2004). A memetic algorithm for a multistage capacitated lot-sizing problem. *International Journal of Production Economics, 87*(1), 67-81. doi: https://doi.org/10.1016/S0925-5273(03)00093-8

Billington, P. J., McClain, J. O., & Thomas, L. J. (1983). Mathematical Programming Approaches to Capacity-Constrained MRP Systems: Review, Formulation and Problem Reduction. *Management Science, 29*(10), 1126-1141. doi: 10.1287/mnsc.29.10.1126

Billington, P. J., McClain, J. O., & Thomas, L. J. (1986). Heuristics for Multilevel Lot-Sizing with a Bottleneck. *Manage. Sci., 32*(8), 989-1006. doi: 10.1287/mnsc.32.8.989

Boonmee, A., & Sethanan, K. (2016). A GLNPSO for multi-level capacitated lot-sizing and scheduling problem in the poultry industry. *European Journal of Operational Research, 250*(2), 652-665. doi: https://doi.org/10.1016/j.ejor.2015.09.020

Briskorn, D. (2006). A note on capacitated lot sizing with setup carry over. *IIE Transactions, 38*(11), 1045-1047. doi: 10.1080/07408170500245562

Buschkühl, L., Sahling, F., Helber, S., & Tempelmeier, H. (2010). Dynamic capacitated lot-sizing problems: a classification and review of solution approaches. *OR Spectrum, 32*(2), 231-261. doi: 10.1007/s00291-008-0150-7

Caserta, M., & Voß, S. (2012). A Math-Heuristic Dantzig-Wolfe Algorithm for the Capacitated Lot Sizing Problem. In Y. Hamadi & M. Schoenauer (Eds.), *Learning and Intelligent Optimization: 6th International Conference, LION 6, Paris, France, January 16-20, 2012, Revised Selected Papers* (pp. 31-41). Berlin, Heidelberg: Springer Berlin Heidelberg.

Chen, H. (2015). Fix-and-optimize and variable neighborhood search approaches for multi-level capacitated lot sizing problems. *Omega, 56*, 25-36. doi: http://dx.doi.org/10.1016/j.omega.2015.03.002

Chen, X., Benjaafar, S., & Elomri, A. (2013). The carbon-constrained EOQ. *Operations Research Letters, 41*(2), 172-179. doi: http://dx.doi.org/10.1016/j.orl.2012.12.003

Chowdhury, N. T., Baki, M. F., & Azab, A. (2018). Dynamic Economic Lot-Sizing Problem: A new O(T) Algorithm for the Wagner-Whitin Model. *Computers & Industrial Engineering, 117*, 6-18. doi: https://doi.org/10.1016/j.cie.2018.01.010

Degraeve, Z., & Jans, R. (2007). A New Dantzig-Wolfe Reformulation and Branch-and-Price Algorithm for the Capacitated Lot-Sizing Problem with Setup Times. *Operations Research, 55*(5), 909-920. doi: 10.1287/opre.1070.0404

Duarte, A. J. S. T., & de Carvalho, J. M. V. V. (2015). A Column Generation Approach to the Discrete Lot Sizing and Scheduling Problem on Parallel Machines. In J. P. Almeida, J. F. Oliveira & A. A. Pinto (Eds.), *Operational Research: IO 2013 - XVI Congress of APDIO, Bragança, Portugal, June 3-5, 2013* (pp. 157-170). Cham: Springer International Publishing.

Fiorotto, D. J., de Araujo, S. A., & Jans, R. (2015). Hybrid methods for lot sizing on parallel machines. *Computers & Operations Research, 63*(Supplement C), 136-148. doi: https://doi.org/10.1016/j.cor.2015.04.015

Gopalakrishnan, M. (2000). A modified framework for modelling set-up carryover in the capacitated lotsizing problem. *International Journal of Production Research, 38*(14), 3421-3424. doi: 10.1080/002075400418324

Gopalakrishnan, M., Miller, D. M., & Schmidt, C. P. (1995). A framework for modelling setup carryover in the capacitated lot sizing problem. *International Journal of Production Research, 33*(7), 1973-1988. doi: 10.1080/00207549508904793

Gören, H. G., & Tunalı, S. (2015). Solving the capacitated lot sizing problem with setup carryover using a new sequential hybrid approach. *Applied Intelligence, 42*(4), 805-816. doi: 10.1007/s10489-014-0626-x

Haase, K. (1998). Capacitated Lot-Sizing with Linked Production Quantities of Adjacent Periods. In A. Drexl & A. Kimms (Eds.), *Beyond Manufacturing Resource Planning (MRP II): Advanced Models and Methods for Production Planning* (pp. 127-146). Berlin, Heidelberg: Springer Berlin Heidelberg.

Helber, S., & Sahling, F. (2010). A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *International Journal of Production Economics, 123*(2), 247-256. doi: http://dx.doi.org/10.1016/j.ijpe.2009.08.022

Jans, R., & Degraeve, Z. (2004). Improved lower bounds for the capacitated lot sizing problem with setup times. *Operations Research Letters, 32*(2), 185-195. doi: https://doi.org/10.1016/j.orl.2003.06.001

Jans, R., & Degraeve, Z. (2007). Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research, 177*(3), 1855-1875. doi: https://doi.org/10.1016/j.ejor.2005.12.008

Karimi, B., Ghomi, S. M. T. F., & Wilson, J. M. (2006). A tabu search heuristic for solving the CLSP with backlogging and set-up carry-over. *Journal of the Operational Research Society, 57*(2), 140-147. doi: 10.1057/palgrave.jors.2601968

Kimms, A. (1997). *Multi-level lot sizing and scheduling: Methods for capacitated, dynamic, and deterministic models*. Heidelberg: Physica-Verlag: Production and Logistics.

Maes, J., McClain, J. O., & Van Wassenhove, L. N. (1991). Multilevel capacitated lotsizing complexity and LP-based heuristics. *European Journal of Operational Research, 53*(2), 131-148. doi: https://doi.org/10.1016/0377-2217(91)90130-N

Manne, A. S. (1958). Programming of Economic Lot Sizes. *Management Science, 4*(2), 115-135. doi: 10.1287/mnsc.4.2.115

Masmoudi, O., Yalaoui, A., Ouazene, Y., & Chehade, H. (2015, 21-23 Oct. 2015). *A Multi-Level Capacitated Lot-Sizing Problem with energy consideration.* Paper presented at the 2015 International Conference on Industrial Engineering and Systems Management (IESM).

Masmoudi, O., Yalaoui, A., Ouazene, Y., & Chehade, H. (2017). Solving a capacitated flow-shop problem with minimizing total energy costs. *The International Journal of Advanced Manufacturing Technology, 90*(9), 2655-2667. doi: 10.1007/s00170-016-9557-5

Oztürk, C., & Ornek, A. M. (2010). Capacitated lot sizing with linked lots for general product structures in job shops. *Computers & Industrial Engineering, 58*(1), 151-164. doi: https://doi.org/10.1016/j.cie.2009.10.002

Pimentel, C. M. O., Alvelos, F. P. e., & Valério de Carvalho, J. M. (2010). Comparing Dantzig–Wolfe decompositions and branch-and-price algorithms for the multi-item capacitated lot sizing problem. *Optimization Methods and Software, 25*(2), 299-319. doi: 10.1080/10556780902992837

Pitakaso, R., Almeder, C., Doerner, K. F., & Hartl, R. F. (2006). Combining population-based and exact methods for multi-level capacitated lot-sizing problems. *International Journal of Production Research, 44*(22), 4755-4771. doi: 10.1080/00207540600620963

Ramsay, T. E. J. (1981). *Integer Programming Approaches to Capacitated Concave Cost Production Planning Problem.* (Ph.D Dissertation), Georgia Institute of Technology.

Retel Helmrich, M. J., Jans, R., van den Heuvel, W., & Wagelmans, A. P. M. (2015). The economic lot-sizing problem with an emission capacity constraint. *European Journal of Operational Research, 241*(1), 50-62. doi: https://doi.org/10.1016/j.ejor.2014.06.030

Sahling, F., Buschkühl, L., Tempelmeier, H., & Helber, S. (2009). Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a fix-and-optimize heuristic. *Computers & Operations Research, 36*(9), 2546-2553. doi: https://doi.org/10.1016/j.cor.2008.10.009

Seeanner, F., Almada-Lobo, B., & Meyr, H. (2013). Combining the principles of variable neighborhood decomposition search and the fix&optimize heuristic to solve multi-level lot-sizing and scheduling problems. *Computers & Operations Research, 40*(1), 303-317. doi: https://doi.org/10.1016/j.cor.2012.07.002

Sox, C. R., & Gao, Y. (1999). The capacitated lot sizing problem with setup carry-over. *IIE Transactions, 31*(2), 173-181. doi: 10.1023/a:1007520703382

Tempelmeier, H., & Buschkühl, L. (2009). A heuristic for the dynamic multi-level capacitated lotsizing problem with linked lotsizes for general product structures. *OR Spectrum, 31*(2), 385-404. doi: 10.1007/s00291-008-0130-y

Tempelmeier, H., & Derstroff, M. (1996). A Lagrangean-Based Heuristic for Dynamic Multilevel Multiitem Constrained Lotsizing with Setup Times. *Management Science, 42*(5), 738-757.

Toledo, C. F. M., de Oliveira, R. R. R., & Morelato França, P. (2013). A hybrid multi-population genetic algorithm applied to solve the multi-level capacitated lot sizing problem with backlogging. *Computers & Operations Research, 40*(4), 910-919. doi: https://doi.org/10.1016/j.cor.2012.11.002

Vanderbeck, F. (1998). Lot-Sizing with Start-Up Times. *Management Science, 44*(10), 1409-1425. doi: 10.1287/mnsc.44.10.1409

Vanderbeck, F. (2000). On Dantzig-Wolfe Decomposition in Integer Programming and ways to Perform Branching in a Branch-and-Price Algorithm. *Operations Research, 48*(1), 111-128. doi: 10.1287/opre.48.1.111.12453

Vanderbeck, F., & Savelsbergh, M. W. P. (2006). A generic view of Dantzig–Wolfe decomposition in mixed integer programming. *Operations Research Letters, 34*(3), 296-306. doi: https://doi.org/10.1016/j.orl.2005.05.009

Wagelmans, A., Hoesel, S. V., & Antoon, K. (1992). Economic Lot Sizing: An O(n log n) Algorithm That Runs in Linear Time in the Wagner-Whitin Case. *Operations Research, 40*, S145-S156. doi: 10.2307/3840844

Wagner, H. M., & Whitin, T. M. (1958). Dynamic Version of the Economic Lot Size Model. *Management Science, 5*(1), 89-96. doi: 10.1287/mnsc.5.1.89

Wu, T., Akartunalı, K., Song, J., & Shi, L. (2013). Mixed integer programming in production planning with backlogging and setup carryover: modeling and algorithms. *Discrete Event Dynamic Systems, 23*(2), 211-239. doi: 10.1007/s10626-012-0141-3

Wu, T., Shi, L., Geunes, J., & Akartunalı, K. (2011). An optimization framework for solving capacitated multi-level lot-sizing problems with backlogging. *European Journal of Operational Research, 214*(2), 428-441. doi: https://doi.org/10.1016/j.ejor.2011.04.029

Zhao, Q., Xie, C., & Xiao, Y. (2012). A variable neighborhood decomposition search algorithm for multilevel capacitated lot-sizing problems. *Electronic Notes in Discrete Mathematics, 39*(Supplement C), 129-135. doi: https://doi.org/10.1016/j.endm.2012.10.018

CHAPTER 4

LOT-SIZING PROBLEM TO MAXIMIZE SETUP COST SAVINGS: AN APPLICATION OF THE MAXIMUM WEIGHTED INDEPENDENT SET PROBLEM

The Setup Carryover Assignment Problem (SCAP), which consists of determining the setup carryover plan of multiple items for a given lot-size over a finite planning horizon with the objective of maximizing setup costs savings is presented in this Chapter. The SCAP is modelled as a problem of finding Maximum Weighted Independent Set (MWIS) in a chain of cliques, which is formulated as an Integer Linear Programming (ILP) model. It is shown that Linear Program (LP) relaxation of a straightforward formulation of MWIS gives fractional solution. The SCAP is then formulated using a clique constraint and it is proved that the incidence matrix of the SCAP has totally unimodular structure and the LP relaxation of the proposed SCAP formulation always provides integer optimum solution. Moreover, an alternative proof that the relaxed ILP guarantees integer solution is presented in this chapter. Thus, the SCAP and the special case of the MWIS in a chain of cliques are solvable in polynomial time.

4.1 Introduction

Lot-sizing is the process of determining a tentative plan for how much production will occur in the next time periods during an interval of time called planning horizon. In each period that an item is produced a setup is required. A setup may cause setup costs as well as setup time. If an item produced at the end of period t is continued at the beginning

of the next period $(t+1)$, it is cost-effective to maintain the setup of that item into period $(t+1)$ to save the setup cost. This is referred to as setup carryover (Briskorn, 2006). Setup carryover allows the machine setup to be maintained between two adjacent periods. For a given production schedule, the Setup Carryover Assignment Problem (SCAP) is to determine the set of items to carryover from one period to the next such that the total savings of setup cost is maximized.

To illustrate the problem, we use an example. Let us consider an SCAP where multiple items $(j_1, j_2, \dots, j_6)$ are being processed on the same resource over a planning horizon of length $T = 6$. Let us model the SCAP in the form of some connected undirected cliques $G_t \; \forall t = 1..5$ as shown in Figure 1. Note that a clique is a subset of nodes in which every two nodes are connected by an edge. In Figure 1, each clique represents a period. Items produced in period t and $(t+1)$ are placed as nodes in clique $G_t \; \forall t \leq T-1$. Therefore, each node in G_t represents an item that can be carried over from period t to $(t+1) \forall t \leq T-1$. To refer to the condition that only one item can be produced at the end of one period and at the beginning of the next period, we connect all nodes in a clique and formulate a problem that allows us to choose at most one node from two nodes connected by an edge, so at most one node from a clique. Choosing a node from $G_t \; \forall t \leq T-1$ represents producing the corresponding item at the end of period t and at the beginning of period $(t+1)$. Furthermore, the edges between G_t and G_{t+1} refer to the condition that if item j is produced at the end of period t, then it is continued at the beginning of period $(t+1)$. This implies that j cannot be produced at the end of period $(t+1)$ unless j is the only eligible item to carryover. The savings in setup corresponds to the weight of the problem.

The problem of maximizing savings of setup cost is equivalent to the problem of choosing a maximum weighted set of nodes such that no two nodes are connected by an edge. This problem is known as Maximum Weighted Independent Set (MWIS) problem. By definition, an independent set in a graph G is vertex set in which no two vertices are adjacent. If each vertex of G is assigned a positive weight, then we say that G is a weighted graph. The Maximum Weighted Independent Set (MWIS) problem consists of finding in a weighted graph an independent set of maximum total weight.

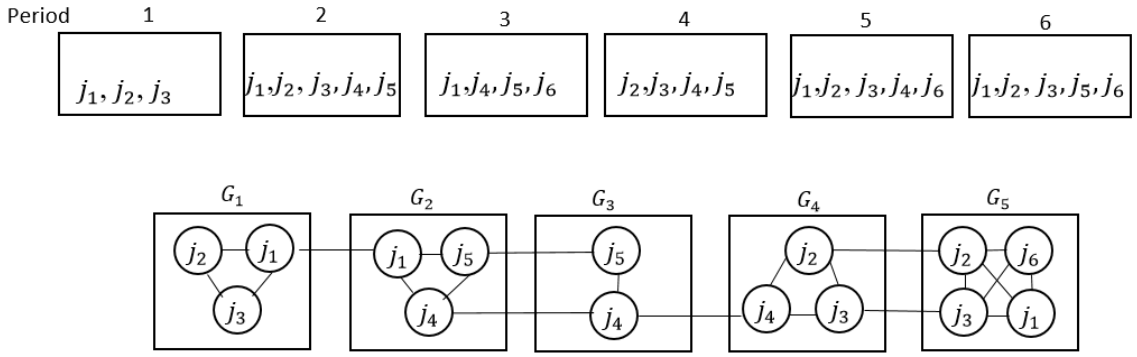


Figure 4.1: A simple undirected graph used to model the SCAP as the MWIS problem

In this chapter, we formally describe a special case of MWIS problem in a chain of cliques, formulate it as an Integer Linear Programming (ILP) model, and present its natural Linear Program (LP) relaxation. We show that LP relaxation of a straightforward formulation of MWIS and solution of SCAP using that formulation gives fractional solution. We model the SCAP as a chain of cliques and show that the SCAP is equivalent to the problem of finding MWIS in chain of cliques. The SCAP is formulated as an ILP model for a given production schedule to maximize the savings in the setup cost. We also prove that the constraint matrix of the ILP has a totally unimodular structure and LP relaxation of the proposed ILP always provides an integer optimum solution. We also give an alternative proof of integer solution of the relaxed ILP. Thus, the SCAP and its

equivalent the special case of the MWIS in a chain of cliques are solvable in polynomial time.

The rest of the chapter is organized as follows: Section 4.2 reviews some relevant literature. Section 4.3 provides a mathematical formulation of the MWIS. Section 4.4 states the problem of SCAP and presents an ILP model addressing the problem. Section 4.5 shows the equivalency of the SCAP to the MWIS problem. Section 4.6 relaxes the proposed ILP model and presents two alternate proofs that the relaxed LP provides integer optimal solution. A numerical example is provided in Section 4.7 and the conclusion along with some future research direction is presented in Section 4.8.

4.2 Literature Review

The production changeovers between different items on the same machine incur setup time and setup cost. Setup time is the time required to prepare the necessary machines to perform a task while setup cost is the cost to setup a machine before the execution of a task (Allahverdi & Soroush, 2008). Setup tasks are expensive in terms of loss of production time, material and labor hours. Therefore, setup reduction is an important feature of the continuous improvement program of any manufacturing/service organization. Allahverdi (2015) provides an up to date survey of lot-sizing problems with setup times/costs and addresses different industry application where setup is a crucial part of production planning process. However, if an item is produced in two consecutive periods, it is possible to conserve the setup state of the machine between those periods, which is referred to as setup carryover (Briskorn, 2006). This may happen over multiple consecutive periods. Since incorporating setup carryover has a significant effect on both cost and lot sizes (Sox & Gao, 1999), it is crucial to determine the setup carryover variables correctly. Many researchers

(Haase, 1998; Sahling, Buschkühl, Tempelmeier, & Helber, 2009; Sox & Gao, 1999; Tempelmeier & Buschkühl, 2009) have considered the lot-sizing with setup carryover and propose various solution methodologies such as priority rule based scheduling procedure (Haase, 1998), Lagrangian decomposition heuristic (Sox & Gao, 1999; Tempelmeier & Buschkühl, 2009), Fix and optimize heuristic (Sahling et al., 2009) and so on to solve the problem. The heuristic solution sometimes generate infeasible solution in terms of setup carryover constraints. Sox and Gao (1999) provide a feasibility procedure and Tempelmeier & Buschkühl (2009) apply post-optimization in order to make sure that the setup carryover constraints are satisfied.

We show in this Chapter that SCAP can be formulated as the problem of finding an MWIS in a chain of cliques. MWIS is a combinatorial optimization problem that naturally arises in many applications. Several real-life problems can be formulated as MWIS including wireless network scheduling (I. C. Paschalidis, F. Huang, & W. Lai, 2015), graph coloring (Pal & Sarma, 2012), graph coding (Etzion & Ostergard, 1998), multi-object tracking (Brendel, Amer, & Todorovic, 2011), and molecular biology (Gardiner, Artymiuk, & Willett, 1997).

The MWIS problem has been extensively studied in the literature. Finding a maximum independent set of a graph is known to be NP-hard (Garey & Johnson, 1979) in general. However, it is known to be solvable in polynomial time for some cases including perfect and interval graphs (Grotschel, Lovász, & Schrijver, 1993), disk graphs (Matsui, 2000), claw-free graphs (Minty, 1980), fork-free graphs (Alekseev, 2004), trees (Chen, Kuo, & Sheu, 1988), circle graphs (Valiente, 2003), growth-bounded graphs (Gfeller & Vicari, 2007) and so on. Moreover, there has been an extensive work on approximating the MWIS (Kako, Ono, Hirata, & Halldórsson, 2009), and specialized algorithms have been

developed for exactly computing the MWIS (Xiao & Nagamochi, 2016) in any graph in general. Although exact approaches provide an optimal solution, they become computationally intractable for the graphs with several hundreds of vertices. Therefore, the application of heuristic approaches are very common when one deals with the MWIS problem on very large graphs. Early attempts to apply different metaheuristic methods to the MWIS problems were made in the beginning of 1990's. Back and Khuri (1994) use genetic algorithms to solve the MWIS problems. Many successful implementations of the evolutionary algorithms have appeared in the literature ever since (Borisovsky & Zavolovskaya, 2003; Hifi, 1997). Simulated Annealing (SA) is another popular metaheuristic approach, which has wide application in the combinatorial optimization problems. An example of SA for the MWIS is described in the textbook by Aarts and Korst (1989). Other well-known metaheuristic methods which have been successfully implemented to the MWIS include greedy randomized adaptive search procedures or GRASP (Feo, Resende, & Smith, 1994) and tabu search (Friden, Hertz, & de Werra, 1990).

4.3 Maximum Weighted Independent Set and its LP relaxation:

SCAP can be modelled as an MWIS problem in a chain of cliques. Given a production schedule that solves an MWIS problem with appropriate weights to decide the machine setup state of which items to preserve for the next period to maximize the savings in setup cost, is the starting point of our work in this Chapter.

Given a chain of K cliques $G = (\bigcup_{t=1}^{K} G_t, E_0)$, where $G_t = (V_t, E_t, W_t) \forall t = 1,2, \dots, K$ is the $t-th$ set of cliques, $V_t = \{1,2, \dots, n\}$ is the $t-th$ set of nodes, $E_t = \{(j,k) | j,k \in V_t and\ j \neq k\}$ is the $t-th$ set of edges, $W_t = \{c_{t1}, c_{t2}, \dots, c_{tn} | c_{tj}$ =weight of the $j-th$ node of the $t-th$ clique$\}$ is the $t-th$ set of weights, and E_0 is the set of

edges such that for any node $j \in G_t$ there can be at most one edge $(j,k)|k \in G_{t+1}\ \forall 1 \leq t \leq K-1$, at most one edge $(k,j)|k \in G_{t-1} \forall 2 \leq t \leq K$, no edge of the type $(j,k')|k' \in G_{t'}\ \forall t' \geq t+2$, and no edge of the type $(j,k')|k' \in G_{t'}\ \forall t' \leq t-2$, the problem addressed in this Chapter is to find an MWIS in G.

Figure 4.2 shows a weighted undirected graph $G = (\bigcup_{t=1}^{3} G_t, E_0)$ consisting of a chain of three cliques G_1, G_2, and G_3, where $G_1 = (\{1,2,3\}, \{(1,2),(2,3),(3,1)\}, \{0.5, 0.8, 0.6\})$, G_2 =($\{4,5,6\}$,$\{(4,5),(5,6),(6,4)\}$,$\{0.8,0.5,0.6\}$), G_3 =($\{7,8,9,10\}$,$\{(7,8),(7,9),(7,10),(8,9),(8,10),(9,10)\}$,$\{0.5,0.9,0.6,0.7\}$) and $E_0 = \{(1,5),(2,4),(3,6),(5,7),(6,9)\}$. We are interested in finding an MWIS x^* in G, which maximizes the sum of the total weights.

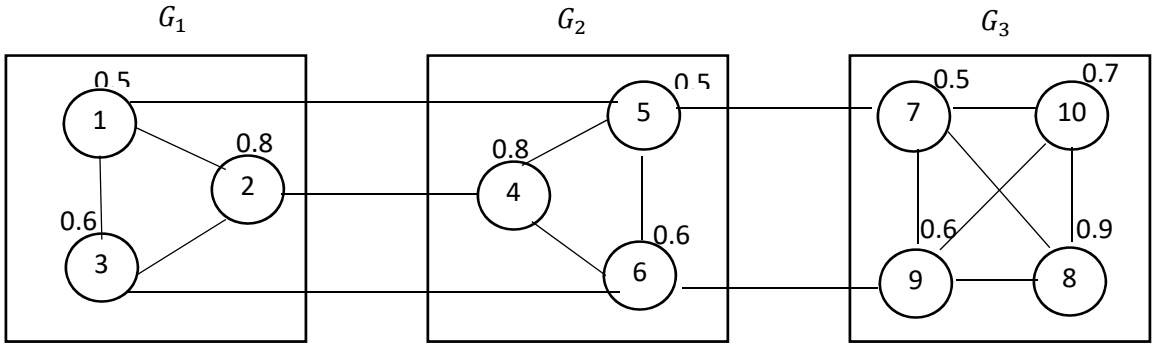


Figure 4.2: A simple chain of three cliques $G = (\bigcup_{t=1}^{3} G_t, E_0)$

Let us introduce the indices and the parameters for the problem as follows:

Indices:

t clique index ($t = 1,2,3,\dots,K$)

j,k node index ($j = 1,2,3,\dots,n; k = 1,2,3,\dots,n$)

Parameters:

w_j positive weight associated with node $j|j \in V_t\ \forall t$

Decision variable:

$$x_j = \begin{cases} 1 & \text{if node } j \text{ is in the independent set} \\ 0 & \text{otherwise} \end{cases}$$

Model: MWIS

$$Max \quad \sum_{j=1}^{n} w_j x_j \tag{1}$$

Subject to,

$$x_j + x_k \leq 1 \ \forall t, \ (j,k) \in E_t \tag{2}$$

$$x_j + x_k \leq 1 \ \ \forall t, (j,k) \in E_0 \tag{3}$$

$$x_j \ \in \{0,1\} \ \forall j \tag{4}$$

The objective function (1) is to maximize the total node weights. Constraints (2) are the edge constraints within a clique and constraints (3) are the edge constraints between two adjacent cliques. The edge constraints (2) and (3) prohibits two nodes of the same edge to be selected at the same time. Constraints (4) is the integrality constraint. The LP relaxation of MWIS is formed by relaxing constraints (4) as $0 \leq x_j \leq 1$. We refer to this LP as the relaxed MWIS. Below we show that the relaxed MWIS does not satisfy the totally unimodular property (i.e., every square non-singular submatrix of the incidence matrix has determinant 0, +1 or −1) and the Relaxed MWIS gives fractional solution.

The optimum solution to the problem illustrated in Figure 4.2 is x^* ={2, 6, 8} and the total weight is 2.3. If the relaxed MWIS is used, the resulting MWIS is x^* ={1,2,3,4,5,6,7,8,9,10} and the total weight becomes 3.25 with fractional $x_j = 0.5 \ \forall j = 1, \ldots, 10$, which is an infeasible solution.

Let A be the {0, 1} edge-vertex incidence matrix for the graph $G = (\bigcup_{t=1}^{K} G_t, E_0)$, defined as follows: A has $|(\bigcup_{t=1}^{K} E_t) \cup E_0|$ rows, one for each edge and $|\bigcup_{t=1}^{K} V_t|$ columns, one for each vertex. $A_{jk} = 1$ if vertex j is incident to vertex k otherwise it is 0. Figure 4.3 shows the incident matrix corresponding to clique G_1.

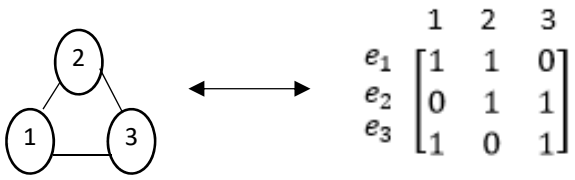


Figure 4.3: Clique G_1 and the correspondent incident matrix

The incidence matrix of model MWIS does not have totally unimodular structure. For example the determinant of $\begin{bmatrix}1 & 1 & 0\\0 & 1 & 1\\1 & 0 & 1\end{bmatrix}$ shown in Figure 4.3 is 2. Moreover, the relaxed MWIS provides fractional solution and the resulting MWIS becomes infeasible.

To avoid infeasibility, we model the SCAP using a clique constraint. In this Chapter, we shall show that SCAP is equivalent to MWIS in a chain of cliques, which is solvable in polynomial time.

4.4 ILP formulation for SCAP:

Let us consider a production schedule where multiple items ($j_1, j_2, \dots, j_k \forall k$) are to be produced on the same machine over a planning horizon of length T. An Integer Linear Programming (ILP) model can be formulated to find the set of items to carryover from one period to the next such that the total savings of setup cost is maximized. Suppose we are given $S(t) \forall t$, where $S(t)$ is the set of items produced in period $t \forall\ t = 1 \dots T$. Let $S'(t) = S(t) \cap S(t+1)\ \forall t = 1 \dots T-1$. Each element of $S'(t)$ represents an item that can be carried over from period t to $(t+1)$ to avoid the machine setup for that item in period $(t+1)$. Since only one item can be carried over to the next period, we have to pick exactly one element from $S'(t)$. We redefine the indices and introduce some new parameters for the problem as follows:

Indices:

t planning period $(t = 1,2,3, \dots, T)$

j item index $(j = 1,2,3, \dots, n)$

Parameters:

c_j Setup cost saving associated with element $j | j \in S'(t)\ \forall t$

$$q_{jt} = \begin{cases} 1 & \text{if item } j \in S'(t) \\ 0 & \text{otherwise} \end{cases}$$

$$r_{jt} = \begin{cases} 1 & \text{if } q_{jt} = q_{j(t+1)} = 1 \text{ and } if\ |S'(t+1)| > 1 \\ 0 & \text{otherwise} \end{cases}$$

Decision variable:

$$z_{jt} = \begin{cases} 1 & \text{if item } j \in S'(t) \text{ is carriedover from period } t \text{ to } (t+1) \\ 0 & \text{otherwise} \end{cases}$$

Model: SCAP

$$Max \quad \sum_{j=1}^{n} \sum_{t=1}^{T-1} c_j z_{jt} \tag{6}$$

Subject to,

$$z_{jt} \leq q_{jt}\ \forall j, t \leq T-1 \tag{7}$$

$$\sum_{j \in S'(t)} z_{jt} \leq 1 \quad \forall t \leq T-1 \tag{8}$$

$$z_{jt} + z_{j(t+1)} \leq 1 \quad \forall j, t \leq T-1 | r_{jt} = 1 \tag{9}$$

$$z_{jt} \in \{0,1\}\ \forall j, t \tag{10}$$

The objective function (6) is to maximize the setup cost savings. Constraints (7) ensure that an item, which is produced in two consecutive periods, should be carried over to the next period. Constraints (8) are the clique constraints which state that at most one item can be carried over to the next period. But for some t, if $q_{jt} = 0 \forall j \in S(t)$, $\sum_{j \in S(t)} z_{jt} = 0$. Constraints (9) prevents same item to be selected to carryover in two consecutive periods if $r_{jt} = 1$, which implies the condition that if item j is carried over

from period t to $(t+1)$ then j cannot be carried over from $(t+1)$ to $(t+2)$. Finally the type of variables are defined in constraints (10).

4.5 The equivalency of SCAP to the problem of finding the MWIS in a chain of cliques:

This section shows that for a given production schedule, the solution of SCAP will yield the MWIS and vice versa.

Theorem 1: The SCAP for inventory lot-sizing over T periods is equivalent to finding the MWIS in a chain of $(T-1)$ cliques.

Proof: Consider an instance of the SCAP, where $S'(t) = \{j_{t,k} \,|k = 1,2,..n_t\} \forall t \leq T-1$ represents a set of items for each of which setup can be carried over from period t to $(t+1)$ to avoid the machine setup for that item in period $(t+1)$. The condition of the SCAP is that only one item can be produced at the end of period t and if item j is produced at the end of period t, the same item cannot be produced at the end of period $(t+1)$ given that there are multiple items to be produced in period $(t+1)$.

Let us create an instance of MWIS in a chain of $(T-1)$- cliques as follows: For each $S'(t)$ there is a clique G_t with nodes $j_{t,k}$ $\forall k = 1,2,..n_t$, $t \leq T-1$ and an edge between each pair of nodes. Weight of node $j_{t,k} \in G_t$ is $c_{j_{t,k}}$, where $c_{j_{t,k}}$ is the savings in setup corresponding to $j_{t,k}$. Choosing a node from G_t represents producing the corresponding item at the end of period t and at the beginning of period $(t+1)$. If $j_{t,k} \in S'(t)$ and $j_{(t+1),k'} \in S'(t+1)$ represent the same item and if $|S'(t+1)| > 1$, there

is an edge connecting node$j_{t,k}$ in G_t to node $j_{(t+1),k'}$ in G_{t+1}, which refers to the condition that if $j_{t,k}$ is picked from G_t then $j_{(t+1),k'}$ cannot be picked from G_{t+1}.

Let us consider a solution of MWIS $\{j_{1,k_1}, j_{2,k_2}, \dots, j_{(T-1),k_{(T-1)}}\}$, which means there is no common edge between node $j_{t,k_t} \in G_t$ and $j_{(t+1),k_{(t+1)}} \in G_{(t+1)}$ $\forall t \leq T-1$. For each t, item representing node $j_{t,k_t} \in G_t$ is produced at the end of period t and the setup of item j_{t,k_t} is carried over from period t to period $(t+1)$. Thus $\{j_{1,k_1}, j_{2,k_2}, \dots, j_{(T-1),k_{(T-1)}}\}$ constitutes a solution of SCAP. Therefore, the problem of maximizing savings of setup cost reduces to the problem of choosing an MWIS in a chain of $T-1$)- cliques.

Consider an instance of MWIS in a chain of $(T-1)$- cliques $G_t$with nodes $\{j_{t,k} | k = 1,2,..n_t\} \forall t \leq T-1$ such that there is no edge of the type $(j_{t,k}, j_{(t+t'),k'} | 1 < t' \leq T-1-t, k' = 1,2,\dots,n_{(t+t')})$; for any node $j_{t,k} \in G_t$, there is at most one edge connecting node $j_{t,k}$ and $j_{(t+1),k'} | \, j_{(t+1),k'} \in G_{t+1}$, and at most one edge between node $j_{(t-1),k''} | \, j_{(t-1),k''} \in G_{t-1}$ and $j_{t,k}$. Weight of node $j_{t,k}$ is $c_{j_{t,k}}$.

Let's create an instance of SCAP as follows: For each node $j_{t,k} \in G_t \, \forall k = 1,2,..n_t, t \leq T-1$ there is an item $j_{t,k}$ that is produced in periods t and $(t+1)$ with setup cost $c_{j_{t,k}}$. Each edge $(j_{t,k}, j_{t,k'})$ in clique G_t refers to the condition that only one item can be produced at the end of period t and at the beginning of period $(t+1)$. Each clique corresponds to a period. For each edge $(j_{t,k}, j_{(t+1),k'})$ between clique G_t and G_{t+1}, the items $j_{t,k}$ and $j_{(t+1),k'}$ are identical and therefore, this item cannot be produced at the end of period t and $(t+1)$ at the same time. More precisely, if $j_{t,k}$ is produced at the end of period

t, it has to be produced at the beginning of period $(t+1)$. Again, if $j_{(t+1),k'}$ is produced at the end of period $(t+1)$, it has to be produced at the beginning of period $(t+2)$. If there is no edge between two nodes of $j_{t,k} \in G_t$ and $j_{(t+1),k'} \in G_{t+1}$, these two items can be produced at the end of their respective periods.

Let us consider a solution of SCAP $\left\{j_{1,k_1}, j_{2,k_2}, \dots, j_{(T-1),k_{(T-1)}}\right\}$ where item j_{t,k_t} is produced at the end of period t and the setup of item j_{t,k_t} is carried over from period t to $(t+1)$ in order to maximize the savings in setup cost. Since each j_{t,k_t} represents a node in $G_t \; \forall t \leq T-1$, there is no common edge between j_{t,k_t} and $j_{(t+1)k_{(t+1)}}$. Thus, $\left\{j_{1,k_1}, j_{2,k_2}, \dots, j_{(T-1),k_{(T-1)}}\right\}$ constitutes an MWIS. Hence, the problem of finding the MWIS in a chain $(T-1)$- cliques reduces to the problem of maximizing the savings of setup cost.

Therefore, the SCAP for inventory lot-sizing is equivalent to finding the MWIS in a chain $(T-1)$- cliques■

4.6 LP Relaxation of Model SCAP:

In this section, we shall show that the LP relaxation of the model SCAP gives integral solution.

Let A be the constraint matrix for the model SCAP which is a 0-1 matrix. Each row of matrix A represents a constraint and each column represents a variable. The constraint matrix A is feasible if it has one of the following properties.

Property 1: If $a_{i,j} = a_{i',j} = a_{i'',j'} = a_{i'',j''} = 1$ and $a_{i'',j} = 0$ then there exists at most one nonzero element among $a_{i,j'}$, $a_{i,j''}$, $a_{i',j'}$, $a_{i',j''}$ where $j \neq j' \neq j'', j \in G_t$ and $j', j'' \in G_{t+1} \forall t \leq T-1$

Proof of property 1: According to constraint (9), node $j \in G_t$ can be connected at most one node in G_{t+1}. Therefore, there exists at most one nonzero element among $a_{i,j'}$, $a_{i,j''}$, $a_{i',j'}$, $a_{i',j''}$. An example of this property is shown in Figure 4.4(a).

Property 2: If $a_{i,j} = a_{i',j} = a_{i'',j'} = a_{i'',j''} = a_{i'',j} = 1$ then $a_{i,j'} = a_{i,j''} = a_{i',j'} = a_{i',j''} = 0$ where $j \neq j' \neq j''$ and $j, j', j'' \in G_t \; \forall t \leq T-1$

Proof of property 2: Since $a_{i'',j} = a_{i'',j'} = a_{i'',j''} = 1$, node j, j', j'' belongs to the same clique. According to constraint (8), nodes in the same clique is represented by a single row of 1s. Therefore, $a_{i,j'} = a_{i,j''} = a_{i',j'} = a_{i',j''} = 0$. An example of this property is shown in Figure 4.4(b).

Figure 4.4: Example of constraint matrix showing (a) property 1 and (b) property 2

Theorem 2: Every $k \times k$ submatrix representing the linear constraints of the model SCAP is totally unimodular.

Proof: We prove by induction method that the constraint matrix A is totally unimodular. Note that a matrix is defined to be totally unimodular if and only if every square submatrix has determinant 0 or ± 1.

It is obvious that 1×1 submatrices have determinant either 0 or 1.

For 2×2 submatrices, we have either i) all four elements are zero in which case the determinant is also zero, or, ii) at least one element is zero in which case the determinant is plus or minus the product of two elements and thus its value is always 0 or 1.

Now, let us assume that all $k \times k$ submatrices of A have determinant 0 or 1.

Let us consider a $(k+1) \times (k+1)$ submatrix A_{k+1} of A. Three situations can arise:

i) A_{k+1} has a zero column, which means that the determinant of A_{k+1} is 0.

ii) A_{k+1} has at least one column with exactly one non-zero element. Suppose the t-th column has exactly one non-zero element which is located in the i-th row. So, $j_{i,t} = 1$. Now if we calculate the determinant with respect to column t, we get, $|A_{k+1}| = j_{i,t} A_k$, where A_k is the $k \times k$ submatrix resulting from the deletion of the t-th column and the i-th row from A_{k+1}. From the induction assumption, $|A_k| \in \{0, \pm 1\}$ and since $j_{i,t} = 1$, we have $|A_{k+1}| \in \{0,1\}$.

iii) Every column of submatrix A_{k+1} has at least two non-zero elements which are equal to 1. If every column of A_{k+1} has at least two non-zero elements, then one of the following holds:

a) A_{k+1} has a row with all elements equal to zero, which means that the determinant of A_{k+1} is 0.

b) A_{k+1} has at least one row which has exactly one non-zero element which are equal to 1. Suppose the i-th row has exactly one non-zero element which is located in the t-th column. So, $j_{i,t} = 1$. Now if we calculate the determinant with respect to row i, we get, $|A_{k+1}|=j_{i,t}A_k$. From the induction assumption, we have $|A_{k+1}| \in \{0, \pm 1\}$.

c) Every row of A_{k+1} has at least two non-zero elements which are equal to 1. If every row has at least two 1s and every column has at least two 1s then for 2×2 matrix the determinant is 0 and for 3×3 matrix the graph is infeasible according to property 1 and 2.

Let us assume that all $m \times m$ submatrices of A_{k+1} have either determinant 0 i.e., $|(A_{k+1})_m| = 0$ or the graph is infeasible.

Let us consider a $(m+1) \times (m+1)$ submatrix $(A_{k+1})_{m+1}$ of A_{k+1}. Suppose the i-th row has at least two 1s located in the j-th, j'-th ,…, j''-th column. So, $a_{i,j} = \cdots = a_{i,j'} = \cdots = a_{i,j''} = 1$. Now if we calculate the determinant with respect to row i, we get, $|(A_{k+1})_{m+1}|=a_{i,j}(A_{k+1})_m + \cdots + a_{i,j'}(A_{k+1})_m + \cdots + a_{i,j''}(A_{k+1})_m = 0$.

Therefore, if every row and every column of A_{k+1} has at least two non-zero elements and the matrix is feasible, the determinant is zero.

Hence, the determinant of every $k \times k$ submatrix of A is either 0 or 1. Therefore, matrix A is totally unimodular ■

ILP with totally unimodular constraint matrix are solved by their LP relaxation, which gives integer solution. According to Theorem 2, the SCAP has a totally unimodular

constraint matrix. The linear programing (LP) relaxation of the above ILP is obtained as follows:

Model: Relaxed SCAP

Max (6)

Subject to,

(7) through (10)

$z_{jt} \geq 0 \; \forall j, t$ (11)

Now we shall provide an alternate proof that the relaxed SCAP always provides integral optimum solution.

Theorem 3: There exists an integer $z_{jt} \; \forall j, t | c_j \in \mathbb{R}$, which is an optimum solution of the Relaxed SCAP.

Proof: Let σ be an optimal solution which has some period t such that $0 < z_{jt} < 1 \forall j$. Out of all such periods, take the first period and out of all such jobs in that period, take the one with highest savings. If there are multiple optimum solution, consider the optimal solution in which there are least fractional z_{jt} values. We shall show that there exists some σ' in which there are fewer fractional z_{jt} values. Let $\bar{c}_\sigma$ and $\bar{c}_{\sigma'}$ be the total setup cost savings associated with t solution σ and σ' respectively.

Case 1: $r_{j_1(t-1)} = r_{j_1 t} = 0$ and j_1 is the only item in $S'(t) | \; q_{j_1 t} = 1$. Let us create σ' from σ as follows:

$z_{jt'}(\sigma') = z_{jt'}(\sigma) \; \forall j \neq j_1, t' \neq t$

$z_{j_1 t}(\sigma') = 1$

$\bar{c}_{\sigma'} \geq \bar{c}_\sigma$

Therefore, σ' is not worse than σ and σ' has fewer fractional z_{jt} values.

Case 2: $r_{j_1(t-1)} = r_{j_1 t} = 0$ and $S'(t)$ has more than one element i.e, $q_{jt} = 1 \forall\, j \in S'(t)$. Let us assume that j_1 has the highest setup cost and $0 \le z_{jt} \le 1 \forall j \in S'(t)$. Let us create σ' from σ as follows:

$z_{jt'}(\sigma') = z_{jt'}(\sigma)\ \forall j, t' \neq t$

$z_{j_1 t}(\sigma') = 1;$

$z_{jt}(\sigma') = 0\ \forall\, j \in S'(t).$

Thus $\bar{c}_{\sigma'} \ge \bar{c}_{\sigma}$ and constraints (3) is not violated. Therefore, σ' is not worse than σ and σ' has fewer fractional z_{jt} values.

Case 3: $r_{jt} = r_{j(t+1)} = \cdots = r_{j(t+k)} = 1\ \forall k = 1,2,..(T-2-t)$ and $t < T$.

Given a solution $\sigma = \{z_{jt} | 0 < z_{jt} < 1\}$, we shall find an $\varepsilon, Set1$ and $Set2$ such that for $z'_{jt} = z_{jt} + \varepsilon\ \forall z_{jt} \in Set\ 1$ and $z'_{jt} = z_{jt} - \varepsilon\ \forall z_{jt} \in Set\ 2$ or $z''_{jt} = z_{jt} - \varepsilon\ \forall z_{jt} \in Set\ 1$ and $z''_{jt} = z_{jt} + \varepsilon\ \forall z_{jt} \in Set\ 2$ either $\sigma' = \{z'_{jt}\}$ or $\sigma'' = \{z''_{jt}\}$ will have at least one more integer value and $\bar{c}_{\sigma'}\ or\ \bar{c}_{\sigma''} \ge \bar{c}_{\sigma}$.

Step 1: Suppose j and j' are two elements of $S'(t) \forall t |\, j' \neq j$. Initialize two sets $Set1$ and $Set2$ as follows:

$Set1 = \{z_{jt} |\, r_{jt} = 1\}$ and $Set\ 2 = \{z_{j(t+1)} | r_{j(t-1)} = 1\ ,\ \max_{j' \in S'(t)} z_{j't} | j' \neq j\}$.

Step 2: Let $t = t + 1$. We compute $Sum = \sum_{j | z_{jt} \in Set1} c_j - \sum_{j | z_{jt} \in Set2} c_j$

If $Sum \ge 0$ and $r_{jt} = 1 | z_{jt} \in Set1$, augment $Set1$ and $Set2$ as follows:

$$Set1' = \min_{j', j'' \in S'(t-1)} \left\{ z_{j'(t-1)} \middle| z_{j'(t-1)} \in Set1,\ z_{j''(t-1)} \middle|\, z_{j''t} \in Set2 \right\}$$

$Set1 = (Set1 \cup Set1')$

$$Set2' = \left\{ z_{j(t+1)},\ \min_{j', j'' \in S'(t)} \left(z_{j't} \middle| r_{j'(t-1)} = 1,\ \max_{j'' \in S'(t)} \left(z_{j''t} | j'' \neq j \right) \right) \right\} \text{ and}$$

$Set2 = (Set2 \cup Set2') - (Set2 \cap Set2')$

If $Sum < 0$ and $r_{jt} = 1 | z_{jt} \in Set2$, augment $Set1$ and $Set2$ as follows:

$$Set1' = \left\{ z_{j(t+1)}, \min_{j', j'' \in S'(t)} \left(z_{j't} \middle| r_{j'(t-1)} = 1, \max_{j'' \in S'(t)} \left(z_{j''t} \middle| j'' \neq j \right) \right) \right\}$$

$Set1 = (Set1 \cup Set1') - (Set1 \cap Set1')$

$$Set2' = \min_{j', j'' \in S'(t-1)} \left\{ z_{j'(t-1)} \middle| z_{j'(t-1)} \in Set2,\ z_{j''(t-1)} \middle|\ z_{j''t} \in Set1 \right\}$$

$Set2 = (Set2 \cup Set2')$

If $Sum \geq 0$ and $r_{jt} = 0 | z_{jt} \in Set1$ or $Sum < 0$ and $r_{jt} = 0 | z_{jt} \in Set2$, go to step 3.

Step 3: If $Sum \geq 0$, then $z'_{jt}(\sigma') = z_{jt} + \varepsilon \ \forall z_{jt} \in Set\ 1$ and $z'_{jt}(\sigma') = z_{jt} - \varepsilon \ \forall z_{jt} \in Set\ 2$, where $\varepsilon = min\left(1 - max\left(z_{jt} \middle| z_{jt} \in Set\ 1\right), min\left(z_{jt} \middle| z_{jt} \in Set\ 2\right)\right)$.

If $Sum < 0$, then $z''_{jt}(\sigma'') = z_{jt} - \varepsilon \ \forall z_{jt} \in Set\ 1$ and $z''_{jt}(\sigma'') = z_{jt} + \varepsilon \ \forall z_{jt} \in Set\ 2$, where $\varepsilon = min\left(min\left(z_{jt} \middle| z_{jt} \in Set\ 1\right),\ 1 - max\left(z_{jt} \middle| z_{jt} \in Set\ 2\right)\right)$.

Therefore, σ' or σ'' is not worse than σ and σ' or σ'' has fewer fractional z_{jt} values ■

4.7 Numerical Example

Essentially, Theorem 3 uses an iterative ε-perturbation procedure, which converts a fractional solution to an integer solution that is not worse. To illustrate this iterative procedure, we use an example. Let us consider an SCAP where the following production schedule is given. $S(1) = \{1,2,3\}$, $S(2) = \{1,2,3,4\}$, $S(3) = \{1,2,4\}$, $S(5) = \{1,4\}$, and $S(5) = \{1\}$.

Items 1, 2, and 3 is produced in period 1 and 2. Thus, items 1, 2, and 3 are the eligible items to carryover from period 1 to period 2. Hence $S'(1) = \{1,2,3\}$. Similarly, item 1, 2, and 4 are produced in periods 2 and 3. Therefore, items 1, 2, and 4 are eligible to carryover from period 2 to period 3 and $S'(2) = \{1,2,4\}$. Let us formulate an undirected graph as shown in Figure 4.5, where each item that are allowed to carryover to the next period represents a node and each period represents a clique. The edges between two cliques states the condition that the corresponding nodes represents identical item and this item cannot be produced at the end of two consecutive periods at the same time. The corresponding z_{jt} values $\forall j, t$ are shown in the parenthesis along with each node (Figure 4.5). Let us assume that the cost savings associated with each item is $(c_1, c_2, c_3, c_4) = (10,8,6,5)$

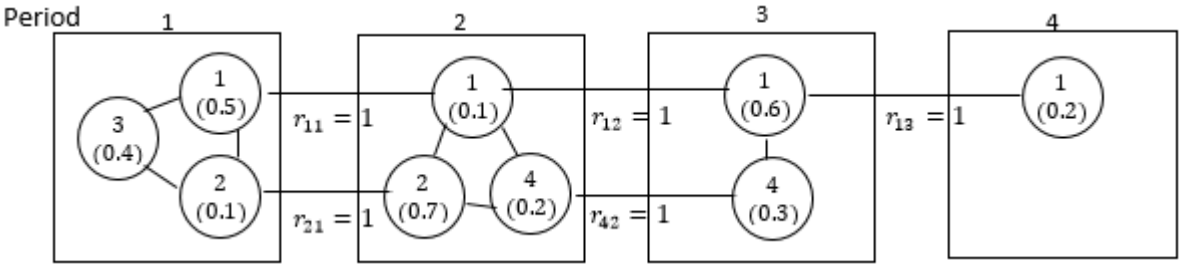


Figure 4.5: A simple undirected graph for the example problem

A step by step procedure of the first iteration is shown in Table 4.1:

Table 4.1: A step by step procedure of the first iteration

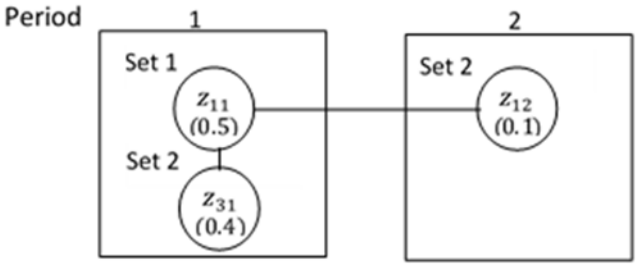
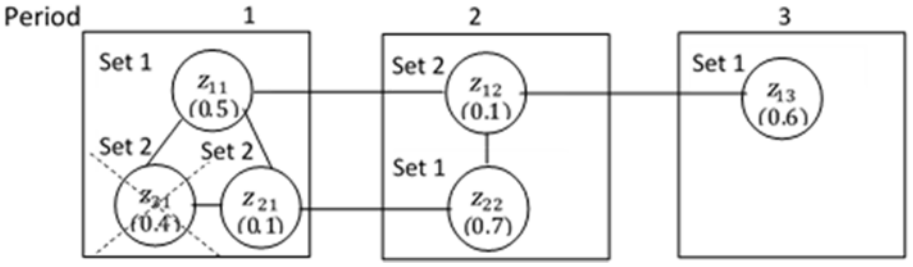
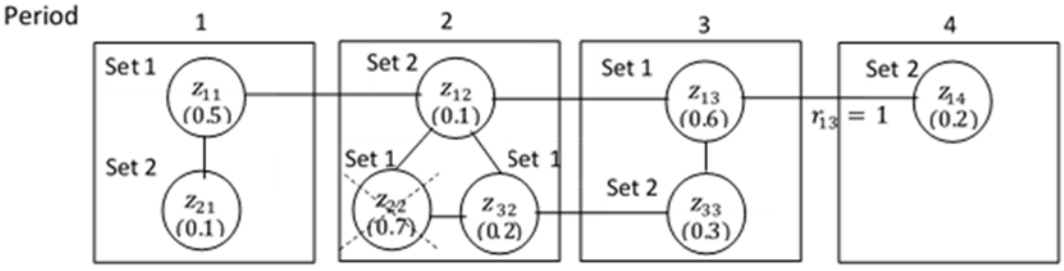
	Step1: Initialization:
$t = 1$	 $Sum = 10 - (6 + 8) = -4$
	Step 2: Augmentation:
$t = 2$	 $Sum = (10 + 8 + 10) - (8 + 10) = 10.$
$t = 3$	 $Sum = (10 + 6 + 10) - (8 + 10 + 6 + 10) = -8$
	Step 3: ε-perturbation
$t = 4$	$Sum < 0$ and $r_{14} = 0$, $\varepsilon = 0.2$. Total savings, $\bar{c}_\sigma = 25.3$. The z_{jt} values after iteration 1 is shown in Table 4.2 below:

Table 4.2: z_{jt} values $\forall j, t$ after iteration 1

Item (j)	Period (t)			
	1	2	3	4
1	z_{11}=0.3	z_{12}=0.3	z_{13}=0.4	z_{14}=0.4
2	z_{21}=0.3			
3				
4		z_{42}=0	z_{43}=0.5	

$\bar{c}_{\sigma'}$ =26.9. Thus, $\bar{c}_{\sigma'} \geq \bar{c}_{\sigma}$ and constraints (3) is not violated and σ' has fewer fractional z_{jt} value. Table 4.3 shows that the number of integer solution increases at least by 1 at each iteration until all of them becomes integer. The setup cost savings is also increases as the number of integer solution increases and the saving is maximum when there is no fractional solution remaining. Note that if there is only one job in a period in a SCAP, there will be no edge connecting the node representing that job in the MWIS problem which is equivalent to that SCAP. This special case satisfies the conditions of the SCAP formulated in this chapter and it is solved by the LP relaxation of SCAP.

Table 4.3: Results of iterations

Iteration	z_{11}	z_{12}	z_{13}	z_{14}	z_{21}	z_{22}	z_{31}	z_{42}	z_{43}	$\lvert\sigma\rvert$	$\lvert\sigma'\rvert$	Cost savings
0	0.5	0.1	0.6	0.2	0.1	0.7	0.4	0.2	0.3	9	0	25.3
1	0.3	0.3	0.4	0.4	0.3	0.7	0.4	0	0.5	8	1	26.9
2	0	0.6	0.1	0.7	0.6	0.4	0.4	0	0.8	7	2	28.4
3	0	0.7	0	0.8	0.6	0.4	0.3	0	0.9	6	3	29.3
4	0	1	0	0.8	1	0	0	0	0.9	2	7	30.5
5	0	1	0	0.8	1	0	0	0	1	1	8	31
6	0	1	0	1	1	0	0	0	1	0	9	33

4.8 Conclusion:

This Chapter shows an application of a special case of MWIS problem in the context of SCAP. We formulate the MWIS problem in a chain of cliques as an ILP model, and present its natural LP relaxation. We show that LP relaxation of a straightforward formulation of MWIS and solution of SCAP using that formulation gives fractional solution. We model the SCAP as a chain of cliques and show that the SCAP is equivalent to the problem of finding MWIS. The SCAP is formulated as an ILP model for a given production schedule to maximize the savings in the setup cost. We also prove that the constraint matrix of the ILP has a totally unimodular structure and the LP relaxation of the proposed ILP always provides integer optimum solution. We also give an alternative proof of integer solution of the relaxed ILP. Thus, the SCAP and the special case of the MWIS in a chain of cliques are solvable in polynomial time.

Acknowledgement:

The research of M. F. Baki and A. Azab is partially supported by Natural Sciences and Engineering Research Council (NSERC) Discovery Grants. M. F. Baki's research is also partially funded by the Research and Teaching Innovation Fund (RTIF), Odette School of Business, University of Windsor. This research has also been funded by Dr. A. Azab's internal faculty funds.

REFERENCES

Aarts, E., & Korst, J. (1989). *Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing*: John Wiley \\& Sons, Inc.

Alekseev, V. E. (2004). Polynomial algorithm for finding the largest independent sets in graphs without forks. *Discrete Applied Mathematics, 135*(1), 3-16. doi: https://doi.org/10.1016/S0166-218X(02)00290-1

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research, 246*(2), 345-378. doi: https://doi.org/10.1016/j.ejor.2015.04.004

Allahverdi, A., & Soroush, H. M. (2008). The significance of reducing setup times/setup costs. *European Journal of Operational Research, 187*(3), 978-984. doi: https://doi.org/10.1016/j.ejor.2006.09.010

Back, T., & Khuri, S. (1994, 27-29 Jun 1994). *An evolutionary heuristic for the maximum independent set problem.* Paper presented at the Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence.

Bitran, G. R., & Yanasse, H. H. (1982). Computational Complexity of the Capacitated Lot Size Problem. *Management Science, 28*(10), 1174-1186. doi: 10.1287/mnsc.28.10.1174

Borisovsky, P. A., & Zavolovskaya, M. S. (2003). Experimental Comparison of Two Evolutionary Algorithms for the Independent Set Problem. In S. Cagnoni, C. G. Johnson, J. J. R. Cardalda, E. Marchiori, D. W. Corne, J.-A. Meyer, J. Gottlieb, M. Middendorf, A. Guillot, G. R. Raidl & E. Hart (Eds.), *Applications of Evolutionary Computing: EvoWorkshops 2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, and EvoSTIM Essex, UK, April 14–16, 2003 Proceedings* (pp. 154-164). Berlin, Heidelberg: Springer Berlin Heidelberg.

Brendel, W., Amer, M., & Todorovic, S. (2011, 20-25 June 2011). Multiobject tracking as maximum weight independent set. Paper presented at the CVPR 2011.

Briskorn, D. (2006). A note on capacitated lot sizing with setup carry over. IIE Transactions, 38(11), 1045-1047. doi: 10.1080/07408170500245562

Chen, G. H., Kuo, M. T., & Sheu, J. P. (1988). An optimal time algorithm for finding a maximum weight independent set in a tree. BIT Numerical Mathematics, 28(2), 353-356. doi: 10.1007/bf01934098

Du, P., & Zhang, Y. (2016). A New Distributed Approximation Algorithm for the Maximum Weight Independent Set Problem. Mathematical Problems in Engineering, 2016, 10. doi: 10.1155/2016/9790629

Etzion, T., & Ostergard, P. R. J. (1998). Greedy and heuristic algorithms for codes and colorings. IEEE Transactions on Information Theory, 44(1), 382-388. doi: 10.1109/18.651069

Feo, T. A., Resende, M. G. C., & Smith, S. H. (1994). A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set. Operations Research, 42(5), 860-878. doi: 10.1287/opre.42.5.860

Florian, M., Lenstra, J. K., & Kan, A. H. G. R. (1980). Deterministic Production Planning: Algorithms and Complexity. Management Science, 26(7), 669-679. doi: 10.1287/mnsc.26.7.669

Friden, C., Hertz, A., & de Werra, D. (1990). Tabaris: An exact algorithm based on tabu search for finding a maximum independent set in a graph. Computers & Operations Research, 17(5), 437-445. doi: https://doi.org/10.1016/0305-0548(90)90048-C

Gardiner, E. J., Artymiuk, P. J., & Willett, P. (1997). Clique-Detection Algorithms for Matching Three-Dimensional Molecular Structures. Journal of Molecular Graphics and Modelling, 15(4), 245-253. doi: https://doi.org/10.1016/S1093-3263(97)00089-2

Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness: W. H. Freeman \& Co.

Gfeller, B., & Vicari, E. (2007). A randomized distributed algorithm for the maximal independent set problem in growth-bounded graphs. Paper presented at the Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing, Portland, Oregon, USA.

Grotschel, M., Lovász, L., & Schrijver, A. (1993). Geometric Algorithms and Combinatorial Optimization Springer, Berlin, Germany.

Haase, K. (1998). Capacitated Lot-Sizing with Linked Production Quantities of Adjacent Periods. In A. Drexl & A. Kimms (Eds.), Beyond Manufacturing Resource Planning (MRP II): Advanced Models and Methods for Production Planning (pp. 127-146). Berlin, Heidelberg: Springer Berlin Heidelberg.

Hifi, M. (1997). A Genetic Algorithm-Based Heuristic for Solving the Weighted Maximum Independent Set and Some Equivalent Problems. The Journal of the Operational Research Society, 48(6), 612-622. doi: 10.2307/3010225

Kako, A., Ono, T., Hirata, T., & Halldórsson, M. M. (2009). Approximation algorithms for the weighted independent set problem in sparse graphs. Discrete Applied Mathematics, 157(4), 617-626. doi: https://doi.org/10.1016/j.dam.2008.08.027

Matsui, T. (2000). Approximation Algorithms for Maximum Independent Set Problems and Fractional Coloring Problems on Unit Disk Graphs, Berlin, Heidelberg.

Minty, G. J. (1980). On maximal independent sets of vertices in claw-free graphs. Journal of Combinatorial Theory, Series B, 28(3), 284-304. doi: https://doi.org/10.1016/0095-8956(80)90074-X

Pal, S. K., & Sarma, S. S. (2012). Graph Coloring Approach for Hiding of Information. Procedia Technology, 4, 272-277. doi: https://doi.org/10.1016/j.protcy.2012.05.042

Paschalidis, I. C., Huang, F., & Lai, W. (2015). A message-passing algorithm for wireless network scheduling. IEEE/ACM Trans. Netw., 23(5), 1528-1541. doi: 10.1109/tnet.2014.2338277

Sahling, F., Buschkühl, L., Tempelmeier, H., & Helber, S. (2009). Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a fix-and-optimize heuristic. Computers & Operations Research, 36(9), 2546-2553. doi: https://doi.org/10.1016/j.cor.2008.10.009

Sox, C. R., & Gao, Y. (1999). The capacitated lot sizing problem with setup carry-over. IIE Transactions, 31(2), 173-181. doi: 10.1023/a:1007520703382

Tempelmeier, H., & Buschkühl, L. (2009). A heuristic for the dynamic multi-level capacitated lotsizing problem with linked lotsizes for general product structures. OR Spectrum, 31(2), 385-404. doi: 10.1007/s00291-008-0130-y

Valiente, G. (2003). A New Simple Algorithm for the Maximum-Weight Independent Set Problem on Circle Graphs, Berlin, Heidelberg.

Xiao, M., & Nagamochi, H. (2016). An exact algorithm for maximum independent set in degree-5 graphs. Discrete Applied Mathematics, 199(Supplement C), 137-155. doi: https://doi.org/10.1016/j.dam.2014.07.009

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Concluding remarks

This dissertation presents a comprehensive study on inventory lot-sizing problem and develops dynamic programming based algorithms, mathematical models, and competitive heuristic solution approaches to solve the problem. In Chapter 2, an efficient linear-time algorithm for ELSPs as well as SMBSPs employing lists and stack data structures is developed. The approach in this dissertation is different from the well established linear time algorithms by Wagelmans *et al.* (1992) (based on geometric approach) and Aggarwal and Park (1993) (based on Monge arrays). The theoretical properties of the developed algorithm are derived and an experimental comparison with the algorithms previously developed by Aggarwal and Park (1993), Wagelmans *et al.* (1992), and Albers and Brucker (1993) is presented. The results indicate that the developed algorithm shows a maximum of 40.54% and 51.40% and an average of 29.84% and 39.27% performance improvement with respect to CPU time over the Wagelmans *et al.* (1992) and Aggarwal and Park (1993) algorithms, respectively. Additionally, the developed algorithm is implemented for SMBSP where it shows a maximum of 29.03% and an average of 25.75% improvement over Albers and Brucker (1993) algorithm. Moreover, the number of times the "If" statements (basic action) are executed by Algorithm 1 is less than that of the algorithms proposed by Wagelmans *et al.* (1992) and Aggarwal and Park (1993) for all the test data sets. The condition of the outer For loop in Algorithm 1 is checked exactly $(T - 1)$ times. Since the inner While loop is nested inside the inner For loop, hence, if the Inner For loop does not run, the inner While loop is not executed. The condition of inner For

loop in line (7) of Algorithm 1 is checked at most $(2T-4)$ times over all possible cases but this loop runs at most $(T-1)$ times. Most of the "If" statements are nested inside the inner For and Inner While loops, which explains why Algorithm 1 checks the "If" conditions fewer number of times than the other comparable algorithms. Furthermore, Algorithm 1 performs fewer list operations than the ones by Wagelmans *et al.* (1992) and Albers and Brucker (1993). Again, the number of matrix cells to be evaluated by Algorithm 1 is less than that by Aggarwal and Park (1993). Therefore, with regards to every metric of comparisons, the new algorithm shows better result than the algorithms proposed by Wagelmans *et al.* (1992), Aggarwal and Park (1993), and Albers and Brucker (1993). In other words, it is obvious that Algorithm 1 outperforms the other three algorithms. Algorithm 1, therefore, is faster.

In chapter 3, first we present an item DW decomposition of the classical MLCLSP. We then extend the MLCLSP by allowing set-up carryover and backlogging. We also include emission capacity constraints and refer the problem as MLCLSP with Set-up Carryover, Backlogging and Emission control (MLCLSP-SCBE). We develop an MILP model for the MLCLSP-SCBE and apply item DW decomposition of the proposed MILP formulation embedded with a CG procedure. We propose a dynamic programming approach to solve each of the sub-problems and develop a CA heuristic to generate feasible solutions. An ILP model is proposed to determine the setup carryover plan optimally for a given production schedule. The solution approach is hybridized with an LP based improvement procedure in order to refine the solution and hence improve the solution quality given by the DW decomposition. The performance of the proposed heuristic for classical MLCLSP is tested by comparing the average percentage of deviation from

optimality with that of Tempelmeier and Derstroff (1996). Overall average optimality gap improves by 20% as compared to Tempelmeier and Derstroff (1996). The quality of the heuristic for MLCLSP_SCBE is tested based on 225 small instances taken from literature. Four new data sets containing a total of 96 problem instances with increasing problem size is generated. Computational results show that the proposed optimization framework provides competitive solutions within a reasonable time.

Chapter 4 shows an application of a special case of MWIS problem in the context of SCAP. We formulate the MWIS problem in a chain of cliques as an ILP model, and present its natural LP relaxation. We show that LP relaxation of a straightforward formulation of MWIS and solution of SCAP using that formulation gives fractional solution. We model the SCAP as a chain of cliques and show that the SCAP is equivalent to the problem of finding MWIS. The SCAP is formulated as an ILP model for a given production schedule to maximize the savings in the setup cost. We also prove that the constraint matrix of the ILP has a totally unimodular structure and the LP relaxation of the proposed ILP always provides integer optimum solution. We also give an alternative proof of integer solution of the relaxed ILP. Thus, the SCAP and the special case of the MWIS in a chain of cliques are solvable in polynomial time.

5.2 Future Works

Future work will address the case of parallel machines, which makes the MLCLSP_SCBE formulation much more relevant for industrial applications. If there exists multiple identical machines within a machine group, it may be economically attractive to have some machines continuously setup over several periods for a product with high

regular demand while the setup of the other machines producing products with low and irregular demand is frequently changed. It might also be interesting to broadening the computational basis of the numerical evaluation. Another interesting line of future research involves extending the MLCLSP_SCBE model to a production–distribution system with emissions. Another extension would be to incorporate a cap-and-trade mechanism like Hua, Cheng, and Wang (2011) do in an EOQ setting.

The DW decomposition based heuristic solution approach is depicted in this dissertation. In future, other decomposition methods such as Benders Decomposition can be implemented to solve the MLCLSP with different extensions. Also metaheuristic techniques such as Genetic Algorithm, Tabu Search, and Simulated Annealing may be used to investigate if the percentage gap from optimality improves.

In this work, all problem parameters including demand are assumed to be known with absolute certainty which may not be acceptable for certain markets and products. To take into account uncertainty we can consider stochastic dynamic programming, robust optimization and even Discrete event simulation.

REFERENCES

Aggarwal, A., & Park, J. K. (1993). Improved algorithms for economic lot size problems. *Oper. Res., 41*(3), 549-571. doi: 10.1287/opre.41.3.549

Albers, S., & Brucker, P. (1993). The complexity of one-machine batching problems. *Discrete Applied Mathematics, 47*(2), 87-107. doi: http://dx.doi.org/10.1016/0166-218X(93)90085-3

Hua, G., Cheng, T. C. E., & Wang, S. (2011). Managing carbon footprints in inventory management. *International Journal of Production Economics, 132*(2), 178-185. doi: https://doi.org/10.1016/j.ijpe.2011.03.024

Sox, C. R., & Gao, Y. (1999). The capacitated lot sizing problem with setup carry-over. *IIE Transactions, 31*(2), 173-181. doi: 10.1023/a:1007520703382

Tempelmeier, H., & Derstroff, M. (1996). A Lagrangean-Based Heuristic for Dynamic Multilevel Multiitem Constrained Lotsizing with Setup Times. *Management Science, 42*(5), 738-757.

Wagelmans, A., Hoesel, S. V., & Antoon, K. (1992). Economic Lot Sizing: An O(n log n) Algorithm That Runs in Linear Time in the Wagner-Whitin Case. *Operations Research, 40*, S145-S156. doi: 10.2307/3840844

VITA AUCTORIS

NAME:	Nusrat Tarin Chowdhury
PLACE OF BIRTH:	Dhaka, Bangladesh
YEAR OF BIRTH:	1985
EDUCATION:	Bangladesh University of Engineering and Technology, B.Sc., Dhaka, Bangladesh, 2008
	Bangladesh University of Engineering and Technology, M.Sc., Dhaka, Bangladesh, 2011