공학박사학위논문

Integer Optimization and Approximate Dynamic Programming Approaches for Lot-sizing and Scheduling Problem with Sequence-dependent Setups

순서의존적 작업준비가 있는 생산계획 문제에 대한 정수 최적화 및 근사 동적 계획법 기반 해법

2022 년 8 월

서울대학교 대학원

산업공학과

이 연 수

Integer Optimization and Approximate Dynamic Programming Approaches for Lot-sizing and Scheduling Problem with Sequence-dependent Setups

순서의존적 작업준비가 있는 생산계획 문제에 대한 정수 최적화 및 근사 동적 계획법 기반 해법

지도교수 이 경 식

이 논문을 공학박사 학위논문으로 제출함

2022 년 5 월

서울대학교 대학원

산업공학과

이 연 수

이연수의 공학박사 학위논문을 인준함

2022 년 6 월

위 원 장 홍 성 필 (인)

부위원장 이 경 식 (인)

위 원 홍 유 석 (인)

위 원 박 경 철 (인)

위 원 이 동 호 (인)

Abstract

Integer Optimization and Approximate Dynamic Programming Approaches for Lot-sizing and Scheduling Problem with Sequence-dependent Setups

Younsoo Lee

Department of Industrial Engineering

The Graduate School

Seoul National University

Lot-sizing and scheduling problem, an integration of the two important decision making problems in the production planning phase of a supply chain, determines both the production amounts and sequences of multiple items within a given planning horizon to meet the time-varying demand with minimum cost. Along with the growing importance of coordinated decision making in the supply chain, this integrated problem has attracted increasing attention from both industrial and academic communities. However, despite vibrant research over the recent decades, the problem is still hard to be solved due to its inherent theoretical complexity as well as the evolving complexity of the real-world industrial environments and the corresponding manufacturing processes. Furthermore, when the setup activity occurs in a sequence-dependent manner, it is known that the problem becomes considerably more difficult.

This dissertation aims to propose integer optimization and approximate dynamic programming approaches for solving the lot-sizing and scheduling problem with sequence-dependent setups. Firstly, to enhance the knowledge of the structure of the problem which is strongly NP-hard, we consider a single-period substructure of the problem. By analyzing the polyhedron defined by the substructure, we derive new families of facet-defining inequalities which are separable in polynomial time via solving maximum flow problems. Through the computational experiments, these inequalities are demonstrated to provide much tighter lower bounds than the existing ones. Then, using these results, we provide new integer optimization models which can incorporate various extensions of the lot-sizing and scheduling problem such as setup crossover and carryover naturally. The proposed models provide tighter linear programming relaxation bounds than standard models. This leads to the development of an efficient linear programming-based heuristic algorithm which provides a primal feasible solution quickly. Finally, we devise an approximate dynamic programming algorithm. The proposed algorithm incorporates the value function approximation approach which makes use of both the tight lower bound obtained from the linear programming relaxation and the upper bound acquired from the linear programming-based heuristic algorithm. The results of computational experiments indicate that the proposed algorithm has advantages over the existing approaches.

Keywords: Lot-sizing and scheduling problem, Sequence-dependent setup, Integer optimization, Approximate dynamic programming, Valid inequality, Extended Formulation

Student Number: 2018-32331

Contents

Abstract **i**

Contents **iii**

List of Tables **vii**

List of Figures **ix**

Chapter 1 Introduction **1**

1.1 Backgrounds 1

1.2 Integrated Lot-sizing and Scheduling Problem 6

1.3 Literature Review 9

1.3.1 Optimization Models for LSP 9

1.3.2 Recent Works on LSP 14

1.4 Research Objectives and Contributions 16

1.5 Outline of the Dissertation 19

Chapter 2 Polyhedral Study on Single-period Substructure of Lot-sizing and Scheduling Problem with Sequence-dependent Setups **21**

2.1 Introduction 22

2.2 Literature Review 27
2.3 Single-period Substructure 30
2.3.1 Assumptions 31
2.3.2 Basic Polyhedral Properties 32
2.4 New Valid Inequalities 37
2.4.1 `S-STAR` Inequality 37
2.4.2 Separation of `S-STAR` Inequality 42
2.4.3 `U-STAR` Inequality 47
2.4.4 Separation of `U-STAR` Inequality 49
2.4.5 General Representation of the Inequalities 52
2.5 Extended Formulations 55
2.5.1 Single-commodity Flow Formulations 55
2.5.2 Multi-commodity Flow Formulations 58
2.5.3 Time-flow Formulations 59
2.6 Computational Experiments 63
2.6.1 Experiment Settings 63
2.6.2 Experiment Results on Single-period Instances 65
2.6.3 Experiment Results on Multi-period Instances 69
2.7 Summary 73

Chapter 3 New Optimization Models for Lot-sizing and Scheduling Problem with Sequence-dependent Setups, Crossover, and Carryover 75

3.1 Introduction 76
3.2 Literature Review 78

3.3 Integer Optimization Models 80
3.3.1 Standard Model (ST) 82
3.3.2 Time-flow Model (TF) 84
3.3.3 Comparison of (ST) and (TF) 89
3.3.4 Facility Location Reformulation 101
3.4 LP-based Naive Fixing Heuristic Algorithm 104
3.5 Computational Experiments 108
3.5.1 Test Instances 108
3.5.2 LP Bound 109
3.5.3 Computational Performance with MIP Solver 111
3.5.4 Performance of LPNF Algorithm 113
3.6 Summary 115

Chapter 4 Approximate Dynamic Programming Algorithm for Lot-sizing and Scheduling Problem with Sequence-dependent Setups 117

4.1 Introduction 118
4.1.1 Markov Decision Process 118
4.1.2 Approximate Dynamic Programming Algorithms 121
4.2 Markov Decision Process Reformulation 124
4.3 Approximate Dynamic Programming Algorithm 127
4.4 Computational Experiments 131
4.4.1 Comparison with (TF-FL) Model 131
4.4.2 Comparison with Big Bucket Model 134
4.5 Summary 138

Chapter 5 Conclusion **139**

5.1 Summary and Contributions 139

5.2 Future Research Directions 141

Bibliography **145**

Appendix A Pattern-based Formulation in Chapter 2 **159**

Appendix B Detailed Test Results in Chapter 2 **163**

Appendix C Detailed Test Results in Chapter 3 **169**

국문초록 **173**

List of Tables

Table 2.1 Nomenclature for big bucket models 24

Table 2.2 Test results on single-period instances: Relative formulation size 68

Table 2.3 Test results on single-period instances: Computation time . . 69

Table 2.4 Test results on single-period instances: Number of added inequalities 69

Table 2.5 Test results on multi-period instances: Relative formulation size 72

Table 2.6 Test results on multi-period instances: Computation time . . 72

Table 2.7 Test results on multi-period instances: Number of added inequalities 73

Table 3.1 Nomenclature 81

Table 3.2 Summary of the computational performance of the models . . 112

Table 4.1 Test results for the ADP algorithm: $\lambda = 2$ 132

Table 4.2 Test results for the ADP algorithm: $\lambda = 3$ 133

Table 4.3 Test results for the ADP algorithm: $\lambda = 5$ 134

Table 4.4 Comparison of the models and the solution algorithms 137

Table B.1 Test results on single-period instances: LP gap (%) 164

Table B.2 Test results on single-period instances: Closed gap (%) 165

Table B.3 Test results on single-period instances: Ratio of instances with LP gap below certain value 166
Table B.4 Test results on multi-period instances: LP strength 167
Table B.5 Test results on multi-period instances: Ratio of instances with LP strength above certain value 168

Table C.1 Comparison of the LP bound of the models 169
Table C.2 Comparison of the computational performance of the models: *Gap* and *Opt* 170
Table C.3 Comparison of the computational performance of the models: *#Node* and *Time* 171
Table C.4 Test results for the LPNF algorithm 172

List of Figures

Figure 1.1 Supply chain planning network (adapted from Stadtler et al., 2015) 2

Figure 1.2 Illustration of the production schedules 6

Figure 1.3 Integrated lot-sizing and scheduling problem (adapted from Stadtler et al., 2015) 7

Figure 1.4 Illustration of a big bucket model 10

Figure 1.5 Illustration of a small bucket model 12

Figure 1.6 Illustration of a hybrid model 13

Figure 2.1 Production sequence and its representation as a big bucket model with a cycle for each period 22

Figure 2.2 Illustration of the `S-STAR` inequality 37

Figure 2.3 Illustration of the separation procedure of `S-STAR` inequality 45

Figure 2.4 Construction of the network for the separation procedure of `U-STAR` inequality 52

Figure 2.5 Illustration of the extended formulations 61

Figure 2.6 Test results on single-period instances: LP gap 66

Figure 2.7 Test results on single-period instances: Closed gap 67

Figure 2.8 Test results on single-period instances: Ratio of instances with LP gap below certain values 68
Figure 2.9 Test results on multi-period instances: LP strength 70
Figure 2.10 Test results on multi-period instances: Ratio of instances with LP strength above certain values 71

Figure 3.1 Illustration of the network flow corresponding to a production plan 84
Figure 3.2 Illustration of the constraint (3.2c) 86
Figure 3.3 Definition of the variable w'_{ijs} 88
Figure 3.4 Illustration of the fractional solution of `(ST)` 93
Figure 3.5 Illustration of the fixing procedure 107
Figure 3.6 Comparison of the LP bound of the models: LP Gap (%) . . 110
Figure 3.7 Comparison of the LP bound of the models: Gap Closed (%) 111
Figure 3.8 Comparison of the computational performance of the models 112
Figure 3.9 Test results for the LPNF algorithm 114

Figure 4.1 Illustration of the dynamic programming recursion 126
Figure 4.2 Illustration of ADP algorithm 130
Figure 4.3 Test results for the ADP algorithm: Solution quality 135
Figure 4.4 Test results for the ADP algorithm: Computation time . . . 135

Chapter 1

Introduction

1.1 Backgrounds

A supply chain is defined as "a network of organizations that are involved in different processes and activities that produce value in the form of products and services in the hands of the ultimate consumer" (Christopher, 2016). Specifically, the supply chain of a manufacturing organization typically comprises four stages: procurement, production, distribution, and sales. Within each of these stages, various decisions which range from short-term (e.g., weekly) to long-term (e.g., yearly) should be made. As illustrated in Figure 1.1 (Stadtler et al., 2015), these various decisions have influence on each other. Therefore, they should be made coordinated to obtain high efficiency and profitability of the whole supply chain.

The importance of the integration of the decision problems is emphasized when they are closely related. *Lot-sizing problem* and *scheduling problem* are examples that are closely intertwined decision problems in the production stage. The lot-sizing problem is to determine the size of *lots* which refer to bundles of identical product which is produced in one production run without being interrupted by the production of other products. Before producing the lot, a *setup activity* is needed which refers to preparation operations of the production machine such as cleansing

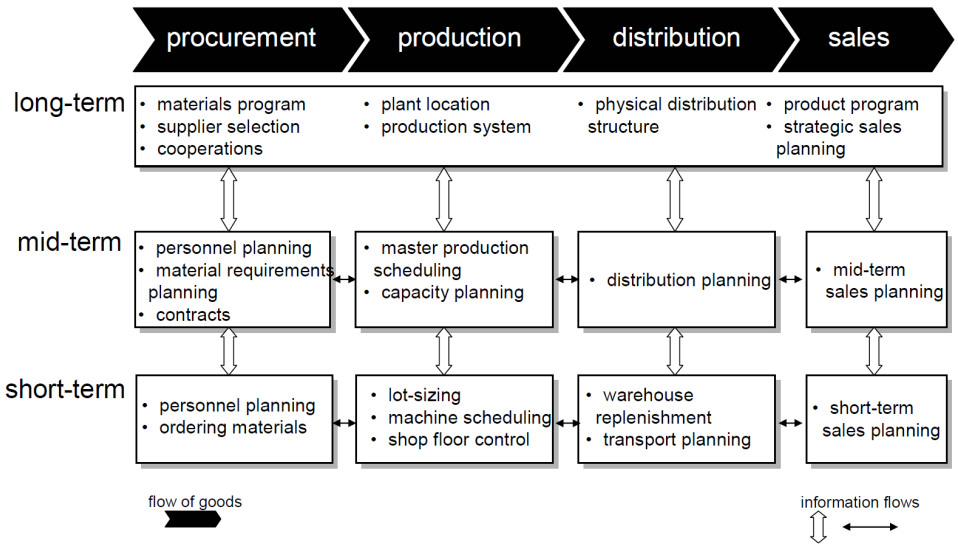


Figure 1.1: Supply chain planning network (adapted from Stadtler et al., 2015)

or adjustment. Because the setup incurs the corresponding costs, too frequent setups are not desirable generally. Therefore, when deciding the optimal size of lots, the trade-off between the inventory holding and setup costs must be considered, that is, as the sizes of lots increase, the number of required setups and the corresponding setup cost reduce whereas the amount of inventory and the corresponding holding cost increase.

Research on the lot-sizing problem dates back to the seminal work of Harris (1913) which dealt with the single-item lot-sizing problem with constant demand and sufficiently large production capacity over an infinite horizon. In this simple setting, known as the economic order quantity (EOQ) model, the optimal lot size can be derived as a closed-form solution. Later, Rogers (1958) generalized EOQ model by considering the limited production capacity and multiple items, whereas Wagner & Whitin (1958) generalized it by considering the time-varying demand

over a finite planning horizon. The former was shown to be an NP-hard problem (Florian et al., 1980; Bitran & Yanasse, 1982), while the latter can be solved within polynomial time (Zangwill, 1966; Federgruen & Tzur, 1991).

Manne (1958) studied the problem considering both the limited production capacity and time-varying demand extensions, denoted as *capacitated lot-sizing problem*, which can be defined as follows: Within a planning horizon consisting of multiple time periods, the production capacity and demand of each item are given for each period. Under these settings, the objective is to allocate the limited production capacity efficiently to meet the demand at the minimum cost which includes production, setup, and inventory holding costs. We give a simple illustrative example as follows.

Example 1.1. *Consider an instance of capacitated lot-sizing problem with five items and two periods. Let* $\mathbf{d}_1 = (15, 40, 0, 10, 0)$ *and* $\mathbf{d}_2 = (0, 5, 15, 10, 10)$ *be the demand vectors of items for period 1 and 2, respectively, which are given in time unit. Assume that the production capacity is 100. Then, consider the following solutions which are represented as the vector of production amounts:*

$$\mathbf{x} := (\mathbf{x}_1, \mathbf{x}_2) = \big((15, 40, 0, 10, 0), (0, 5, 15, 10, 10)\big)$$
$$\mathbf{x}' := (\mathbf{x}'_1, \mathbf{x}'_2) = \big((15, 45, 15, 20, 0), (0, 0, 0, 0, 10)\big)$$
$$\mathbf{x}'' := (\mathbf{x}''_1, \mathbf{x}''_2) = \big((15, 45, 15, 20, 10), (0, 0, 0, 0, 0)\big)$$

Both $\mathbf{x}$ *and* $\mathbf{x}'$ *are feasible because they cover the entire demand without loss, while* $\mathbf{x}''$ *is infeasible because the capacity restriction for period* 1 *is violated. In terms of the costs,* $\mathbf{x}$ *does not incur inventory holding costs because it uses a lot-for-*

lot strategy, whereas $\mathbf{x}'$ *incurs holding costs for items 2,3, and 4. On the other hand,* $\mathbf{x}'$ *requires fewer setups than* $\mathbf{x}$*, which leads to lower setup costs.*

Despite its simplicity, capacitated lot-sizing problem with multiple items is shown to be strongly NP-hard (Chen & Thizy, 1990). Even when only a single item is considered, the problem belongs to NP-hard as demonstrated by Florian et al. (1980) and Bitran & Yanasse (1982).

As addressed by Trigeiro et al. (1989), one of the important variants of capacitated lot-sizing problem is the additional consideration of the setup time. Besides the corresponding costs, the setup activity may require a certain amount of time and consume a portion of the production capacity. If the setup takes a considerable amount of time or incurs substantial costs, it should be considered explicitly and carefully in the planning stage. Long setup times are common in many manufacturing processes. In the fine-chemical process industry considered by Sung & Maravelias (2008), for instance, the setup takes a significant amount of time, as it includes several activities such as cleansing and testing. Similarly, in the flat-panel display manufacturing process studied by Lee & Lee (2020), the setup for highly automated equipment can take even longer than a day.

Moreover, in many real manufacturing processes, the setup occurs in a *sequence-dependent* manner; that is, its cost and time depend on both the item that was produced before and that which will be produced subsequently. The sequence-dependent setups occur in various industries from food or beverage production (Ferreira et al., 2010; Claassen et al., 2016; Larroche et al., 2021) to high-tech industries (Chiang & Fu, 2009; Xiao et al., 2015; Lee & Lee, 2020).

To properly address the sequence-dependent setups, the production sequences

need to be considered explicitly. The traditional capacitated lot-sizing problem, however, does not consider the production sequence of the lots within each period, although it determines which items to be produced in which period. In fact, the sequencing decisions have been often ignored when establishing the production plan and decided on the shop floor. Alternatively, they are addressed by solving another decision making problem, called the *scheduling problem* (Pinedo, 2012). The scheduling problem aims to determine the optimal schedule of machine(s) considering the given set of jobs (lots) and their due dates.

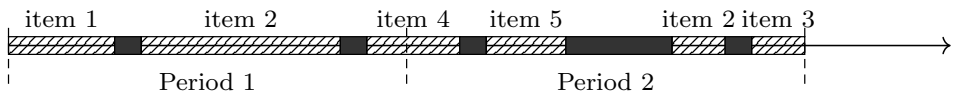
However, it is not enough to consider the two problems separately in sequence. For instance, when the setups are sequence-dependent, the available production capacity is determined after the sequencing decisions are made. Meanwhile, to determine the optimal sequence, the set of jobs to be produced within each period should be given which is determined by lot-sizing decisions. However, the optimal lot-sizing decisions can be made only when the available production capacity is determined. Therefore, the separated decision-making procedure can make the quality of solutions deteriorate significantly as illustrated in the example below. Therefore, these two decisions should be integrated and considered simultaneously.

Example 1.2. *Let us consider the solution* $\mathbf{x}$ *given in Example 1.1 and the following production sequences* $\mathbf{s}$ *and* $\mathbf{s}'$.

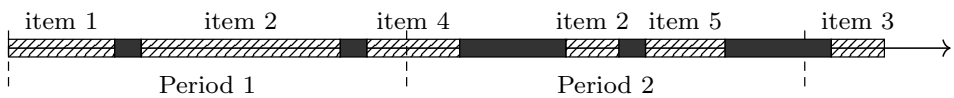
$$\mathbf{s} := (1 \to 2 \to 4) \to (4 \to 5 \to 2 \to 3)$$

$$\mathbf{s}' := (1 \to 2 \to 4) \to (4 \to 2 \to 5 \to 3)$$

Assume that the setup times are sequence-dependent and classified into short and



(a) A feasible production schedule corresponding to $\mathbf{s}$



(b) An infeasible production schedule corresponding to $\mathbf{s}'$

Figure 1.2: Illustration of the production schedules

long setups: the setup from a high-indexed item to a low-indexed item takes a long time, whereas the converse takes a short time. In practice, this situation can occur, for instance, in a painting shop. Suppose that the index of an item indicates the darkness of the color of the item. Then, the setup from the darker item to the lighter one requires further work such as cleansing of a painting spray. In this setting, the production schedules corresponding to sequence $\mathbf{s}$ *and* $\mathbf{s}'$ *are represented in Figures 1.2(a) and 1.2(b), respectively. There is a single long setup in Figure 1.2(a) and the corresponding solution is feasible with respect to the capacity constraint. On the other hand, long setups are presented twice in Figure 1.2(b) which leads to the violation of the capacity constraint.*

1.2 Integrated Lot-sizing and Scheduling Problem

Along with the growing importance of integrated decision making in the supply chain, the consolidation of the lot-sizing problem and scheduling problem also has become an important issue because of their closely intertwined nature. In the early 1990s, several researchers including Potts & Van Wassenhove (1992), Lasserre (1992),

and Dauzere-Peres & Lasserre (1994) emphasized the integration of the two problems. Most of these earlier studies proposed solution approaches based on the iterative procedure that solves the two problems repeatedly, exchanging the solution information with each other. However, these approaches have a drawback that it is not guaranteed to obtain optimal decisions in view of the integrated problem.

Later, the problems are further integrated into a monolithic problem, called a *lot-sizing and scheduling problem* (LSP), which has received increasing attention over the recent decades both from industrial and academic communities. See Figure 1.3 which illustrates the integrated LSP within the supply chain (Stadtler et al., 2015). LSP simultaneously determines the sizes of production lots and the production sequences within a given planning horizon. The objective is to meet the time-varying demand while minimizing the total cost which is the sum of various components such as production cost, inventory holding cost, backlog penalty cost, and setup cost. In

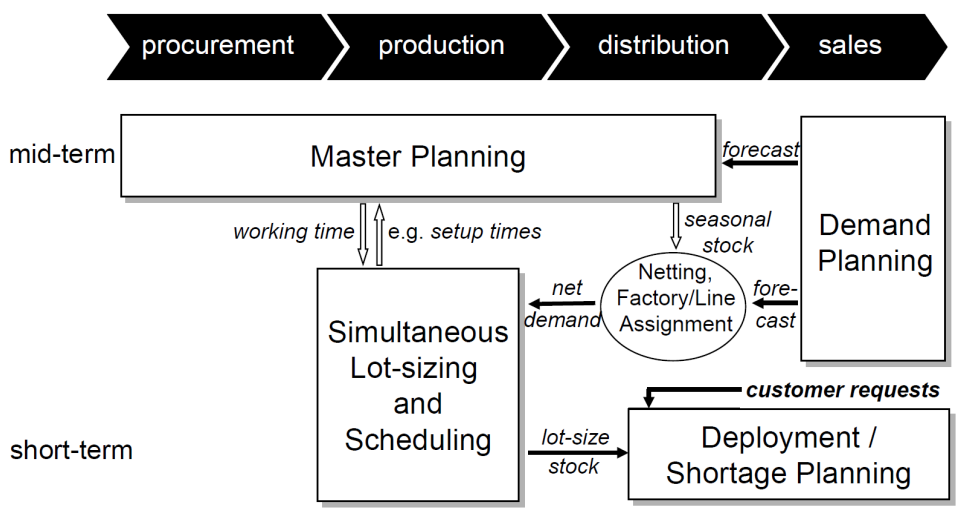


Figure 1.3: Integrated lot-sizing and scheduling problem (adapted from Stadtler et al., 2015)

addition, various extensions are often considered to model practical problems as will be introduced later. Earlier reviews on the integrated LSP can be found in Drexl & Kimms (1997) and Zhu & Wilhelm (2006).

LSP has been widely investigated by a great deal of study in the operations research community resulting in substantial advancements. However, despite the effort made by researchers, there remain outstanding issues not yet completely resolved. The sources of the difficulty in conquering LSP are twofold. The first one lies in the inherent theoretical complexity of LSP. As shown by Chen & Thizy (1990), the problem is strongly NP-hard even without the scheduling decisions. When the scheduling decisions need to be considered explicitly, for instance, to incorporate the sequence-dependent setups, the computational complexity further increases. This is because, even when the lot sizes of items are fixed, the remaining scheduling problem still belongs to NP-hard. This can be demonstrated by the fact that the traveling salesman problem which is a popular NP-hard problem can be reduced to the problem. Therefore, LSP is naturally NP-hard which makes it unpromising to devise solution algorithms that efficiently solve this problem.

The second source of the difficulty lies in rapidly evolving industries and the corresponding growth in complexity of the manufacturing processes. Because real-world manufacturing processes have a variety of unique characteristics, there is no unified solution methodology that is capable of resolving all LSP instances from various industries. Consequently, active research on optimization models and solution approaches tailored to solve real problems continues.

This dissertation aims to provide modeling frameworks and solution approaches that are suitable for LSP with sequence-dependent setups. We provide new opti-

mization models based on the time-flow modeling approaches which are derived from the analysis of substructures of the problem. Moreover, we devise an approximate dynamic programming (ADP) algorithm with a value function approximation procedure that uses bound information of states. The proposed algorithm shows competitiveness in solving LSP instances from both the real-world industry and literature.

1.3 Literature Review

In this section, we review some fundamental literature on optimization models as well as the recent works on the LSP. The literature relevant to the contents of each chapter is reviewed there.

1.3.1 Optimization Models for LSP

There is abundant research that proposed various optimization models to formulate the integrated LSP (see, e.g., Pochet & Wolsey, 2006; Copil et al., 2017, for the comprehensive review on different models). Specifically, we focus on LSP models which take into account of the sequence-dependent setups explicitly.

For most of the LSP models, the planning horizon is discretized into several time buckets, although there exist some alternative frameworks such as continuous time models. The discretized modeling frameworks are natural because, in practice, the demand and inventory amounts are investigated and updated in a periodic manner. For instance, when a firm establishes a three-month production plan, the demand forecast and update of each product are on a daily basis. The amounts of inventory are also inspected at the end of each day. Therefore, it is natural to divide the

three-month planning horizon into several daily buckets.

Depending on the relative size of the time buckets, the models are classified into big bucket, small bucket, and hybrid models. Big bucket models of LSP are natural extensions of the conventional capacitated lot-sizing problem (Manne, 1958). In these models, the planning horizon is divided into multiple periods of length equal to the granularity of demand occurrence, for example, days. In each period, multiple items can be produced and their production sequence should be determined. This makes the difference between the big bucket models and the traditional capacitated lot-sizing problem model.

In big bucket models, the production sequence within a bucket is usually represented as a cycle (Gupta & Magnusson, 2005) as illustrated in Figure 1.4. A dummy item 0 is defined to represent the start and end of the production sequence within each bucket. Additional variables and constraints for the sequencing decisions need to be introduced to the model. In this regard, formulations for various routing problems such as capacitated vehicle routing problem (CVRP) or traveling salesman

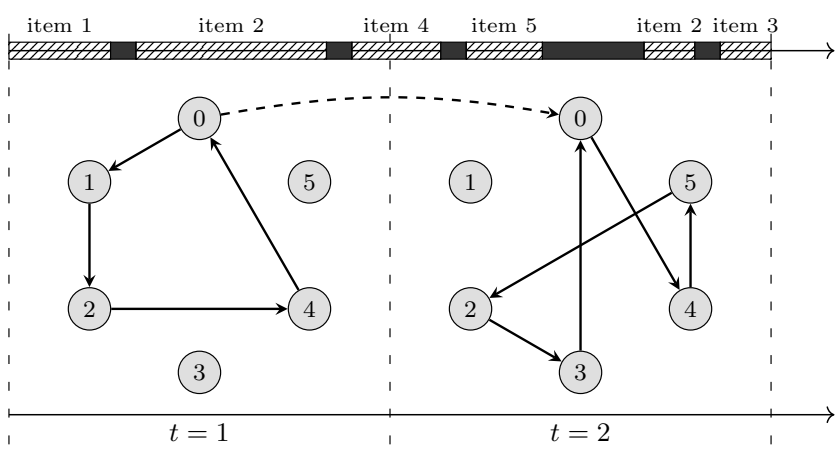


Figure 1.4: Illustration of a big bucket model

problem (TSP) can be naturally adapted for big bucket models.

One of the earliest big bucket models presented by Haase (1996) is called CLSD which is an abbreviation of *capacitated lot-sizing problem with sequence-dependent setup*. To incorporate the sequence-dependent setup costs in the model, the authors adapted Miller–Tucker–Zemlin constraints (Miller, Tucker, et al., 1960) originally proposed for TSP. Later, various big bucket models that differ mainly in the manner of representing the cycle have been proposed. Haase & Kimms (2000) proposed a pattern-based formulation that uses a set of predefined efficient sequences as a pattern set. Guimarães et al. (2014) and Sarin et al. (2011) proposed big bucket models of LSP based on the single-commodity and multi-commodity flow formulations, respectively. Both formulations are natural applications of the similar formulation proposed for TSP. Various big bucket models were investigated and compared computationally by Guimarães et al. (2014).

Contrary to the big bucket models, the small bucket models further discretize the time periods into several shorter buckets, for example, hours, which are used as the unit of production activities such as the production or setup. Therefore, compared with the big bucket models, the decisions to be made within a single bucket is much simpler in small bucket models. As a trade-off, the number of buckets of small bucket models is greater than that of the big bucket models.

Discrete lot-sizing and scheduling problem (DLSP) which is the most representative small bucket model allows only one item to be produced within a single time bucket. Moreover, it assumes that the full capacity of each bucket is used for the production (all-or-nothing assumption). In this regard, unlike big bucket models, the production sequence within each bucket does not need to be considered. In addition,

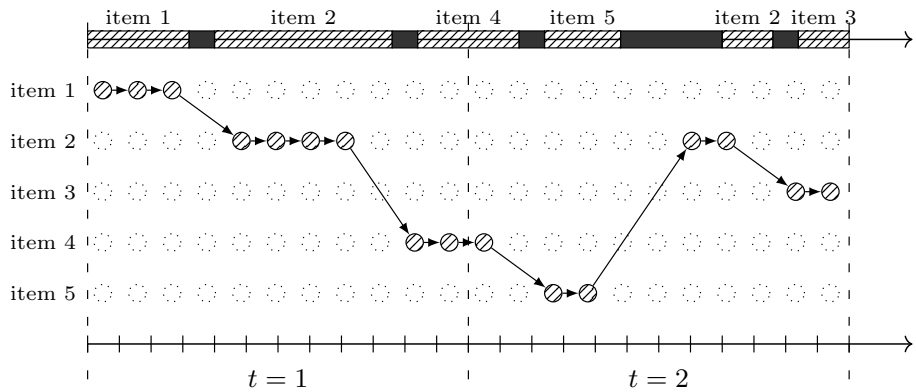


Figure 1.5: Illustration of a small bucket model

because the sequence of the buckets is fixed, the production sequence represented as a path can be obtained straightforwardly as illustrated in Figure 1.5. DLSP with sequence-dependent setup costs was firstly proposed by Fleischmann (1994). Later, Salomon et al. (1997) further considered sequence-dependent setup times. The authors transformed the problem into the TSP with time windows and solved it with a dynamic programming (DP) approach. There exist other small bucket models such as *continuous setup lot-sizing problem* (CSLP) (Karmarkar & Schrage, 1985) and *proportional lot-sizing problem* (PLSP) (Drexl & Haase, 1995) which have received relatively little attention from the researchers.

One alternative model is *general lot-sizing and scheduling problem* (GLSP), which is often called a hybrid of the big and small bucket models because it uses a two-level time structure as illustrated in Figure 1.6. In GLSP, the planning horizon is divided into multiple *macroperiods* the lengths of which are fixed similar to the big bucket models. Each macroperiod is then further divided into several *microperiods* the lengths of which are variables to be determined. External dynamics such as the arrival of the demand or inspection of the inventory levels are modeled at the end

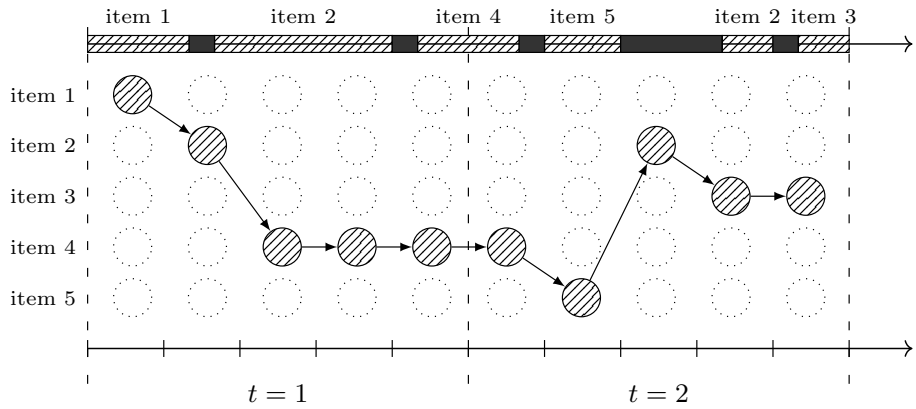


Figure 1.6: Illustration of a hybrid model

of the macroperiods, while internal dynamics such as the production amounts and the start/end of the setups are modeled within the microperiods.

GLSP and small bucket models are similar in a sense that both models discretize the natural time periods into smaller ones. Therefore, the production sequence is represented as a path in both models. The main difference between the two modeling frameworks is that the length of microperiods of GLSP is variable, whereas the length of buckets of small bucket models is fixed. Therefore, contrary to DLSP in which the production amounts are automatically obtained if a path is given, the microperiods in GLSP only capture the sequence of the production and the production amounts should be additionally determined. The number of buckets of GLSP is smaller than that of small bucket models.

The GLSP was first proposed by Fleischmann & Meyr (1997). As its name indicates, the GLSP can be regarded as a generalization of various models of LSP, because it has a two-level time structure consisting of macroperiods of fixed length and microperiods of variable length. The authors formally formulated the model,

clarified the relationship between the GLSP and the existing models, and devised a heuristic algorithm that employs a local-search algorithm. Later, Koçlar & Süral (2005) indicated that the original GLSP provided by Fleischmann & Meyr (1997) has a minor limitation and corrected it. A more comprehensive review on GLSP and its variants is provided in Chapter 3 where GLSP-based optimization models are provided.

1.3.2 Recent Works on LSP

The recent research on LSP includes studies dealing with various practical industrial problems (for example, Rıos-Solıs et al., 2020; Lee & Lee, 2020) and devising efficient solution approaches for general problem instances (for example, Guimarães et al., 2013; Carvalho & Nascimento, 2022). Solution approaches can be categorized into exact and heuristic approaches. The latter can be further classified into metaheuristic and matheuristic, that is, mathematical-programming-based heuristic algorithms. Due to the difficulty in solving the problem exactly, most of the proposed solution approaches are heuristic algorithms. Relatively recent and comprehensive reviews of the LSP with sequence-dependent setups can be found in Guimarães et al. (2014) and Copil et al. (2017).

Rıos-Solıs et al. (2020) addressed LSP in mold-injection production processes which can be classified as a bi-level LSP. To solve problem instances from real-world industry, a decomposition-based heuristic algorithm was proposed. Lee & Lee (2020) studied LSP in a flat-panel display industry which involves unique characteristics regarding the manufacturing process such as production run limits and maximum loading time conditions. To incorporate these complex constraints, the authors pro-

posed an extended formulation based on DLSP model which provides tighter lower bounds than the standard DLSP. Using the advantages of tighter bounds, the formulation is solved by a relax-and-fix algorithm which successfully solved real-world instances in a reasonable amount of time. The fruit beverage industry which is characterized by the temporal cleansing steps was studied by Toscano et al. (2019) and Toscano et al. (2020) where the decomposition-based and MIP-based heuristics were proposed, respectively. Cervantes-Sanmiguel et al. (2021) studied LSP in plastic injection manufacturing systems and proposed a two-stage heuristic algorithm. Koch et al. (2022) studied LSP with parallel machines inspired by a real-world tire industry. The authors proposed a decomposition-based matheuristic algorithm that can solve industrial-scale problem instances successfully. Oyebolu et al. (2017) studied LSP in the biopharmaceutical manufacturing process where the setup costs and times are substantial. The authors proposed problem-specific heuristic algorithms based on the genetic algorithm.

Carvalho & Nascimento (2022) addressed parallel machine LSP with nontriangular sequence-dependent setups and setup carryover. The authors devised matheuristic algorithms by hybridizing mathematical programming and local search heuristics. Melega et al. (2020) studied a two-stage lot-sizing, scheduling and cutting stock problem in which the cutting decision is made in the first stage, while the lot-sizing and scheduling decisions are made in the second stage. Specifically, the authors, considering the sequence-dependent setups in both stages, proposed a heuristic algorithm combining a column generation approach and a relax-and-fix heuristic to deal with their integrated problem. Mahdieh et al. (2018) considered the LSP with nontriangular sequence-dependent setups, setup crossover, and carryover, proposing

a multi-commodity-based big bucket model as an extension of the model presented in Clark et al. (2014). Gicquel, Lisser, et al. (2014) reformulated DLSP models as quadratic binary programs and applied a semidefinite relaxation approach to obtain tightened lower bounds. Together with the valid inequalities they further improved the quality of the lower bounds. Their approaches are successful in solving small-size instances, whereas a significant amount of computation time is required. Gicquel & Minoux (2015) proposed a cut-and-branch approach for LSP based on the DLSP model. They proposed a new family of valid inequalities and their exact and heuristic separation algorithms.

Contrary to most of the above-mentioned research which provided the integrated solution approaches, there are other solution approaches such as hierarchical approaches and iterative approaches to jointly tackle the two problems. See Maravelias & Sung (2009) and Alves et al. (2021) for these types of solution approaches.

1.4 Research Objectives and Contributions

The main objective of this dissertation is to enhance the ability to solve LSP with sequence-dependent setups using integer optimization approaches and ADP algorithms.

Firstly, we study the single-period substructure of the problem to enhance the knowledge of the structure of the problem. Contrary to the problem where the setups are sequence-independent, the single-period substructure of LSP with sequence-dependent setups has not been investigated in previous research which is our particular motivation. Therefore, we conduct polyhedral analysis on the single-period substructure and derive new families of valid inequalities and extended formulations

which are useful in tightening the linear programming (LP) relaxation bounds. We compare the strength of the bounds of the proposed inequalities and extended formulations with those of the existing ones by conducting computational experiments.

Secondly, based on the provided results on the extended formulation, we propose novel optimization models for LSP with sequence-dependent setups which use a set of decision variables representing the time flow. With these time-flow variables, it is demonstrated that the models can easily incorporate various extensions such as setup crossover and carryover. After further tightening the proposed models using the well-known reformulation techniques, we compare the strength of the models with the existing models. Furthermore, we devise a LP-based heuristic algorithm that can provide feasible solutions quickly. The performance of the proposed models and heuristic algorithm is tested using both instances from the real-world industry and those from the previous literature.

Thirdly, we propose an ADP algorithm to efficiently solve LSP with sequence-dependent setups. One of the deficiencies of the traditional DP approach is the so-called *curse-of-dimensionality* issue, that is, the number of states can easily explode as the problem dimension increases (Powell, 2007). Therefore, it becomes hard to evaluate the value of exponentially many states with the traditional DP recursion approaches. To alleviate this drawback, our ADP algorithm adapts the value function approximation approach to approximate the value of the states. The proposed value function approximation approach uses both the lower and upper bound values of the states. Therefore, the ADP algorithm enjoys the advantages of both the tight lower bound values obtained by the proposed optimization model and the upper bound value which can be obtained in a short time by the LP-based heuristic algorithm.

The performance of the ADP algorithm is also tested with various problem instances.

The contributions of the dissertation are threefold:

1. *Polyhedral study on the single-period substructure.*

    - We propose new families of valid inequalities for LSP with sequence-dependent setups by analyzing the single-period substructure, that is, `S-STAR` and `U-STAR` inequalities.
    - The proposed inequalities are demonstrated to define facets of single-period substructure under some conditions. Furthermore, the polynomial-time separation algorithms are also presented.
    - We provide a new type of extended formulation which is shown to provide the same lower bound as that of the original formulation with all `S-STAR` inequalities added.
    - Computational experiment results indicate that the proposed inequalities and formulations are effective in tightening the LP relaxation bounds.

2. *Novel integer optimization models incorporating various extensions of LSP.*

    - We propose novel integer optimization models which are called as time-flow models. The proposed models can incorporate setup crossover and carryover which are important modeling extensions of LSP.
    - The time-flow model is demonstrated to provide much tighter LP relaxation bounds than that of the standard GLSP-based model.
    - From the computational experiments, it is demonstrated that the proposed model has advantages compared with the standard model in terms

of tightness and solvability with the standard mixed integer programming (MIP) solver.

- We propose an LP-based heuristic algorithm that is based on the time-flow model. The proposed algorithm is shown to provide feasible solutions quickly.

3. *Approximate dynamic programming algorithm.*

   - We devise an ADP algorithm for LSP with sequence-dependent setups to mitigate the "curse-of-dimensionality" issue of the traditional DP approaches.
   - We devise a value function approximation scheme that estimates the value of each state using both the lower and upper bounds, without recursive evaluation of future states.
   - Computational experiment results indicate that the proposed ADP algorithm has computational benefits over the state-of-the-art optimization model solved by the standard commercial MIP solver.

1.5 Outline of the Dissertation

The remainder of this dissertation is organized as follows.

- In Chapter 2, the single-period substructure of the big bucket model of LSP with sequence-dependent setups is investigated. By conducting a polyhedral study, new families of facet-defining inequalities which can be separated in polynomial time are derived. In addition, new extended formulations are proposed and compared with the existing formulations. The proposed inequalities

and formulations are shown to facilitate tightening LP relaxation bound.

- In Chapter 3, novel optimization models which can incorporate setup carryover and crossover are provided. Compared with the existing models, the newly proposed models show advantages in both theoretical and computational aspects. Furthermore, LP-based heuristic algorithms are also provided in this chapter with the corresponding experimental results.
- In Chapter 4, we present an ADP algorithm based on the time-flow model. The value function approximation approach which is incorporated in the ADP algorithm is introduced. The results of computational experiments which demonstrate the proposed algorithm has computational benefits over the commercial solver are also provided.
- In Chapter 5, we summarize the results of the dissertation and discuss possible future research directions.

Chapter 2

Polyhedral Study on Single-period Substructure of Lot-sizing and Scheduling Problem with Sequence-dependent Setups

In this chapter, we propose new valid inequalities and extended formulations for the LSP, which are derived by investigating the single-period substructure of the problem. By conducting a polyhedral study on the single-period substructure, we derive two new families of valid inequalities and identify their facet-defining conditions. Additionally, we demonstrate that these inequalities can be separated in polynomial time. After introducing the existing extended formulations for the problem, we provide new extended formulations, called time-flow formulations, and compare the theoretical strengths of the various formulations and valid inequalities, including the proposed ones. Finally, we conduct computational experiments to demonstrate the effectiveness of the proposed inequalities and formulations. The test results indicate that the proposed inequalities and extended formulations facilitate tightening the LP relaxation bounds.

2.1 Introduction

Firstly, we present a generic mathematical formulation of the big bucket models of LSP with sequence-dependent setups. Let us consider the problem with sets of items $\mathcal{I} = \{1, \ldots, I\}$ and periods $\mathcal{T} = \{1, \ldots, T\}$. We additionally define a fictitious item 0 to represent the start and end of the production sequence, and let $\mathcal{I}_0 = \mathcal{I} \cup \{0\}$. Then, the production sequence within each period is represented as a cycle including item 0, as illustrated in Figure 2.1.

As shown in the figure, each item corresponds to a node, whereas the setup between the two items corresponds to an arc. Throughout this chapter, the terms item and node are used interchangeably, and setup and arc are also used as such. Additionally, for a single period, we define a directed graph $\mathcal{G} = (\mathcal{I}_0, \mathcal{A}_0)$, where the arc set is defined as $\mathcal{A}_0 := \{(i, j) : i \in \mathcal{I}_0, j \in \mathcal{I}_0, i \neq j\}$. Also, we let $\mathcal{A} := \{(i, j) : i \in \mathcal{I}, j \in \mathcal{I}, i \neq j\}$. For sets of nodes $S, T \subseteq \mathcal{I}_0$, we denote $E(S : T)$ as the set of arcs (i, j), such that $i \in S$ and $j \in T$. Using the definition of $E(S : T)$,

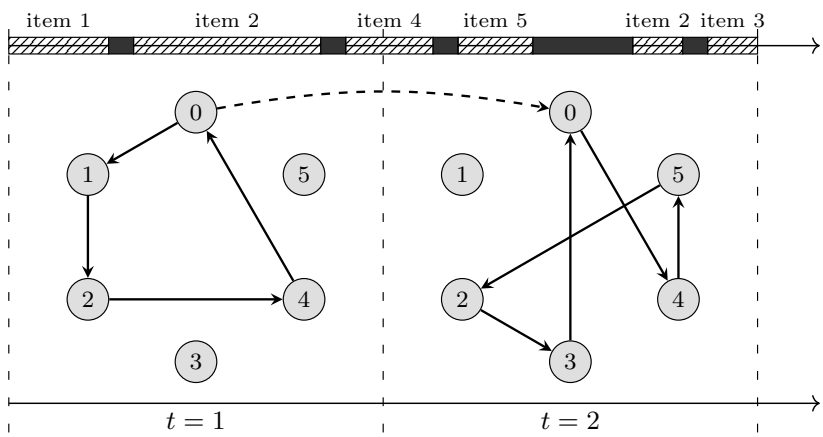


Figure 2.1: Production sequence and its representation as a big bucket model with a cycle for each period

we also can define $E(S) := E(S : S)$, the set of arcs with both endpoints in set S, $\delta^+(S) := E(S : \mathcal{I}_0 \setminus S)$, set of outgoing arcs from S, $\delta^-(S) := E(\mathcal{I}_0 \setminus S : S)$, set of incoming arcs to S, and $\delta(S) := \delta^+(S) \cup \delta^-(S)$, set of arcs with one endpoint in S and another in $\mathcal{I}_0 \setminus S$.

Throughout the exposition, $i, j \in \mathcal{I}$, $t \in \mathcal{T}$ are used as indices. Let hc_{it}, bc_{it}, pc_{it}, and d_{it} denote the unit inventory holding cost, backlogging cost, production cost, and demand for item i and period t, respectively. The setup cost and time between items i and j in period t are denoted by sc_{ijt} and st_{ijt}, respectively. For notational convenience, we also define st_{i0t} and st_{0it} and let their values be zero. Production capacity of period t, given in time units, is denoted by K_t, whereas the unit production time of item i is denoted by a_i.

Let s_{it} and b_{it} be the decision variables representing the inventory and backlog amounts of item i at the beginning of period t, respectively. The initial inventory and backlog amounts of item i are denoted by s_{i0} and b_{i0}, respectively, and are assumed to be zero. Variable x_{it} represents the production amount of item i in period t. The binary variable y_{it} is equal to one if item i is produced in period t. The binary variable z_{ijt} is equal to one if the setup from item i to item j occurs in period t. Moreover, let z_{0it} and z_{i0t} be the binary variables which represent whether item i is the first and last item produced in period t, respectively. The notations used are summarized in Table 2.1. We use boldface to denote vectors and matrices; for example, $\mathbf{x} = (x_{it})_{i \in \mathcal{I}, t \in \mathcal{T}}$.

Table 2.1: Nomenclature for big bucket models

Sets	
$\mathcal{I}$	Set of items which are indexed by i and j; $\mathcal{I} = \{1, \ldots, I\}$
$\mathcal{I}_0$	Set of items including the fictitious item 0; $\mathcal{I}_0 = \mathcal{I} \cup \{0\}$
$\mathcal{T}$	Set of time periods which are indexed by t; $\mathcal{T} = \{1, \ldots, T\}$
Parameters	
hc_{it}	Inventory holding cost of item i in period t
bc_{it}	Backlogging cost of item i in period t
pc_{it}	Production cost of item i in period t
d_{it}	Demand of item i in period t
sc_{ijt}	Cost incurred when setup occurs from item i to j in period t
st_{ijt}	Time needed for setup from item i to j
K_t	Production capacity of period t given in time unit
a_i	Production time per unit of item i
Variables	
s_{it}	Inventory amount of item i at the end of period t, $s_{i0} = 0$
b_{it}	Backlog amount of item i at the end of period t, $b_{i0} = 0$
x_{it}	Production amount of item i in period t
y_{it}	$= 1$ if item i is produced in period t
z_{ijt}	$= 1$ if setup from item i to j occurs in period t
z_{0it} (z_{i0t})	$= 1$ if item i is the first (last) produced item in period t

The generic mathematical formulation of the big bucket models of LSP can be written as follows (Guimarães et al., 2014):

$$\text{minimize} \quad \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\Big(hc_{it}s_{it} + bc_{it}b_{it} + pc_{it}x_{it}\Big) + \sum_{(i,j)\in\mathcal{A}}\sum_{t\in\mathcal{T}} sc_{ijt}z_{ijt} \tag{2.1a}$$

$$\text{subject to} \quad s_{it-1} - b_{it-1} + x_{it} = d_{it} + s_{it} - b_{it} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{2.1b}$$

$$\sum_{i\in\mathcal{I}} a_i x_{it} + \sum_{(i,j)\in\mathcal{A}} st_{ijt}z_{ijt} \leq K_t \qquad \forall t \in \mathcal{T} \tag{2.1c}$$

$$x_{it} \leq K_t y_{it} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{2.1d}$$

$$z_{i0t} = z_{0it+1} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \setminus \{T\} \tag{2.1e}$$

$$\sum_{j\in\mathcal{I}_0\setminus\{i\}} z_{jit} = \sum_{j\in\mathcal{I}_0\setminus\{i\}} z_{ijt} = y_{it} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{2.1f}$$

$$\text{Do not include cycles without item 0} \qquad \forall t \in \mathcal{T} \tag{2.1g}$$

$$x_{it},\ s_{it} \geq 0,\ y_{it} \in \{0,1\} \qquad \forall i \in \mathcal{I}, \forall t \in \mathcal{T} \tag{2.1h}$$

$$z_{ijt} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}, \forall t \in \mathcal{T} \tag{2.1i}$$

The objective function (2.1a) is the sum of the inventory holding, backlogging penalty, production, and setup costs, the total of which must be minimized. Constraints (2.1b) are balance equations between the demand, inventory, backlog, and production amounts. Constraints (2.1c) ensure that the sum of the production and setup times does not exceed the available capacity in each period. Constraints (2.1d) indicate that an item can only be produced if the corresponding setup occurs. Constraints (2.1e) indicate that the setup for the last item in the previous period is

carried over to the next period. Constraints (2.1f) logically link the binary variables to ensure a balanced flow of setups. Constraints (2.1h)–(2.1i) ensure the domain of variables.

As mentioned previously, a production sequence within each period is represented as a cycle on $\mathcal{G}$, including item 0. To ensure the validity of the cycle, the constraints (2.1g) which prevent cycles without item 0 are necessary. One possible option for the constraints (2.1g) is the *generalized subtour elimination constraints* (`GSEC`s, Toth & Vigo, 2002) which are written as follows:

$$\sum_{(j,i)\in\delta^-(S)} z_{jit} \geq y_{kt} \qquad \forall S \subseteq \mathcal{I},\ k \in S,\ t \in \mathcal{T} \tag{2.2}$$

`GSEC`s ensure that an item contained in S can be produced only if at least one incoming arc from the outside of S is selected. Note that there are several other alternatives, such as the Miller–Tucker–Zemlin formulation (Miller, Tucker, et al., 1960) or the single-commodity flow formulation (Gavish & Graves, 1978) which can also eliminate invalid cycles using additionally defined decision variables. These formulations, with additional variables other than those used in (2.1), are denoted as *extended formulations*.

Efficiently solving LSP with sequence-dependent setups seems unlikely because even the parts of the problem cannot be solved easily. In particular, it is well-known that even with a single item, the capacitated lot-sizing problem is already NP-hard without the scheduling decisions (Bitran & Yanasse, 1982). Moreover, even with the given lot-sizing decisions, that is, fixed $\mathbf{x}$, the remaining scheduling problem is NP-hard because it can be reduced to the traveling salesman problem (TSP). Because of

the difficulty inherited from both, LSP with sequence-dependent setups is strongly NP-hard.

One possible approach to tackle NP-hard problems is to exploit the substructures of the problems using polyhedral analysis. There are many cases where valid inequalities and extended formulations which are the results of the polyhedral analysis play a major role in solving NP-hard problems (Wolsey, 2020). Regarding the big bucket models of LSP, some research has been conducted on substructures. However, most of these studies address the single-item substructure, where sequence-dependent setups between different items cannot be incorporated. In this regard, by incorporating sequence dependency, we study the *single-period substructure* of LSP which is formally described in Section 2.3.

The remainder of this chapter is organized as follows. In Section 2.2, we review the relevant literature. In Section 2.3, a single-period substructure of LSP is formally presented, and the basic polyhedral properties are provided. In Section 2.4, we provide new families of valid inequalities and discuss their properties. In Section 2.5, we propose new extended formulations and compare them with the existing formulations. In Section 2.6, we present the results of the computational experiments. In Section 2.7, we summarize the chapter and provide concluding remarks.

2.2 Literature Review

In this section, we narrow the scope of the literature review, that is, we focus on polyhedral studies, such as valid inequalities and extended formulations of LSP and the related problems.

Most polyhedral studies on the substructure of LSP have primarily considered

the *single-item* structure. For instance, Barany et al. (1984) studied a single-item uncapacitated lot-sizing problem (ULSP) and provided a complete linear description of the convex hull of the ULSP in its original space using valid inequalities, denoted as (l, S)–inequalities. Krarup & Bilde (1977) and Eppen & Martin (1987) proposed extended formulations for ULSP which can provide an optimal solution by solving their LP relaxations, based on the facility location and shortest path reformulations, respectively.

Wolsey (1989) studied the single-item ULSP with start-up costs, while Constantino (1996) studied the capacitated version of the problem. The authors proposed several families of valid inequalities. The facet-defining conditions and efficient separation algorithms are also provided. Pochet & Wolsey (1994) studied ULSP with various extensions such as backlogging, start-up costs, and constant limited capacity. Assuming the Wagner-Whitin cost structure (Wagner & Whitin, 1958), the authors provided integral polyhedra, extended formulations, and separation algorithms of these extensions. Subsequently, Küçükyavuz & Pochet (2009) provided an explicit description of the convex hull of the ULSP with backlogging in the original space without the assumption of Wagner-Whitin cost structure. Leung et al. (1989) and Van Vyve (2007) addressed a single-item capacitated problem with constant capacity. They analyzed the polyhedral structure, proposed facet-defining inequalities, and devised polynomial-time solution algorithms. The results of the aforementioned studies on single-item substructures have been successfully adapted to generalized problems with multiple items or time-varying capacity.

Contrary to the single-item cases, there have been only a limited number of studies on the *single-period* substructure, and even those studies did not consider

the sequence-dependent setups. Miller et al. (2003b) studied the single-period relaxation of the capacitated lot-sizing problem with sequence-independent setups. By incorporating multiple items competing for a limited production capacity, the authors derived valid inequalities and their facet-defining conditions. They considered a special case in Miller et al. (2003a), where the demand and setup times were constant for all items. In this special case, the authors proposed a polynomial-time algorithm and derived an extended formulation. Akartunalı et al. (2016) considered two-period relaxation of the big bucket models and proposed cut generating procedure to cut off given fractional solutions. More recently, Doostmohammadi & Akartunalı (2018) studied the two-period substructure and derived facet-defining inequalities, but assumed zero setup times. Unfortunately, these results cannot be directly applied to LSP because the sequence-dependent setups are not considered. To the best of our knowledge, the studies on the single-period substructure of LSP incorporating sequence-dependent setups, are deficient.

Meanwhile, the solution set of the single-period LSP is closely related to that of routing-type problems such as the *capacitated vehicle routing problem* (CVRP) (Toth & Vigo, 2002) in that, both problems select a subset of items to produce (or customers to visit) from the given sets and determine the sequence of the production (or sequence of the visit). Therefore, the valid inequalities and extended formulations of CVRP are also relevant to those of LSP. Gouveia (1995) studied the projection of single-commodity flow formulations of CVRP. As a result of the projection, they derived valid inequalities called multi-star inequalities. These results were generalized by Letchford, Eglese, et al. (2002) and Letchford & Salazar-González (2006). Letchford, Eglese, et al. (2002) introduced generalized multi-star inequalities and re-

ported their computational effects. Letchford & Salazar-González (2006) conducted a survey of various formulations of CVRP and inequalities derived from the projection of the formulations and analyzed the relations between them. Later, Letchford & Salazar-González (2015) provided stronger formulations. These results are relevant to our problem, as discussed in Section 2.4. However, in contrast to CVRP, the single-period LSP should make additional decisions regarding the production amount. In this respect, the results of CVRP are not sufficient for LSP.

Guimarães et al. (2014) reviewed various formulations of LSP and proposed classification criteria. There are various formulations, such as GSEC-based formulation with exponentially many constraints, pattern-based formulations (Guimarães et al., 2013) with exponentially many variables, and single/multi-commodity flow formulations. They conducted extensive computational experiments to compare their computational performance and reported that, on average, the single-commodity flow formulation showed the best performance. However, the theoretical strengths of the formulations and their relationships were not investigated.

2.3 Single-period Substructure

The single-period substructure of LSP is represented as follows. Because we consider only a single period, we omit period index t.

$$\text{minimize} \quad \sum_{i \in \mathcal{I}} \left(hc_i s_i^+ + bc_i s_{it}^- + pc_i x_i \right) + \sum_{(i,j) \in \mathcal{A}} sc_{ij} z_{ij} \tag{2.3a}$$

$$\text{subject to} \quad s_i^- + x_i = d_i + s_i^+ \qquad \forall i \in \mathcal{I} \tag{2.3b}$$

$$\sum_{i \in \mathcal{I}} a_i x_i + \sum_{(i,j) \in \mathcal{A}} st_{ij} z_{ij} \leq K \tag{2.3c}$$

$$x_i \leq u_i y_i \qquad \forall i \in \mathcal{I} \quad (2.3d)$$

$$\sum_{i \in \mathcal{I}} z_{0i} = 1 \quad (2.3e)$$

$$\sum_{j \in \mathcal{I}_0 \setminus \{i\}} z_{ji} = \sum_{j \in \mathcal{I}_0 \setminus \{i\}} z_{ij} = y_i \qquad \forall i \in \mathcal{I} \quad (2.3f)$$

$$\sum_{(i,j) \in \delta^+(S)} z_{ij} \geq y_k \qquad \forall k \in S, S \subseteq \mathcal{I} \quad (2.3g)$$

$$x_i, s_i^-, s_i^+ \geq 0,\ y_i \in \{0,1\} \qquad \forall i \in \mathcal{I} \quad (2.3h)$$

$$z_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}_0 \quad (2.3i)$$

Let $\mathcal{X} = \{(\mathbf{x}, \mathbf{s}^+, \mathbf{s}^-, \mathbf{y}, \mathbf{z}) \in \mathbb{R}_+^{3I} \times \mathbb{B}^{I^2+2I} : \text{ satisfies constraints (2.3b)} - \text{(2.3i)}\}$. Unlike the multi-period problem (2.1), we introduce the variables s_i^- and s_i^+ which represent the shortage and surplus for the demand of item i, respectively. Owing to these variables, demand constraints (2.3b) can always be satisfied, and the solution set is not empty. In addition, to enrich the analysis, we introduce the parameter u_i, the upper bound on the production amount of item $i \in \mathcal{I}$. The individual upper bounds can be dropped by letting $u_i = K$, $\forall i \in \mathcal{I}$. We denote $\mathcal{X}_0$ as the solution set (2.3) with $u_i = K$, $\forall i \in \mathcal{I}$. We use `GSEC`s (2.3g) to prevent invalid cycles.

2.3.1 Assumptions

Before presenting basic properties of the single-period substructure, we make some assumptions. Firstly, without loss of generality, we let $a_i = 1$, $\forall i \in \mathcal{I}$. $\mathcal{X}$ can be transformed with general a_i into an equivalent set with $a_i = 1$, by considering variables $x_i' = a_i x_i$, $s_i^{-\prime} = a_i s_i^-$, $s_i^{+\prime} = a_i s_i^+$, and the modified coefficients $d_i' = a_i d_i$ and $u_i' = a_i u_i$. We also assume that $0 < st_{ij} \leq K$, $\forall (i,j) \in \mathcal{A}$ and $u_i \leq K$, $\forall i \in \mathcal{I}$.

2.3.2 Basic Polyhedral Properties

We analyze the basic polyhedral structure of the convex hull of $\mathcal{X}$, that is, $conv(\mathcal{X})$. We note that, for the proofs in this section, we only exhibit the values of some variables which are relevant for the sake of simplicity. The variables $\mathbf{x}, \mathbf{y}$, and $\mathbf{z}$ are assumed to be zero unless otherwise mentioned. The values of $\mathbf{s}^+$ and $\mathbf{s}^-$ are automatically set with respect to the corresponding $\mathbf{x}$ values and constraints (2.3b). We start by presenting the dimension of $conv(\mathcal{X})$.

Proposition 2.1. *Dimension of* $conv(\mathcal{X}) = I^2 + 2I - 1$.

Proof. The number of the total variables is $I^2 + 5I$. As there are $3I + 1$ equality constraints (2.3b) and (2.3e) – (2.3f) which are linearly independent, $dim(conv(\mathcal{X})) \leq I^2 + 2I - 1$. Therefore, it is sufficient to find $I^2 + 2I$ linearly independent points. Because there are I extreme rays of form $s_i^+ = s_i^- = 1, \forall i \in \mathcal{I}$, it is sufficient to find $I^2 + I$ points.

- For each $i \in \mathcal{I}$, $z_{0i} = y_i = z_{i0} = 1$. ($I$ points)
- For each $i \in \mathcal{I}$, $z_{0i} = y_i = z_{i0} = 1$ and $x_i = u_i$. (I points)
- For each $(i, j) \in \mathcal{A}$, $z_{0i} = y_i = z_{ij} = y_j = z_{j0} = 1$, $x_i = \min\{u_i, (K - st_{ij})/2\}$, and $x_j = \min\{u_j, (K - st_{ij})/2\}$. ($I^2 - I$ points)

It is obvious that the above $I^2 + I$ points are linearly independent. □

We use the following Lemma 2.2 from Nemhauser & Wolsey (1988) to prove other propositions.

Lemma 2.2 (Nemhauser & Wolsey (1988)). *Given a polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, let $(A^=, b^=)$ be the equality set of $P \subseteq \mathbb{R}^n$ and $M^=$ be the corresponding constraint index set. Also, let $F = \{x \in P : \pi x = \pi_0\}$ be a proper face of P. The following two statements are equivalent:*

1. *F is a facet of P.*
2. *If $\lambda x = \lambda_0$ for all $x \in F$, then*

$$(\lambda, \lambda_0) = (u\pi + vA^=, u\pi_0 + vb^=) \text{ for some } u \in \mathbb{R}^1 \text{ and some } v \in \mathbb{R}^{|M^=|}.$$

Proposition 2.3. *Constraint* (2.3c) *is a facet-defining inequality of $conv(\mathcal{X})$ if $u_i = K$ for all $i \in \mathcal{I}$, that is, it defines a facet of $conv(\mathcal{X}_0)$.*

Proof. Let us consider a hyperplane

$$\sum_{i \in \mathcal{I}} (\alpha_i x_i + \beta_i^+ s_i^+ + \beta_i^- s_i^- + \gamma_i y_i) + \sum_{(i,j) \in \mathcal{A}_0} \delta_{ij} z_{ij} = \pi_0$$

which contains all points $(\mathbf{x}, \mathbf{s}^+, \mathbf{s}^-, \mathbf{y}, \mathbf{z})$ in the face defined by constraint (2.3c). We show that $(\boldsymbol{\alpha}, \boldsymbol{\beta}^+, \boldsymbol{\beta}^-, \boldsymbol{\gamma}, \boldsymbol{\delta})$ is the sum of a scalar multiple of the coefficients in constraint (2.3c) and the equality system.

Firstly, without loss of generality, we can let $\gamma_i = 0$ for all $i \in \mathcal{I}$ because y_i can be replaced with $\sum_{j \in \mathcal{I}_0} z_{ij}$. Moreover, as there are I extreme rays of form $s_i^+ = s_i^- = 1$, we can let $\beta_i^+ + \beta_i^- = 0$. Due to constraints (2.3b), $\beta_i^+ s_i^+ + \beta_i^- s_i^- = \beta_i^+ (s_i^+ - s_i^-) = \beta_i^+ (d_i - x_i)$. Therefore, we also can let $\beta_i^+ = \beta_i^- = 0$ for all $i \in \mathcal{I}$ without loss of generality. Now, consider the following points which satisfy constraint (2.3c) at equality:

- For $i \in \mathcal{I}$, let $z_{0i} = y_i = z_{i0} = 1$ and $x_i = K$. Then, $\alpha_i K + \delta_{0i} + \delta_{i0} = \pi_0$. Consequently, $\delta_{0i} = \pi_0 - \alpha_i K - \delta_{i0}$.
- For $(i,j) \in \mathcal{A}$, let $z_{0i} = y_i = z_{ij} = y_j = z_{j0} = 1$. Then, $x_i + x_j = K - st_{ij}$ and the following two cases are possible.

  1. $(x_i, x_j) = (0, K - st_{ij})$, then $\alpha_j(K - st_{ij}) + \delta_{0i} + \delta_{ij} + \delta_{j0} = \pi_0$.
  2. $(x_i, x_j) = (K - st_{ij}, 0)$, then $\alpha_i(K - st_{ij}) + \delta_{0i} + \delta_{ij} + \delta_{j0} = \pi_0$.

  From the above cases, $\alpha_i = \alpha_j = \alpha^*$ and

$$\begin{aligned}
\delta_{ij} &= \pi_0 - \delta_{0i} - \delta_{j0} - \alpha^*(K - st_{ij}) \\
&= \pi_0 - (\pi_0 - \alpha^* K - \delta_{i0}) - \delta_{j0} - \alpha^*(K - st_{ij}) \\
&= \delta_{i0} - \delta_{j0} + \alpha^* st_{ij}.
\end{aligned}$$

Therefore, the following relations hold between the equations:

$$\begin{aligned}
& \sum_{i\in\mathcal{I}}(\alpha_i x_i + \beta_i^+ s_i^+ + \beta_i^- s_i^- + \gamma_i y_i) + \sum_{(i,j)\in\mathcal{A}_0} \delta_{ij} z_{ij} = \pi_0 \\
(\Leftrightarrow) \quad & \sum_{i\in\mathcal{I}} \alpha^* x_i + \sum_{i\in\mathcal{I}} \Big(\delta_{i0} z_{i0} + \delta_{0i} z_{0i}\Big) + \sum_{(i,j)\in\mathcal{A}} \delta_{ij} z_{ij} = \pi_0 \\
(\Leftrightarrow) \quad & \sum_{i\in\mathcal{I}} \alpha^* x_i + \sum_{i\in\mathcal{I}} \Big(\delta_{i0} z_{i0} + (\pi_0 - \alpha^* K - \delta_{i0}) z_{0i}\Big) \\
& \qquad + \sum_{(i,j)\in\mathcal{A}} (\delta_{i0} - \delta_{j0} + \alpha^* st_{ij}) z_{ij} = \pi_0 \\
(\Leftrightarrow) \quad & \alpha^*\Big(\sum_{i\in\mathcal{I}} \big(x_i - K z_{0i}\big) + \sum_{(i,j)\in\mathcal{A}} st_{ij} z_{ij}\Big) \\
& \qquad + \sum_{i\in\mathcal{I}} \delta_{i0}\Big(z_{i0} - z_{0i} + \sum_{j\in\mathcal{I}\setminus\{i\}} \big(z_{ij} - z_{ji}\big)\Big) + \pi_0\big(\sum_{i\in\mathcal{I}} z_{0i}\big) = \pi_0
\end{aligned}$$

$$(\Leftrightarrow) \qquad \alpha^*\Big(\sum_{i\in\mathcal{I}} x_i + \sum_{(i,j)\in\mathcal{A}} st_{ij}z_{ij} - K\Big) + \sum_{i\in\mathcal{I}} \delta_{i0}\Big(\sum_{j\in\mathcal{I}_0\setminus\{i\}} z_{ij} - \sum_{j\in\mathcal{I}_0\setminus\{i\}} z_{ji}\Big) + \pi_0\Big(\sum_{i\in\mathcal{I}} z_{0i} - 1\Big) = 0$$

In the last equation, the first term is a scalar multiplication of the constraint (2.3c), whereas the second and third terms are scalar multiplications of the equality set. Therefore, from Lemma 2.2, it is shown that constraint (2.3c) is a facet-defining inequality if $u_i = K$ for all $i \in \mathcal{I}$. □

Proposition 2.4. *For a given $i \in \mathcal{I}$, if $u_i \leq K - \max\{st_{ij}, st_{ji}\}$ for all $j \in \mathcal{I} \setminus \{i\}$, constraint* (2.3d) *defines a facet of conv$(\mathcal{X})$.*

Proof. For a given $i \in \mathcal{I}$, consider the following points:

- $z_{0i} = y_i = z_{i0} = 1$ and $x_i = u_i$. (1 point)
- For each $j \in \mathcal{I} \setminus \{i\}$, $z_{0j} = y_j = z_{j0} = 1$ and $x_j = 0$. ($I - 1$ points)
- For each $j \in \mathcal{I} \setminus \{i\}$, $z_{0j} = y_j = z_{j0} = 1$ and $x_j = u_j$. ($I - 1$ points)
- For each $j \in \mathcal{I} \setminus \{i\}$, $z_{0i} = y_i = z_{ij} = y_j = z_{j0} = 1$, $x_i = u_i$, and $x_j = \min\{K - st_{ij} - u_i, u_j\}$. ($I - 1$ points)
- For each $j \in \mathcal{I} \setminus \{i\}$, $z_{0j} = y_j = z_{ji} = y_i = z_{i0} = 1$, $x_i = u_i$, and $x_j = \min\{K - st_{ji} - u_i, u_j\}$. ($I - 1$ points)
- For each $(j, k) \in \mathcal{A}$ such that $j \neq i$ and $k \neq i$, $z_{0j} = y_j = z_{jk} = y_k = z_{k0} = 1$, $x_j = \min\{u_j, (K - st_{jk})/2\}$, and $x_k = \min\{u_k, (K - st_{jk})/2\}$. ($(I-1)(I-2)$ points)

There are total $I^2 + I - 1$ linearly independent points. Together with I linearly independent extreme lays of form $s_i^+ = s_i^- = 1$ for all $i \in \mathcal{I}$, it is demonstrated that

constraint (2.3d) is a facet defining inequality of $conv(\mathcal{X})$. □

When the condition given in Proposition 2.4 does not hold, even when $u_i = K$ for all $i \in \mathcal{I}$, constraint (2.3d) does not define the facets of $conv(\mathcal{X}_0)$ in general. Instead, in this case, constraint (2.3d) can be tightened to be the facet-defining inequality of $conv(\mathcal{X}_0)$, as shown in Proposition 2.5.

Proposition 2.5. *For a given $i \in \mathcal{I}$, the following tightened upper bound constraint* (2.4) *is a facet-defining inequality of $conv(\mathcal{X})$ when $u_i = K$, that is, it defines a facet of $conv(\mathcal{X}_0)$.*

$$x_i \leq Ky_i - \sum_{j \in \mathcal{I} \setminus \{i\}} (st_{ij} z_{ij} + st_{ji} z_{ji}) \qquad \forall i \in \mathcal{I} \tag{2.4}$$

Proof. For a given $i \in \mathcal{I}$, consider the following points:

- $z_{0i} = y_i = z_{i0} = 1$ and $x_i = K$. (1 point)
- For each $j \in \mathcal{I} \setminus \{i\}$, $z_{0j} = y_j = z_{j0} = 1$ and $x_j = 0$. ($I - 1$ points)
- For each $j \in \mathcal{I} \setminus \{i\}$, $z_{0j} = y_j = z_{j0} = 1$ and $x_j = u_j$. ($I - 1$ points)
- For each $j \in \mathcal{I} \setminus \{i\}$, $z_{0i} = y_i = z_{ij} = y_j = z_{j0} = 1$ and $x_i = K - st_{ij}$. ($I - 1$ points)
- For each $j \in \mathcal{I} \setminus \{i\}$, $z_{0j} = y_j = z_{ji} = y_i = z_{i0} = 1$ and $x_i = K - st_{ji}$. ($I - 1$ points)
- For each $(j, k) \in \mathcal{A}$ such that $j \neq i$ and $k \neq i$, $z_{0j} = y_j = z_{jk} = y_k = z_{k0} = 1$, $x_j = \min\{u_j, (K - st_{jk})/2\}$ and $x_k = \min\{u_k, (K - st_{jk})/2\}$. ($(I-1)(I-2)$ points)

There are total $I^2 + I - 1$ linearly independent points. Together with I linearly independent extreme lays of form $s_i^+ = s_i^- = 1$ for all $i \in \mathcal{I}$, it is demonstrated that constraint (2.4) defines a facet of $conv(\mathcal{X}_0)$. □

2.4 New Valid Inequalities

2.4.1 S-STAR Inequality

In this section, we propose new families of valid inequalities for $conv(\mathcal{X})$ and identify their facet-defining conditions. The first is called the S-STAR inequality because it forms the shape of a star centered on a given set of nodes $S \subseteq \mathcal{I}$, as illustrated in Figure 2.2. The inequality can also eliminate all invalid cycles which do not contain item 0. This indicates that the S-STAR inequalities can replace GSEC (2.3g).

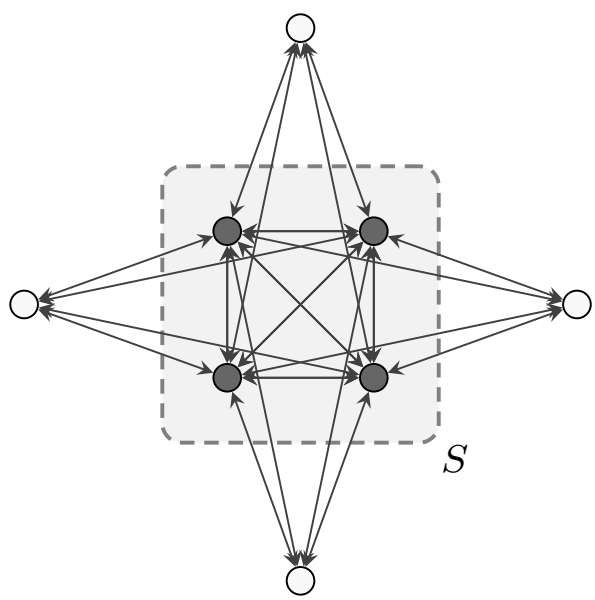


Figure 2.2: Illustration of the S-STAR inequality

Proposition 2.6. *For a given node subset $S \subseteq \mathcal{I}$, the following inequality is valid for $\mathcal{X}$:*

$$\sum_{i \in S} x_i + \sum_{(i,j) \in \delta(S)} st_{ij} z_{ij} + \sum_{(i,j) \in E(S)} st_{ij} z_{ij} \leq K \sum_{(i,j) \in \delta^-(S)} z_{ij}. \tag{2.5}$$

In addition, these inequalities are sufficient to eliminate all cycles which do not include item 0.

Proof. For a given feasible solution of $\mathcal{X}$, if $\sum_{(i,j) \in \delta^-(S)} z_{ij} \geq 1$, the inequality trivially holds because the right-hand side becomes greater than or equal to the capacity K. On the other hand, if $\sum_{(i,j) \in \delta^-(S)} z_{ij} = 0$, there are no incoming arcs to the nodes in S which indicates that the items in S cannot be produced, and the corresponding setups that start or end with the items in S cannot be conducted. Therefore, the left-hand side value also becomes zero and the inequality holds.

To demonstrate that the `S-STAR` inequality can prevent any invalid cycles, suppose that we are given a cycle that does not include item 0, and let N be the set of nodes in the cycle ($|N| \geq 2$). Then, from the definition, there are no incoming and outgoing arcs for N. Therefore, $\sum_{(i,j) \in \delta^-(N)} z_{ij} = 0$, and the right-hand side of inequality (2.5) defined by N becomes zero. On the other hand, as $\sum_{(i,j) \in E(N)} z_{ij} = |N|$ and $st_{ij} > 0$, $\forall (i,j) \in \mathcal{A}$, the left-hand side is greater than 0 which indicates that the given solution violates the `S-STAR` inequality. This demonstrates that invalid cycles can be eliminated by adding the `S-STAR` inequalities. □

We note that the `S-STAR` inequality is closely related to the *generalized large multistar* (`GLM`) inequality proposed for CVRP by Gouveia (1995) and Letchford, Eglese, et al. (2002). To demonstrate this more explicitly, consider the following generic inequality which subsumes both `S-STAR` and `GLM` inequalities, although it

has nonlinear terms $x_i z_{ij}$.

$$\sum_{i \in S} x_i + \sum_{(i,j) \in \delta^+(S)} (st_{ij} + x_j) z_{ij} + \sum_{(i,j) \in \delta^-(S)} (x_i + st_{ij}) z_{ij} + \sum_{(i,j) \in E(S)} st_{ij} z_{ij} \leq K \sum_{(i,j) \in \delta^-(S)} z_{ij} \qquad \forall S \subseteq \mathcal{I} \quad (2.6)$$

In CVRP, the value of x_i which indicates the delivery amount for customer i, cannot have any value between 0 and u_i, but must be either 0 or d_i, that is, $x_i = d_i y_i$. In addition, no setup time is considered in CVRP. In this regard, by setting $st_{ij} = 0$ and $x_i = d_i y_i$ for inequality (2.6), we can obtain the `GLM` inequality immediately (note that $y_i z_{ij}$ can be linearized as z_{ij}). On the other hand, by dropping variables x_j and x_i in the second and third terms of inequality (2.6), respectively, we can obtain the `S-STAR` inequality.

In contrast to the `GLM` inequalities, the facet-defining conditions of the `S-STAR` inequalities can be identified. Specifically, if the individual upper bound for each item is not imposed, the `S-STAR` inequality becomes a facet-defining inequality of $conv(\mathcal{X})$.

Proposition 2.7. *Given $S \subseteq \mathcal{I}$, the `S-STAR` inequality defines a facet of $conv(\mathcal{X})$ when $u_i = K$ for all $i \in \mathcal{I}$. In other words, it defines a facet of $conv(\mathcal{X}_0)$.*

Proof. Let us consider a hyperplane $\sum_{i \in \mathcal{I}} \alpha_i x_i + \sum_{(i,j) \in \mathcal{A}_0} \delta_{ij} z_{ij} = \pi_0$ which contains all points $(\mathbf{x}, \mathbf{s}^+, \mathbf{s}^-, \mathbf{y}, \mathbf{z})$ in the face defined by inequality (2.5), given $S \subseteq \mathcal{I}$. As shown in the proof of Proposition 2.3, we can assume that the coefficients of $\mathbf{s}^+$, $\mathbf{s}^-$, and $\mathbf{y}$ is zero. Consider the following points which satisfy inequality (2.5) at equality, given S:

- For each $i \in S$, $z_{0i} = y_i = z_{i0} = 1$. Then, $x_i = K$ and $\alpha_i K + \delta_{0i} + \delta_{i0} = \pi_0$, that is,

$$\delta_{i0} = \pi_0 - \delta_{0i} - \alpha_i K \qquad \forall i \in S.$$

- For each $i \in \mathcal{I} \setminus S$, $z_{0i} = y_i = z_{i0} = 1$. Then, x_i can have any value between 0 and K which indicates $\alpha_i = 0$ and

$$\delta_{i0} = \pi_0 - \delta_{0i} \qquad \forall i \in \mathcal{I} \setminus S.$$

- For each $(i, j) \in E(S)$, $z_{0i} = y_i = z_{ij} = y_j = z_{j0} = 1$. In this case, x_i and x_j can have any values satisfying $x_i + x_j = K - st_{ij}$. There exist many combinations satisfying $x_i + x_j = K - st_{ij}$. Therefore, $\alpha_i = \alpha_j = \alpha^*$ and

$$\delta_{ij} = \pi_0 - \alpha^*(K - st_{ij}) - \delta_{0i} - \delta_{j0} = \alpha^* st_{ij} - \delta_{0i} + \delta_{0j} \qquad \forall (i, j) \in E(S).$$

- For each $(i, j) \in E(S : \mathcal{I} \setminus S)$, $z_{0i} = y_i = z_{ij} = y_j = z_{j0} = 1$ and $x_i = K - st_{ij}$. Therefore, $\alpha^*(K - st_{ij}) + \delta_{0i} + \delta_{ij} + \delta_{j0} = \pi_0$, that is,

$$\delta_{ij} = \alpha^*(st_{ij} - K) - \delta_{0i} + \delta_{0j} \qquad \forall (i, j) \in E(S : \mathcal{I} \setminus S).$$

- For each $(i, j) \in E(\mathcal{I} \setminus S : S)$, $z_{0i} = y_i = z_{ij} = y_j = z_{j0} = 1$ and $x_j = K - st_{ij}$. Therefore, $\alpha^*(K - st_{ij}) + \delta_{0i} + \delta_{ij} + \delta_{j0} = \pi_0$, that is,

$$\delta_{ij} = \alpha^* st_{ij} - \delta_{0i} + \delta_{0j} \qquad \forall (i, j) \in E(\mathcal{I} \setminus S : S).$$

- For each $(i,j) \in E(\mathcal{I} \setminus S)$, $z_{0i} = y_i = z_{ij} = y_j = z_{j0} = 1$. In this case,

$$\delta_{ij} = \pi_0 - \delta_{0i} - \delta_{j0} = -\delta_{0i} + \delta_{0j} \qquad \forall (i,j) \in E(\mathcal{I} \setminus S).$$

The term $\sum_{(i,j)\in\mathcal{A}_0} \delta_{ij} z_{ij}$ can be decomposed as

$$\underbrace{\sum_{i\in\mathcal{I}} (\delta_{i0} z_{i0} + \delta_{0i} z_{0i})}_{(i)} + \underbrace{\sum_{(i,j)\in E(S)} \delta_{ij} z_{ij}}_{(ii)} +$$
$$\underbrace{\sum_{(i,j)\in E(S:\mathcal{I}\setminus S)} \delta_{ij} z_{ij}}_{(iii)} + \underbrace{\sum_{(i,j)\in E(\mathcal{I}\setminus S:S)} \delta_{ij} z_{ij}}_{(iv)} + \underbrace{\sum_{(i,j)\in E(\mathcal{I}\setminus S)} \delta_{ij} z_{ij}}_{(v)}.$$

The decomposed terms (i) to (v) can be stated as follows:

$$\begin{aligned}
(i) \quad &= \sum_{i\in S} (\delta_{i0} z_{i0} + \delta_{0i} z_{0i}) + \sum_{i\in\mathcal{I}\setminus S} (\delta_{i0} z_{i0} + \delta_{0i} z_{0i}) \\
&= \sum_{i\in S} \big((\pi_0 - \delta_{0i} - \alpha^* K) z_{i0} + \delta_{0i} z_{0i}\big) + \sum_{i\in\mathcal{I}\setminus S} \big((\pi_0 - \delta_{0i}) z_{i0} + \delta_{0i} z_{0i}\big) \\
&= \pi_0 \sum_{i\in\mathcal{I}} z_{i0} + \sum_{i\in\mathcal{I}} \delta_{0i}(z_{0i} - z_{i0}) - \alpha^* K \sum_{i\in S} z_{i0}
\end{aligned}$$

$$\begin{aligned}
(ii) \quad &= \sum_{(i,j)\in E(S)} \big(\alpha^* st_{ij} - \delta_{0i} + \delta_{0j}\big) z_{ij} \\
&= \alpha^* \sum_{(i,j)\in E(S)} st_{ij} z_{ij} + \sum_{i\in S} \delta_{0i} \Big(\sum_{j\in S} z_{ji} - \sum_{j\in S} z_{ij} \Big)
\end{aligned}$$

$$(iii) \quad = \sum_{(i,j)\in E(S:\mathcal{I}\setminus S)} \big(\alpha^* st_{ij} - \alpha^* K - \delta_{0i} + \delta_{0j}\big) z_{ij}$$

$$=\alpha^* \sum_{(i,j)\in E(S:\mathcal{I}\setminus S)} (st_{ij}-K)z_{ij} + \sum_{j\in\mathcal{I}\setminus S}\delta_{0j}\Big(\sum_{i\in S} z_{ij}\Big) - \sum_{i\in S}\delta_{0i}\Big(\sum_{j\in\mathcal{I}\setminus S} z_{ij}\Big)$$

$$(iv) \quad = \sum_{(i,j)\in E(\mathcal{I}\setminus S:S)} \big(\alpha^* st_{ij} - \delta_{0i} + \delta_{0j}\big) z_{ij}$$

$$=\alpha^* \sum_{(i,j)\in E(\mathcal{I}\setminus S:S)} st_{ij}z_{ij} + \sum_{j\in S}\delta_{0j}\Big(\sum_{i\in\mathcal{I}\setminus S} z_{ij}\Big) - \sum_{i\in\mathcal{I}\setminus S}\delta_{0i}\Big(\sum_{j\in S} z_{ij}\Big)$$

$$(v) \quad = \sum_{(i,j)\in E(\mathcal{I}\setminus S)} (-\delta_{0i}+\delta_{0j})z_{ij} = \sum_{i\in\mathcal{I}\setminus S}\delta_{0i}\Big(\sum_{j\in\mathcal{I}\setminus S} z_{ji} - \sum_{j\in\mathcal{I}\setminus S} z_{ij}\Big)$$

Therefore, $\sum_{i\in\mathcal{I}}\alpha_i x_i + \sum_{(i,j)\in\mathcal{A}_0}\delta_{ij}z_{ij} = \pi_0$ is reduced to

$$\alpha^*\Big(\sum_{i\in S} x_i + \sum_{(i,j)\in\delta(S)} st_{ij}z_{ij} + \sum_{(i,j)\in E(S)} st_{ij}z_{ij} - K\sum_{(i,j)\in\delta^+(S)} z_{ij}\Big) +$$
$$\pi_0\Big(\sum_{i\in\mathcal{I}} z_{i0} - 1\Big) + \sum_{i\in\mathcal{I}}\delta_{0i}\Big(\sum_{j\in\mathcal{I}_0} z_{ji} - \sum_{j\in\mathcal{I}_0} z_{ij}\Big) = 0.$$

Because the second and third terms are scalar multiplications of the equality set and the first term is that of the inequality (2.5), it defines a facet of $conv(\mathcal{X}_0)$. □

2.4.2 Separation of S-STAR Inequality

Given a fractional solution $(\bar{\mathbf{x}}, \bar{\mathbf{s}}^+, \bar{\mathbf{s}}^-, \bar{\mathbf{y}}, \bar{\mathbf{z}})$, the separation problem of the S-STAR inequality is to find a subset $N \subseteq \mathcal{I}$ such that

$$\sum_{i\in N}\bar{x}_i + \sum_{(j,i)\in E(N)} st_{ji}\bar{z}_{ji} + \sum_{(i,j)\in\delta(N)} st_{ij}\bar{z}_{ij} > K\sum_{(i,j)\in\delta^+(N)}\bar{z}_{ij}$$

or equivalently,

$$\sum_{i \in N} \Big(\bar{x}_i + \sum_{j \in \mathcal{I}_0 \setminus \{i\}} st_{ji} \bar{z}_{ji} \Big) + \sum_{i \in N} \sum_{j \notin N} \big(st_{ij} - K \big) \bar{z}_{ij} > 0.$$

To formulate the above separation problem as an integer program, we define the following notations and variables.

- For $i \in \mathcal{I}$, let $\alpha_i := \bar{x}_i + \sum_{j \in \mathcal{I}_0 \setminus \{i\}} st_{ij} \bar{z}_{ij}$ which is non-negative.
- For $(i,j) \in \mathcal{A}_0$, $\beta_{ij} := \big(K - st_{ij} \big) \bar{z}_{ij}$ which is non-negative.
- For $i \in \mathcal{I}$, let binary variable $p_i = 1$ if $i \in N$. Let $p_0 = 0$.
- For $(i,j) \in \mathcal{A}_0$, let binary variable $q_{ij} = 1$ if $i \in N$ and $j \notin N$.

The separation problem of the `S-STAR` inequalities can be formulated as follows.

$$\begin{aligned}
\texttt{(SEP-S-STAR)} \quad \text{maximize} \quad & \sum_{i \in \mathcal{I}} \alpha_i p_i - \sum_{(i,j) \in \mathcal{A}_0} \beta_{ij} q_{ij} && && \text{(2.7a)} \\
\text{subject to} \quad & q_{ij} \leq p_i && \forall (i,j) \in \mathcal{A}_0 && \text{(2.7b)} \\
& q_{ij} \leq 1 - p_j && \forall (i,j) \in \mathcal{A}_0 && \text{(2.7c)} \\
& p_i - p_j \leq q_{ij} && \forall (i,j) \in \mathcal{A}_0 && \text{(2.7d)} \\
& p_i \in \{0,1\} && \forall i \in \mathcal{I} && \text{(2.7e)} \\
& q_{ij} \in \{0,1\} && \forall (i,j) \in \mathcal{A}_0 && \text{(2.7f)}
\end{aligned}$$

Proposition 2.8. *`(SEP-S-STAR)` is polynomially solvable.*

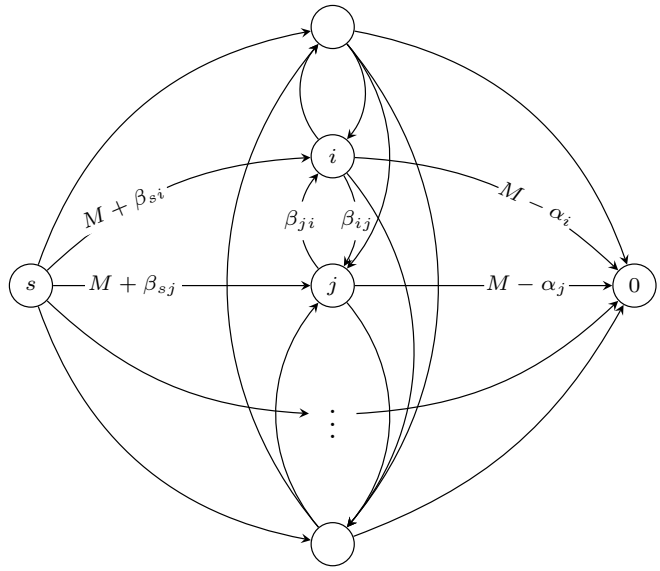
Proof. The objective coefficients of the variables q_{ij} are non-positive. Therefore, there exists an optimal solution with q_{ij} value as small as possible. In this regard,

the value of q_{ij} is determined by constraint (2.7d), that is, $q_{ij} = p_i - p_j$. Because $p_i - p_j \leq p_i$ and $p_i - p_j \leq 1 - p_j$ always hold, constraints (2.7b) and (2.7c) are automatically satisfied and can be dropped. Then, a matrix corresponding to the remaining constraints (2.7d) is of the form $[I|Q]$, where I is an identity matrix and each row of matrix Q contains only two nonzero coefficients of 1 and -1 which indicates that it is totally unimodular (Nemhauser & Wolsey, 1988). Therefore, the problem can be solved by solving its LP relaxation which can be performed in polynomial time. □

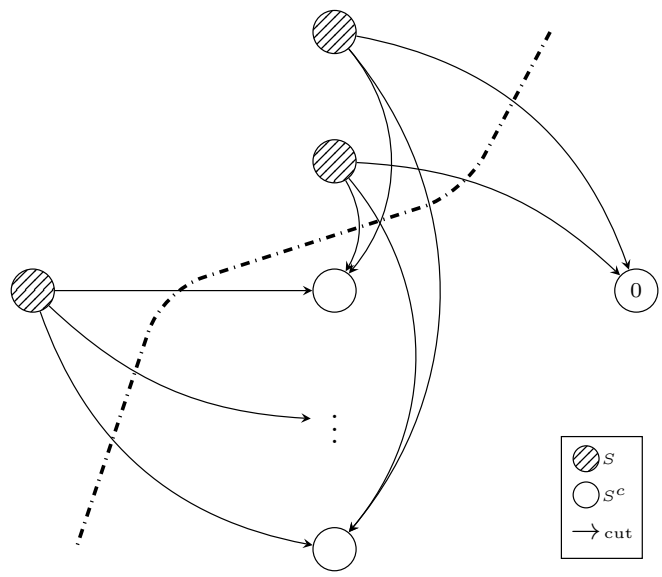
Separation of `S-STAR` inequality can be conducted by solving minimum cut problems (or equivalently maximum flow problems) rather than solving `(SEP-S-STAR)`. Given a fraction solution obtained after solving LP relaxation (2.3), one can compute $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ which are defined above. By letting an item $s \in \mathcal{I}$ be the source node, the corresponding capacitated network $\mathcal{G}^s = (\mathcal{I}_0, \mathcal{A}_0, \mathbf{C})$ on which the minimum $(s, 0)$ cut problem is defined can be constructed by setting arc capacity C_{ij} as follows:

- $C_{si} = M + \beta_{si}$ for all $i \in \mathcal{I} \setminus \{s\}$,
- $C_{i0} = M - \alpha_i$ for all $i \in \mathcal{I} \setminus \{s\}$,
- $C_{ij} = \beta_{ij}$ for all $(i, j) \in \mathcal{A}$ such that $i \neq s$,
- $C_{s0} = 0$.

The maximum value of α_i, denoted by $M = \max_{i \in \mathcal{I}}\{\alpha_i\}$, is added to arcs (s, i) and $(i, 0)$ for all $i \in \mathcal{I} \setminus \{s\}$ to prevent negative arc capacity. Note that among these $2(I-1)$ arcs with additionally added M, exactly $I-1$ arcs are always selected when we find minimum $(s, 0)$ cut. The resulting graph is illustrated in Figure 2.3(a).



(a) Construction of the network with a source node s



(b) A cut resulting from the maximum flow problem

Figure 2.3: Illustration of the separation procedure of `S-STAR` inequality

Algorithm 2.1: Separation procedure of `S-STAR` inequality

$\mathcal{C} = \emptyset$; `// storage of violated cut information`
$(\boldsymbol{\alpha}, \boldsymbol{\gamma}, M) \leftarrow Calc(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}})$;
forall $s \in \mathcal{I}$ **do**
  if *s was not selected as a member of S in previous iteration* **then**
    construct $\mathcal{G}^s$ with $(\boldsymbol{\alpha}, \boldsymbol{\gamma}, M)$;
    $(obj, cut) \leftarrow maxFlow(\mathcal{G}^s, s, 0)$; `// get max flow value and cut`
    if $obj < (I-2)M + \alpha_s$ **then** `// violated cut found`
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{cut\}$;
    end
  end
end
return $\mathcal{C}$

Given the graph, minimum $(s, 0)$ cut can be found in $\mathcal{O}(I^3)$ by a maximum flow algorithm. Therefore, considering the iteration over all possible source nodes $s \in \mathcal{I}$, the overall separation routine takes $\mathcal{O}(I^4)$ time. When the capacity of the found minimum $(s, 0)$ cut, denoted as $(S : S^C)$ as illustrated in 2.3(b), is smaller than $(I-2)M + \alpha_s$, the `S-STAR` inequality defined by the set S is violated, and therefore, added to (2.3). Otherwise, there is no violated `S-STAR` inequality which is defined by a subset of nodes containing an item s.

In our implementation of the separation algorithm, we choose the source node $s \in \mathcal{I}$ in increasing order of the index. To speed up the separation procedure, we prevented nodes which were selected as a member of S in the previous minimum cut problem from becoming source nodes. From the practical point of view, various separation strategies can be used to further speed up the separation procedure which, however, is not the primary interest of this chapter and can be a future research direction. The overall separation procedure of `S-STAR` inequality, given a fractional solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}})$, is described in Algorithm 2.1.

2.4.3 U-STAR Inequality

The S-STAR inequality presented above defines a facet if there is no individual upper bound for the production amount of each item. On the other hand, the following valid inequality, denoted as U-STAR inequality, can be beneficial when the non-trivial upper bound for each item is presented.

Proposition 2.9. *For a given $S \subseteq \mathcal{I}$, the inequality*

$$\sum_{i \in S} x_i - \sum_{(i,j) \in E(S)} \lambda_{ij} z_{ij} \leq \sum_{(i,j) \in \delta^-(S)} u_j z_{ij} \tag{2.8}$$

is valid for $\mathcal{X}$, where $\lambda_{ij} := \min\{K - st_{ij} - u_i, u_j\}$ for all $(i,j) \in \mathcal{A}$.

Proof. Inequality (2.8) can be rewritten as

$$\sum_{i \in S} x_i \leq \sum_{i \in S} u_i y_i - \sum_{(i,j) \in E(S)} [u_i + st_{ij} + u_j - K]^+ z_{ij},$$

where $[a]^+ = \max\{0, a\}$. To show that this inequality is valid for $\mathcal{X}$, let us given a feasible solution $(\bar{\mathbf{x}}, \bar{\mathbf{s}}^+, \bar{\mathbf{s}}^-, \bar{\mathbf{y}}, \bar{\mathbf{z}})$. Then, $\bar{\mathbf{z}}$ forms a cycle $C = \{(i,j) : \bar{z}_{ij} = 1,\ (i,j) \in \mathcal{A}_0\}$ which includes node 0. The set of nodes included in C is denoted as $V(C)$.

If $u_i + st_{ij} + u_j \leq K$ for all $(i,j) \in E(S) \cap C$, then the inequality trivially holds because it is reduced to an aggregated upper bound constraint for $i \in S$. Therefore, let $R \subseteq E(S) \cap C$ such that $u_i + st_{ij} + u_j > K$ for $(i,j) \in R$, and let $R \neq \emptyset$. In addition, assume that R forms a path, that is, $R = \{(i_1, i_2), (i_2, i_3), \cdots, (i_{l-2}, i_{l-1}), (i_{l-1}, i_l)\}$. If not, R can be partitioned into several mutually exclusive paths. Applying the following logic to each of them separately is straightforward and the proof still

holds. Then, from the assumption, the following relations hold.

$$\begin{aligned}
&\sum_{i\in S} u_i\bar{y}_i - \sum_{(i,j)\in E(S)} [u_i + st_{ij} + u_j - K]^+\bar{z}_{ij} \\
&\quad = \sum_{i\in S\cap V(C)} u_i - \sum_{(i,j)\in R} (u_i + st_{ij} + u_j - K) \\
&\quad = \sum_{i\in S\cap V(C)} u_i - \sum_{k=1}^{l-1}(u_{i_k} + u_{i_{k+1}} + st_{i_k i_{k+1}} - K) \\
&\quad \geq (l-1)K - (u_{i_2} + \cdots u_{i_{l-1}}) - \sum_{k=1}^{l-1} st_{i_k i_{k+1}} \\
&\quad = K + (K - u_{i_2}) + \cdots + (K - u_{i_{l-1}}) - \sum_{k=1}^{l-1} st_{i_k i_{k+1}} \geq K - \sum_{k=1}^{l-1} st_{i_k i_{k+1}} \geq \sum_{i\in S} \bar{x}_i.
\end{aligned}$$

As a result, it is demonstrated that the `U-STAR` inequality is valid for $\mathcal{X}$. □

Proposition 2.10. *For a given $S \subseteq \mathcal{I}$, the `U-STAR` inequality* (2.8) *defines a facet of conv($\mathcal{X}$) if the following conditions hold.*

1. *For all $i \in S$ and $j \in \mathcal{I} \setminus S$, $u_i + \max\{st_{ij}, st_{ji}\} \leq K$.*
2. *For all $(i,j) \in E(S)$, $K - st_{ij} < u_i + u_j$.*

Proof. For a given $S \subseteq \mathcal{I}$, consider the following points satisfying constraint (2.8) at equality:

- For each $i \in S$, $z_{0i} = y_i = z_{i0} = 1$ and $x_i = u_i$. ($|S|$ points)
- For each $i \in \mathcal{I} \setminus S$, $z_{0i} = y_i = z_{i0} = 1$ and $x_i = 0$. ($I - |S|$ points)
- For each $i \in \mathcal{I} \setminus S$, $z_{0i} = y_i = z_{i0} = 1$ and $x_i = u_i$. ($I - |S|$ points)
- For each $(i,j) \in E(\mathcal{I} \setminus S)$, $z_{0i} = y_i = z_{ij} = y_j = z_{j0} = 1$ and $x_i = x_j = 0$. ($(I - |S|)(I - |S| - 1)$ points)

- For each $(i,j) \in E(S : \mathcal{I} \setminus S)$, $z_{0i} = y_i = z_{ij} = y_j = z_{j0} = 1$ and $x_i = u_i$. ($|S|(I - |S|)$ points)

- For each $(i,j) \in E(\mathcal{I} \setminus S : S)$, $z_{0i} = y_i = z_{ij} = y_j = z_{j0} = 1$ and $x_j = u_j$. ($|S|(I - |S|)$ points)

- For each $(i,j) \in E(S)$, let $z_{0i} = y_i = z_{ij} = y_j = z_{j0} = 1$. In this case, $x_i + x_j$ should be equal to $K - st_{ij}$. From the second condition of the proposition, $K - st_{ij} < u_i + u_j$. Therefore, one can let $(x_i, x_j) = (u_i, K - st_{ij} - u_i)$. ($|S|(|S| - 1)$ points)

- Choose one element from S, say i^*. Then, for each $j \in S \setminus \{i^*\}$, let $z_{0i^*} = y_{i^*} = z_{i^*j} = y_j = z_{j0} = 1$ and $(x_{i^*}, x_j) = (K - st_{i^*j} - u_j, u_j)$. ($|S| - 1$ points)

It is not hard to see that the above $I^2 + I - 1$ points are affinely independent. □

Note that when $u_i = K$, the first condition of the Proposition 2.10 is violated. Particularly, in this case, the `U-STAR` inequalities are reduced to the weakened version of the `S-STAR` inequalities and therefore, are dominated by them.

2.4.4 Separation of `U-STAR` Inequality

The separation problem of the `U-STAR` inequality for a given fractional solution $(\bar{\mathbf{x}}, \bar{\mathbf{s}}^+, \bar{\mathbf{s}}^-, \bar{\mathbf{y}}, \bar{\mathbf{z}})$, is to find a subset $N \subseteq \mathcal{I}$ such that

$$\sum_{i \in N} \bar{x}_i - \sum_{(i,j) \in E(N)} \min\{K - st_{ij} - u_i, u_j\} \bar{z}_{ij} > \sum_{(i,j) \in \delta^-(N)} u_j \bar{z}_{ij}.$$

Because one can rewrite $\min\{K - st_{ij} - u_i, u_j\} = u_j - [u_j + u_i + st_{ij} - K]^+$, where $[X]^+ = \max\{0, X\}$, the problem is reduced to finding N such that

$$\sum_{i \in N} (u_i \bar{y}_i - \bar{x}_i) - \sum_{(i,j) \in E(N)} [u_j + u_i + st_{ij} - K]^+ \bar{z}_{ij} < 0.$$

To formulate the above separation problem as an integer program, we define the following notations and variables.

- For $i \in \mathcal{I}$, let $\gamma_i := u_i \bar{y}_i - \bar{x}_i$ which is non-negative.
- For $(i,j) \in \mathcal{A}$, let $\delta_{ij} := [u_j + u_i + st_{ij} - K]^+ \bar{z}_{ij}$ which is non-negative.
- For $i \in \mathcal{I}$, let binary variable $p_i = 1$, if $i \in N$.
- For $(i,j) \in \mathcal{A}$, let binary variable $q_{ij} = 1$, if $i \in N$ and $j \in N$.

Using the notation, the separation problem can be formulated as follows.

$$\begin{aligned}
\texttt{(SEP-U-STAR)} \quad \text{minimize} \quad & \sum_{i \in \mathcal{I}} \gamma_i p_i - \sum_{(i,j) \in \mathcal{A}} \delta_{ij} q_{ij} && && \text{(2.9a)} \\
\text{subject to} \quad & q_{ij} \leq p_i && \forall (i,j) \in \mathcal{A} && \text{(2.9b)} \\
& q_{ij} \leq p_j && \forall (i,j) \in \mathcal{A} && \text{(2.9c)} \\
& p_i + p_j - 1 \leq q_{ij} && \forall (i,j) \in \mathcal{A} && \text{(2.9d)} \\
& p_i \in \{0,1\} && \forall i \in \mathcal{I} && \text{(2.9e)} \\
& q_{ij} \in \{0,1\} && \forall (i,j) \in \mathcal{A} && \text{(2.9f)}
\end{aligned}$$

Proposition 2.11. *(SEP-U-STAR) is polynomially solvable.*

Proof. The objective coefficients of variables q_{ij} are non-positive. As (SEP-U-STAR)

is a minimization problem, in contrast to (SEP-S-STAR), there exists an optimal solution with a q_{ij} value as large as possible. Therefore, the lower bounding constraints (2.9d) are redundant and can be dropped. Then, each row of the matrix corresponding to the remaining constraints (2.9b) and (2.9c) contains only two nonzero coefficients of 1 and -1 which indicates that it is totally unimodular (Nemhauser & Wolsey, 1988). Therefore, the problem can be solved by solving its LP relaxation within a polynomial time. □

The separation problem (SEP-U-STAR) also can be converted to maximum flow problems which are defined on networks $\mathcal{G}^s$ for $s \in \mathcal{I}$, similar to (SEP-S-STAR). The capacity C_{ij} is now defined as follows:

- $C_{si} = M + \delta_{si}$ for all $i \in \mathcal{I} \setminus \{s\}$,
- $C_{i0} = M + \gamma_i - \sum_{j \in \mathcal{I}} \delta_{ij}$ for all $i \in \mathcal{I} \setminus \{s\}$,
- $C_{ij} = \delta_{ij}$ for all $(i, j) \in \mathcal{A}$ such that $i \neq s$,
- $C_{s0} = 0$.

M, which is defined as $\max\big\{ - \min_{i \in \mathcal{I}}\{\gamma_i - \sum_{j \in \mathcal{I}} \delta_{ij}\}, 0\big\}$, is added to arcs (s, i) and $(i, 0)$ for all $i \in \mathcal{I} \setminus \{s\}$ to prevent negative arc capacity. Note that among these $2(I-1)$ arcs with additionally added M, exactly $I-1$ arcs are always selected when we find minimum $(s, 0)$ cut. The resulting graph is illustrated in Figure 2.4. When the capacity of the found minim $(s, 0)$ cut is smaller than $(I-2)M - (\gamma_s - \sum_{i \in \mathcal{I}} \delta_{si}$, the violated U-STAR inequality is found. Otherwise, there is no violated U-STAR inequality which is defined by a subset of nodes containing an item s. The overall separation procedure of U-STAR inequality is omitted here because it is similar to that of S-STAR inequality provided in Algorithm 2.1

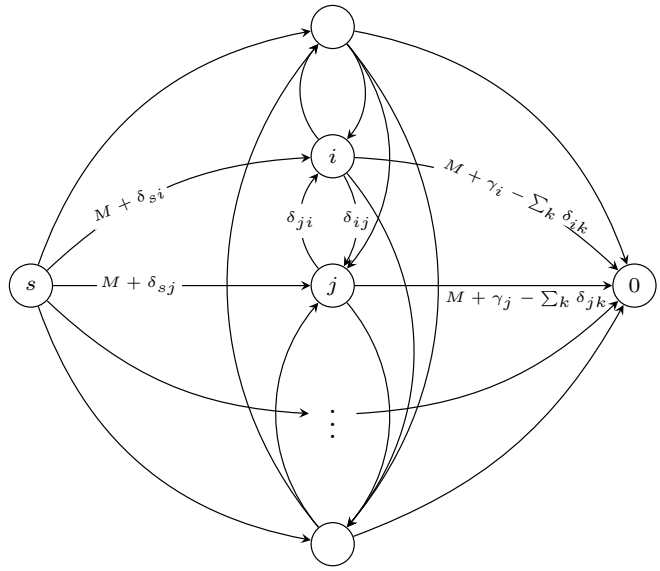


Figure 2.4: Construction of the network for the separation procedure of `U-STAR` inequality

It is well-known that `GSEC` also can be separated in polynomial time by solving maximum-flow problems (Wolsey, 2020). Therefore, in our implementation of the separation algorithms, we used the maximum flow algorithms for all three inequalities. We denote the formulations obtained by replacing the constraints (2.3g) in formulation (2.3) with the `S-STAR` inequality (2.5), and `U-STAR` inequality (2.8) as `(S-STAR)` and `(U-STAR)`, respectively.

2.4.5 General Representation of the Inequalities

Lastly, we note that the `S-STAR` and `U-STAR` inequalities can be represented in a more generalized form which is related to the results of Avella et al. (2018) who addressed *inventory routing problem* (IRP). IRP considers both the management of inventory and routing of vehicles over multiple periods. The authors studied the single-period substructure of IRP and proposed valid inequalities which are denoted

as *disjoint route inequalities*. The disjoint route inequality can be applied to LSP after making slight modifications which are shown in the following proposition.

Proposition 2.12. *Given a subset $S \subseteq \mathcal{I}$ and coefficients μ_{ij} for all $(i,j) \in \mathcal{A}_0$, the inequality*

$$\sum_{(i,j)\in\mathcal{A}_0} \mu_{ij} z_{ij} \geq \sum_{i\in S} x_i \tag{2.10}$$

is valid if

$$\sum_{(i,j)\in C} \mu_{ij} \geq \min\Big(K - \sum_{(i,j)\in C} st_{ij}, \sum_{i\in V(C)\cap S} u_i\Big) \tag{2.11}$$

for every feasible cycle $C = \{(i,j) : \bar{z}_{ij} = 1,\ (i,j) \in \mathcal{A}_0\}$ that can be formed by a feasible solution $(\bar{\mathbf{x}}, \bar{\mathbf{s}}^+, \bar{\mathbf{s}}^-, \bar{\mathbf{y}}, \bar{\mathbf{z}})$. Let $V(C)$ denotes the set of nodes included in C.

Proof. For a given feasible solution $(\bar{\mathbf{x}}, \bar{\mathbf{s}}^+, \bar{\mathbf{s}}^-, \bar{\mathbf{y}}, \bar{\mathbf{z}}) \in \mathcal{X}$,

$$\sum_{(i,j)\in\mathcal{A}_0} \mu_{ij}\bar{z}_{ij} = \sum_{(i,j)\in C} \mu_{ij}$$

for the cycle C and

$$\sum_{i\in\mathcal{I}} \bar{x}_i + \sum_{(i,j)\in\mathcal{A}} st_{ij}\bar{z}_{ij} = \sum_{i\in V(C)} \bar{x}_i + \sum_{(i,j)\in C} st_{ij}\bar{z}_{ij} \leq K.$$

Therefore,

$$\sum_{i\in V(C)\cap S} \bar{x}_i \leq K - \sum_{(i,j)\in C} st_{ij}\bar{z}_{ij} - \sum_{i\in V(C)\cap S^c} \bar{x}_i \leq K - \sum_{(i,j)\in C} st_{ij}$$

which leads to

$$\min\Big(K - \sum_{(i,j)\in C} st_{ij}, \sum_{i\in V(C)\cap S} u_i\Big) \geq \min\Big(\sum_{i\in V(C)\cap S} x_i, \sum_{i\in V(C)\cap S} u_i\Big) \geq \sum_{i\in V(C)\cap S} x_i.$$

This shows the proposition holds. □

The above inequality (2.10) is quite general: there is a high degree of freedom in determining the coefficients μ_{ij}. Therefore, the separation of the inequality (2.10) in its current form seems difficult even for a given S as mentioned by Avella et al. (2018). The authors considered subfamilies by restricting the structure of the inequality. The `S-STAR` and `U-STAR` inequalities also can be regarded as subfamilies of the above inequalities which are demonstrated by setting μ_{ij} values as follows:

- `S-STAR` inequality:

$$\mu_{ij} = \begin{cases} K - st_{ij}, & (i,j) \in \delta^+(S) \\ -st_{ij}, & (i,j) \in \delta^-(S) \cup E(S) \\ 0, & \text{otherwise.} \end{cases}$$

- `U-STAR` inequality:

$$\mu_{ij} = \begin{cases} u_j, & (i,j) \in \delta^-(S) \\ \min\{K - st_{ij} - u_i, u_j\}), & (i,j) \in E(S) \\ 0, & \text{otherwise.} \end{cases}$$

It is easy to show that, by setting μ_{ij} as above, the condition (2.11) is satisfied.

2.5 Extended Formulations

In this section, we introduce extended formulations—single/multi-commodity flow formulations and time-flow formulations—for the single-period substructure. Although we present them for the single-period substructure, they can be straightforwardly adapted to LSP with multiple time periods.

2.5.1 Single-commodity Flow Formulations

The single-commodity flow formulation, first proposed by Gavish & Graves (1978) for TSP, has been frequently used to model many routing problems. Guimarães et al. (2014) used this formulation to model LSP. By defining variable f_{ij} for $(i,j) \in \mathcal{A}_0$ representing the commodity flow along the arc (i,j), the formulation can be written as follows.

$$\sum_{i \in \mathcal{I}} f_{0i} = \sum_{i \in \mathcal{I}} y_i \tag{2.12a}$$

$$\sum_{j \in \mathcal{I}_0 \setminus \{i\}} f_{ji} - \sum_{j \in \mathcal{I}_0 \setminus \{i\}} f_{ij} = y_i \qquad \forall i \in \mathcal{I} \tag{2.12b}$$

$$0 \leq f_{ij} \leq I z_{ij} \qquad \forall (i,j) \in \mathcal{A}_0 \tag{2.12c}$$

Constraints (2.12a) indicate that the amount of commodities sent from node 0 is equal to the total number of items produced. Constraints (2.12b) ensure that if the item i is produced, the amount of commodity decreases by one, whereas it does not change if i is not produced. Constraints (2.12c) represent the relation between the variables $\mathbf{f}$ and $\mathbf{z}$ and impose the upper bound on the amount of the commodity. If $z_{ij} = 1$, f_{ij} represents the amount of commodity that flows along arc (i,j) and

$f_{ij} = 0$, if $z_{ij} = 0$. By replacing constraints (2.3g) with (2.12), the first single-commodity flow formulation which is denoted as `(SCF1)` can be obtained.

It is commonly known (Gouveia, 1995) that a stronger formulation denoted as `(SCF2)` can be obtained by replacing the bound constraints (2.12c) of `(SCF1)` with the tighter constraints (2.13).

$$z_{0i} \leq f_{0i} \leq I z_{0i} \qquad \forall i \in \mathcal{I} \tag{2.13a}$$

$$z_{ij} \leq f_{ij} \leq (I-1) z_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{2.13b}$$

$$f_{i0} = 0 \qquad \forall i \in \mathcal{I} \tag{2.13c}$$

Both formulations `(SCF1)` and `(SCF2)` prevent any cycles without node 0. Furthermore, the following relations hold among `(SCF1)`, `(SCF2)`, and `(GSEC)`.

Proposition 2.13. *Let $z(\mathtt{F})$ be the LP relaxation bound obtained from the given formulation* (`F`). *Then,*

$$z(\mathtt{SCF1}) \leq z(\mathtt{SCF2}) \leq z(\mathtt{GSEC}).$$

Proof. By projecting both `(SCF1)` and `(SCF2)` onto the original space, one can obtain the following inequalities (2.14) and (2.15), respectively.

$$\sum_{i \in S} y_i \leq I \sum_{(i,j) \in \delta^-(S)} z_{ij} \qquad \forall S \subseteq \mathcal{I} \tag{2.14}$$

$$\sum_{i \in S} y_i \leq I \sum_{(i,j) \in \delta^-(S)} z_{ij} - \sum_{i \in S} \sum_{j \in \mathcal{I} \setminus S} z_{ij} - \sum_{i \in \mathcal{I} \setminus S} \sum_{j \in S} z_{ij} \qquad \forall S \subseteq \mathcal{I} \tag{2.15}$$

No other inequalities can be obtained by the projection because of Hoffman's circu-

lation theorem (Hoffman, 1976). It is trivial that inequalities (2.14) are dominated by inequalities (2.15). We show that inequalities (2.15) are implied by the `GSEC`. For a given $S \subseteq \mathcal{I}$, by aggregating the `GSEC` for all $i \in S$, we obtain $\sum_{i \in S} y_i \leq |S| \sum_{(i,j) \in \delta^-(S)} z_{ij}$. Therefore, it is sufficient to show that

$$\sum_{i \in S} \sum_{j \in \mathcal{I} \setminus S} z_{ij} + \sum_{i \in \mathcal{I} \setminus S} \sum_{j \in S} z_{ij} \leq (I - |S|) \sum_{(i,j) \in \delta^-(S)} z_{ij}. \tag{2.16}$$

When $|S| = I$, it naturally holds because $\mathcal{I} \setminus S = \emptyset$. When $|S| = I - 1$, $|\mathcal{I} \setminus S| = 1$, and let k be the only element in $\mathcal{I} \setminus S$. Then,

$$\sum_{i \in S} \sum_{j \in \mathcal{I} \setminus S} z_{ij} + \sum_{i \in \mathcal{I} \setminus S} \sum_{j \in S} z_{ij} = \sum_{i \in S} (z_{ik} + z_{ki})$$

and

$$(I - |S|) \sum_{(i,j) \in \delta^-(S)} z_{ij} = \sum_{i \in S} \sum_{j \in \mathcal{I}_0 \setminus S} z_{ij} = \sum_{i \in S} (z_{ik} + z_{i0}) = \sum_{i \in S} z_{ik} + 1 - z_{k0}$$

as $\sum_{i \in \mathcal{I}} z_{i0} = 1$. Because $\sum_{i \in S} z_{ki} + z_{k0} \leq 1$, inequality (2.16) also holds. Finally, when $|S| \leq I - 2$, inequality (2.16) can be written as

$$\begin{aligned} \sum_{i \in S} \sum_{j \in \mathcal{I} \setminus S} z_{ij} + \sum_{i \in \mathcal{I} \setminus S} \sum_{j \in S} z_{ij} &\leq \sum_{(i,j) \in \delta^+(S) \cup \delta^-(S)} z_{ij} \\ &= 2 \sum_{(i,j) \in \delta^-(S)} z_{ij} \leq (I - |S|) \sum_{(i,j) \in \delta^-(S)} z_{ij}, \end{aligned}$$

and therefore, it also holds. □

2.5.2 Multi-commodity Flow Formulations

Multi-commodity flow formulations have also been studied for various routing problems. Sarin et al. (2011) presented a multi-commodity flow formulation for Chesapeake problem which is an instance of LSP. In contrast to single-commodity flow formulations, multi-commodity flow formulations define one commodity per item. Let us define the binary variable q_{ij}^k for $i \in \mathcal{I}_0$ and $j, k \in \mathcal{I}$ which represents whether arc (i, j) is traversed on the way from node 0 to node k. Accordingly, the following constraints are formed.

$$\sum_{j \in \mathcal{I}} q_{0j}^k = y_k \qquad \forall k \in \mathcal{I} \qquad (2.17a)$$

$$\sum_{j \in \mathcal{I}_0 \setminus \{k\}} q_{jk}^k = y_k \qquad \forall k \in \mathcal{I} \qquad (2.17b)$$

$$\sum_{j \in \mathcal{I}_0 \setminus \{i\}} q_{ji}^k = \sum_{j \in \mathcal{I} \setminus \{i\}} q_{ij}^k \qquad \forall i, k \in \mathcal{I}, k \neq i \qquad (2.17c)$$

$$q_{kj}^k = 0 \qquad \forall j, k \in \mathcal{I} \qquad (2.17d)$$

$$0 \leq q_{ij}^k \leq z_{ij} \qquad \forall i \in \mathcal{I}_0, j, k \in \mathcal{I} \qquad (2.17e)$$

Constraints (2.17a) and (2.17b) ensure that when item k is produced, the corresponding commodity should flow from node 0 to node k. Constraints (2.17c) are the flow balance constraints. Constraints (2.17d) and (2.17e) ensure that the commodity variable can have a nonzero value only when the corresponding arc is traversed. By replacing constraints (2.3g) with the set of constraints (2.17), the multi-commodity flow formulation denoted as `(MCF1)` can be obtained. It is known that by projecting `(MCF1)` onto the original space, `(GSEC)` is obtained, and they provide the same LP

bounds (Padberg & Sung, 1991).

Because (MCF1) only uses additional variables q_{ij}^k regarding the sequencing decisions, there are no considerations for lot-sizing decisions. In this regard, to further enhance the LP relaxation bound, we derive the following additional constraints that incorporate both decisions.

$$\sum_{(i,j)\in\mathcal{A}} st_{ij} q_{ij}^k + x_k \leq K y_k \qquad \forall k \in \mathcal{I} \qquad (2.18a)$$

$$st_{ij} q_{ij}^j + st_{ji} q_{ji}^i + x_i + x_j \leq K(y_i + y_j - z_{ij} - z_{ji}) \qquad \forall (i,j) \in \mathcal{A} \qquad (2.18b)$$

It is not difficult to show that inequalities (2.18a) and (2.18b) are valid. Incorporating these inequalities into (MCF1), a tighter formulation (MCF2) is obtained. Their strength and effectiveness are further investigated through computational experiments. Additionally, we obtain the following corollary.

Corollary 2.14. $z(\texttt{SCF1}) \leq z(\texttt{SCF2}) \leq z(\texttt{GSEC}) = z(\texttt{MCF1}) \leq z(\texttt{MCF2})$.

2.5.3 Time-flow Formulations

The time-flow formulations are similar to the single-commodity flow formulations, except that they represent the flow of time instead of the commodity. To present the formulations, we define a set of time-flow variables w_{ij} for $(i,j) \in \mathcal{A}_0$ which represents the remaining capacity when the setup from item i to item j begins.

$$\sum_{j\in\mathcal{I}_0\setminus\{i\}} (w_{ji} - st_{ji} z_{ji}) - x_i = \sum_{j\in\mathcal{I}_0\setminus\{i\}} w_{ij} \qquad \forall i \in \mathcal{I} \qquad (2.19a)$$

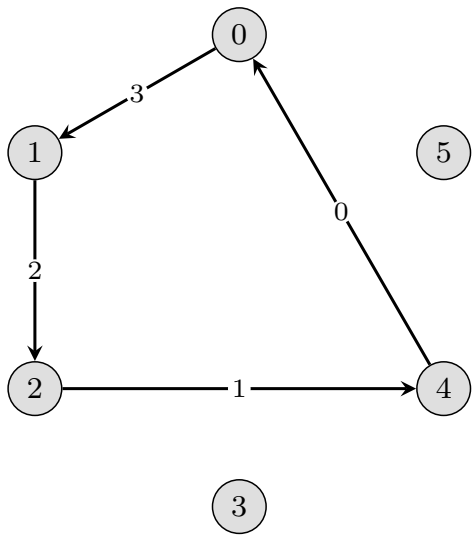
$$0 \leq w_{ij} \leq K z_{ij} \qquad \forall (i,j) \in \mathcal{A}_0 \qquad (2.19b)$$

Constraints (2.19a) are time flow balance equations that ensure that by subtracting the setup and production time for item i from the given capacity, one can obtain the remaining capacity. Constraints (2.19b) impose upper bounds on the time-flow variables. We call this formulation `(TF1)`. Similar to `(SCF1)`, `(TF1)` can be further strengthened with tighter bound constraints (2.20) instead of (2.19b) which we call `(TF2)`.

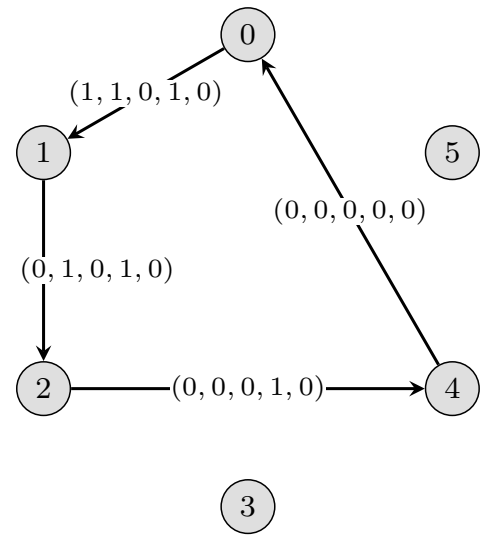
$$st_{ij} z_{ij} \leq w_{ij} \leq K z_{ij} \qquad \forall (i,j) \in \mathcal{A}_0,\ i \neq 0 \tag{2.20a}$$

$$w_{0i} = K z_{0i} \qquad \forall i \in \mathcal{I} \tag{2.20b}$$

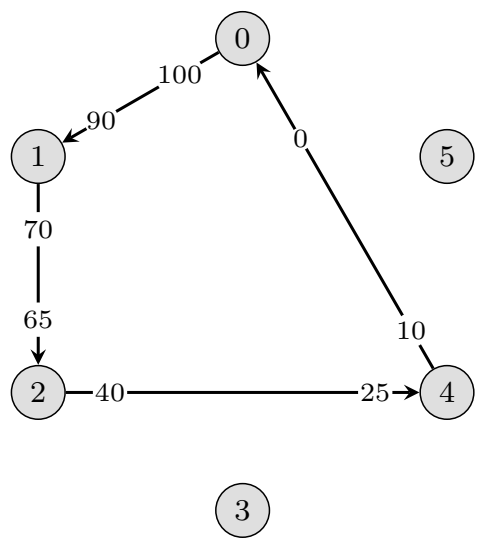
The time-flow formulations can represent the consumption of the capacity and therefore, can additionally incorporate the production amount and setup times which distinguishes them from the single-commodity flow formulations. This difference is illustrated in Figure 2.5. The single and multi-commodity flow variables in Figures 2.5(a) and 2.5(b) contain only production sequence information. On the other hand, the time-flow variables in Figure 2.5(c) additionally contain information on the production amount and the remaining capacity which can be more beneficial in modeling LSP.



(a) Single-commodity flow formulation



(b) Multi-commodity flow formulation



(c) Time-flow formulation

Figure 2.5: Illustration of the extended formulations

Finally, we provide the relation between the time-flow formulations and S-STAR inequalities.

Proposition 2.15. *Projection of* (TF2) *onto the original space results in* $\mathtt{S-STAR}$ *inequalities* (2.5)*. Furthermore,* $z(\mathtt{TF1}) \leq z(\mathtt{TF2}) = z(\mathtt{S-STAR})$.

Proof. For a given $S \subseteq \mathcal{I}$, by adding the inequalities (2.19a) for all $i \in S$ and rearranging the terms, one can obtain

$$\sum_{(j,i)\in E(S)\cup\delta^-(S)} st_{ji}z_{ji} + \sum_{i\in S} x_i = \sum_{i\in S}\Big(\sum_{j\in\mathcal{I}_0\setminus\{i\}} w_{ji} - \sum_{j\in\mathcal{I}_0\setminus\{i\}} w_{ij}\Big) = \sum_{(j,i)\in\delta^-(S)} w_{ji} - \sum_{(i,j)\in\delta^+(S)} w_{ij}.$$

From the bound constraints (2.20),

$$\sum_{(j,i)\in E(S)\cup\delta^-(S)} st_{ji}z_{ji} + \sum_{i\in S} x_i = \sum_{(j,i)\in\delta^-(S)} w_{ji} - \sum_{(i,j)\in\delta^+(S)} w_{ij} \leq K \sum_{(i,j)\in\delta^-(S)} z_{ij} - \sum_{(i,j)\in\delta^+(S)} st_{ij}z_{ij}$$

can be obtained, which is the S-STAR inequality for $S \subseteq \mathcal{I}$. This indicates that all feasible solutions of (TF2) satisfy the S-STAR inequalities. Moreover, due to Hoffman's circulation theorem (Hoffman, 1976) it can be shown that $z(\mathtt{TF2}) = z(\mathtt{S-STAR})$. □

There are no other dominance relations between the time-flow formulations and commodity-flow formulations. Their strengths are compared through computational experiments, as discussed in the next section.

2.6 Computational Experiments

2.6.1 Experiment Settings

We stress that the aim of the computational experiments here is to compare the strengths of various inequalities and formulations, rather than to solve real-world LSP instances. Thus, we compare the LP relaxation bounds of the extended formulations, that is, `(SCF1)`, `(SCF2)`, `(MCF1)`, `(MCF2)`, `(TF1)`, and `(TF2)`, and the formulations with valid inequalities, that is, `(GSEC)`, `(S-STAR)`, and `(U-STAR)`.

To compute the bound value with a particular type of valid inequalities, we exhaustively separate the violated inequalities at the root node until none is found or the objective value does not improve during the last 100 iterations of the separation procedure. Furthermore, for the purpose of comparison, we also report the results when all three types of inequalities are separated which is denoted as `(ALL)`.

All experiments were conducted on an Intel Core 3.10 GHz PC with 16 GB RAM under Windows 10 Pro. The separation algorithms and mathematical formulations were implemented using C++. FICO Xpress 8.12 with its default parameter settings was used as the LP solver.

We use two sets of test instances, that is, single-period and multi-period instances that are generated following the instance-generation scheme proposed by Almada-Lobo, Klabjan, et al. (2007) which has been frequently used in the literature. The descriptions of the instances are given below.

Single-period Instances

The type of single-period instance is defined by the combination of the number of items (I), capacity utilization parameter (UTIL), setup cost parameter (SC), and upper bound parameter (IUB). These parameters are adopted from the study of Almada-Lobo, Klabjan, et al. (2007), except for the last parameter which is additionally defined to determine whether there is an individual upper bound for the production amount of each item (IUB = T) or not (IUB = F).

The demand for item i, d_i, is generated from $DU[40, 60]$, and capacity K is set to $I \cdot d_{avg}/\rho$. The unit surplus and shortage costs of item i, that is, hc_i and bc_i, respectively, are generated from $DU[2, 10]$. In addition, we use a negative unit production cost, that is, profit, $pc_i = -1$, to make production profitable. The setup time st_{ij} is drawn from $DU[0.05K, 0.1K]$, and the setup cost is set as $sc_{ij} = \texttt{SC} \cdot st_{ij}$. The upper bound u_i for item i is generated from $DU[d_i + 1, K]$ if IUB = T, while it is set to K if IUB = F. We use the following parameters: $I \in \{5, 15, 25, 35\}$, UTIL = $\{0.6, 0.8, 1\}$, SC $\in \{50, 100\}$, IUB $\in \{\texttt{T}, \texttt{F}\}$. For each combination, we generated 100 instances, resulting in a total of 4800 single-period instances.

Multi-period Instances

Multi-period instances are also generated, similar to the single-period instances. The main difference is in setting the cost parameters which are set as $pc_{it} = 1$, $hc_{it} \sim DU[2, 10]$, $bc_{it} \sim DU[10, 50]$, and $sc_{ij} = \texttt{SC} \cdot st_{ij}$. We use the following parameters: $I \in \{5, 15, 25\}$, $T \in \{5, 15, 25\}$, UTIL = $\{0.6, 0.8, 1\}$, SC $\in \{50, 100\}$, IUB $\in \{\texttt{T}, \texttt{F}\}$. For each combination, we generated 10 instances, resulting in a total of 1080 multi-period instances.

2.6.2 Experiment Results on Single-period Instances

We compared the strengths of the extended formulations and valid inequalities using the following measures:

- *LP gap* (%): $\frac{\texttt{(OPT)} - z(*)}{\texttt{(OPT)}} \times 100$,
- *Closed gap* (%): $\frac{z(*) - z\texttt{(PURE)}}{\texttt{(OPT)} - z\texttt{(PURE)}} \times 100$,

where `(OPT)` is an optimal objective value, and `(PURE)` is the basic lower bound obtained using formulation (2.3) without `GSEC` (2.3g). Note that, therefore, `(PURE)` itself is not a valid formulation and is only used for comparison. The *LP gap* and *Closed gap* results for single-period instances are provided in Figures 2.6 and 2.7, respectively. The corresponding detailed results are reported in Tables B.1 and B.2 in Appendix B.

First, the following relations established in Section 2.4 and 2.5 are verified based on the results:

- $z\texttt{(SCF1)} \leq z\texttt{(SCF2)} \leq z\texttt{(MCF1)} = z\texttt{(GSEC)} \leq z\texttt{(MCF2)}$,
- $z\texttt{(TF1)} \leq z\texttt{(TF2)} = z\texttt{(S-STAR)}$, and
- $z\texttt{(U-STAR)} \leq z\texttt{(S-STAR)}$ when $\texttt{IUB} = \texttt{F}$.

The newly proposed formulations `(TF1)` and `(TF2)` provide considerably tighter LP bounds than the other extended formulations. They can close approximately 40% of the gap on average, whereas only approximately 3% can be closed by `(MCF1)`. The effect of the constraints added in `(MCF2)` is considerably significant, whereas the strengthened bounds of `(SCF2)` and `(TF2)` are not particularly significant compared with `(SCF1)` and `(TF1)`, respectively.

When IUB = F, (S-STAR) provides better results than (U-STAR) because U-STAR inequalities are dominated by S-STAR inequalities in this case. On the other hand, (U-STAR) provides slightly better results than (S-STAR) when IUB = T which indicates that U-STAR inequalities are beneficial when there exists a non-trivial upper bound on the production amount of each item. Obviously, the tightest bound is obtained when all three types of inequalities are added in (ALL). The results also show that the three types of inequalities do not dominate each other.

Additionally, we report the ratio of instances in which the LP gap is below certain values in Figure 2.8 and Table B.3. Similar to the above results, (TF2) provides the best results among the extended formulations. Using (TF2), in approximately 8.6% of the instances, one can obtain the same LP bound as the optimal objective value.

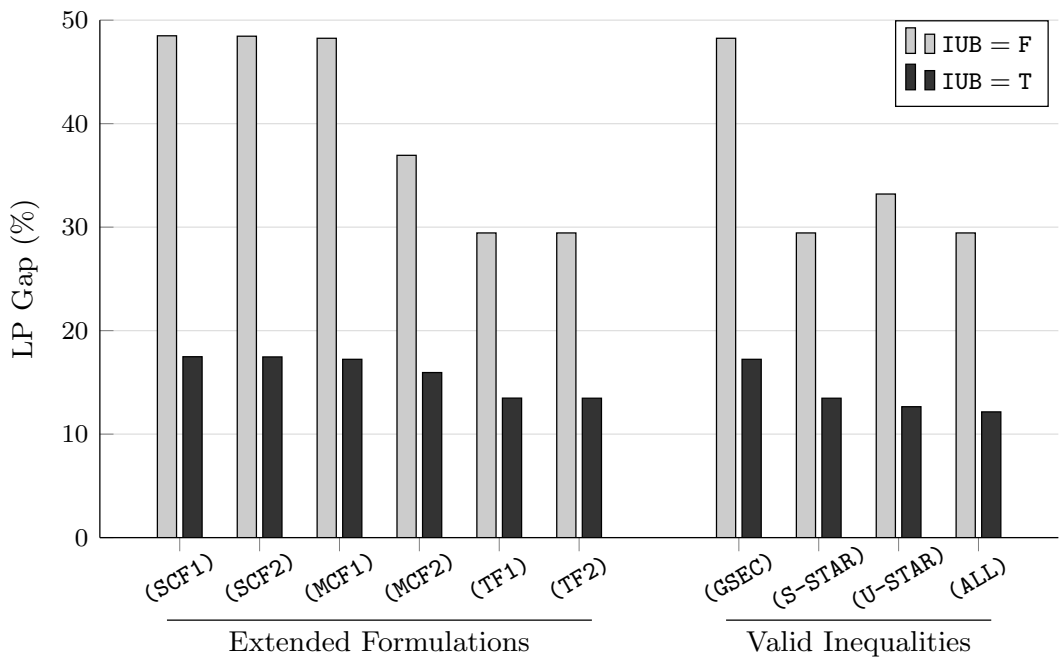


Figure 2.6: Test results on single-period instances: LP gap

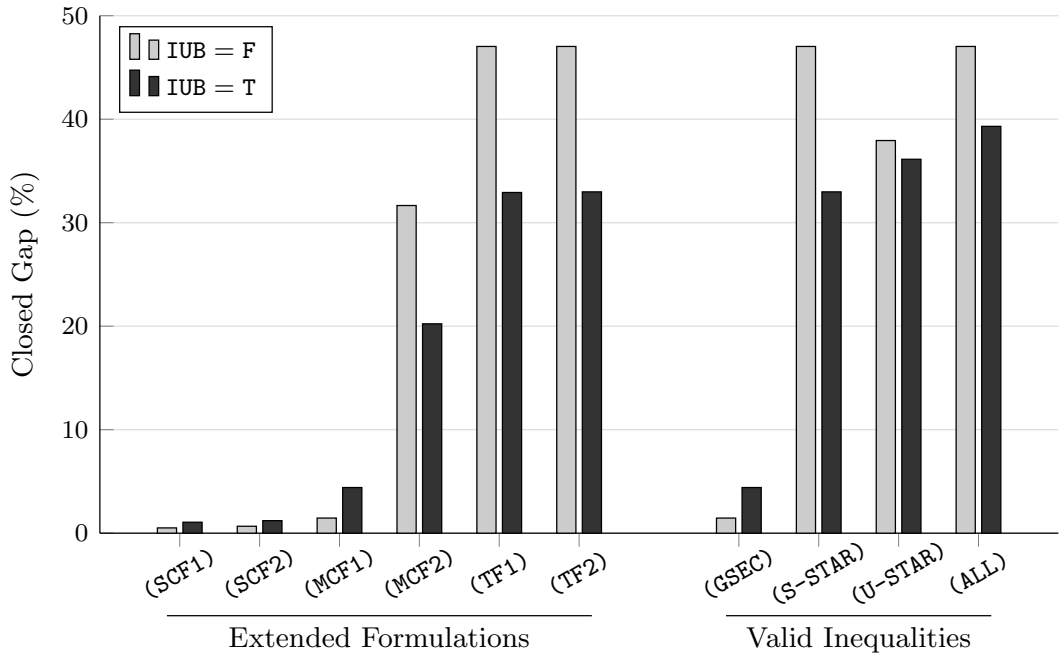


Figure 2.7: Test results on single-period instances: Closed gap

Moreover, in approximately 40% of the instances, the LP gap is smaller than 10%.

In Table 2.2, we present the sizes of the extended formulations relative to the smallest formulation `(SCF1)`. As expected, the size of the multi-commodity flow formulations is the largest, and the relative size increases with the number of items. In particular, owing to additional constraints, `(MCF2)` has the largest number of variables and constraints. Averagely, `(MCF2)` requires 9.95 times more variables and 19.33 times more constraints than `(SCF1)`. This results in an increase in computation time, as demonstrated in Table 2.3, which shows the difficulty of using multi-commodity flow formulations in practice.

The number of separated inequalities is reported in Table 2.4. On average, approximately four times more inequalities are added when using the `S-STAR` inequality

Table 2.2: Test results on single-period instances: Relative formulation size

I	*Variables*			*Constraints*			
	(SCF1) =(TF1)	(SCF2) =(TF2)	(MCF1) =(MCF2)	(SCF1) =(TF1)	(SCF2) =(TF2)	(MCF1)	(MCF2)
5	1.00	1.00	2.67	1.00	1.47	3.81	4.66
15	1.00	1.00	7.43	1.00	1.74	12.88	14.31
25	1.00	1.00	12.36	1.00	1.83	22.58	24.21
35	1.00	1.00	17.33	1.00	1.87	32.43	34.15
Average	1.00	1.00	9.95	1.00	1.73	17.93	19.33

than when using GSEC. This can be considered as a trade-off for the tighter bound. When using all three types, (ALL), the number of added inequalities is smaller than the sum of that when each type is used individually.

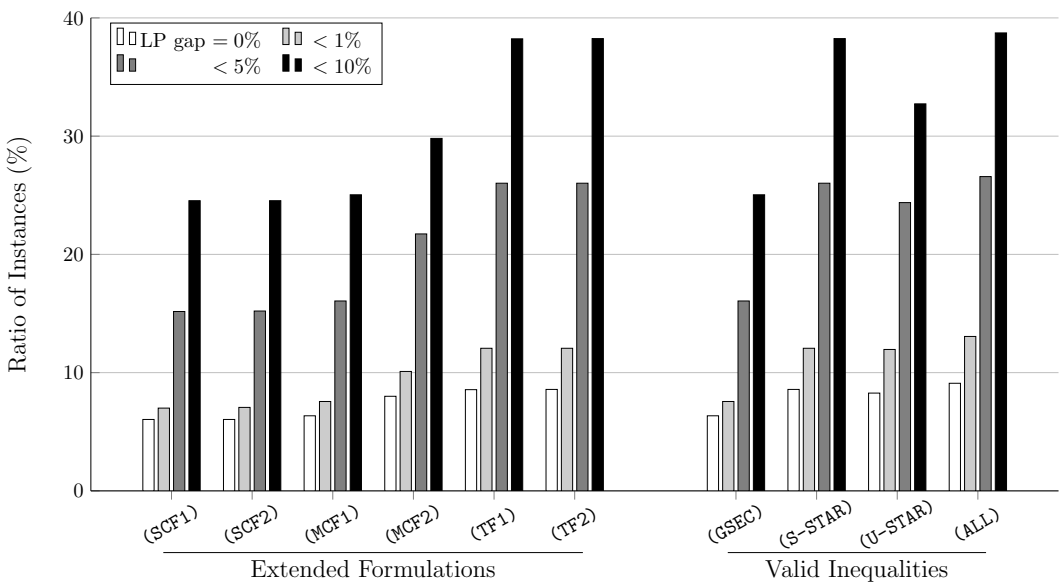


Figure 2.8: Test results on single-period instances: Ratio of instances with LP gap below certain values

Table 2.3: Test results on single-period instances: Computation time

I	*Computation Time (s)*					
	(SCF1)	(SCF2)	(TF1)	(TF2)	(MCF1)	(MCF2)
5	0.002	0.002	0.002	0.003	0.003	0.004
15	0.005	0.006	0.007	0.008	0.030	0.144
25	0.012	0.015	0.026	0.032	0.292	1.579
35	0.028	0.035	0.071	0.073	2.297	11.368
Average	0.012	0.014	0.027	0.029	0.656	3.274

Table 2.4: Test results on single-period instances: Number of added inequalities

Factors		*Number of added inequalities*			
		(GSEC)	(S-STAR)	(U-STAR)	(ALL)
I	5	0.8	1.9	0.9	3.0
	15	4.0	10.3	3.4	14.8
	25	5.7	22.8	6.7	30.9
	35	6.2	38.0	10.2	49.0
UTIL	60	4.1	18.3	5.0	23.8
	80	4.2	18.0	4.9	23.9
	100	4.2	18.5	6.0	25.6
SC	50	4.7	29.6	8.2	38.1
	100	3.6	6.9	2.3	10.7
IUB	F	4.7	16.9	2.7	22.5
	T	3.6	19.6	7.9	26.4
Average		4.2	18.2	5.3	24.4

2.6.3 Experiment Results on Multi-period Instances

For the results of the multi-period instances, we report the strength of the formulations relative to the bound which can be obtained when we know the ideal formulation of the single-period substructure, that is, $conv(\mathcal{X})$. This bound is de-

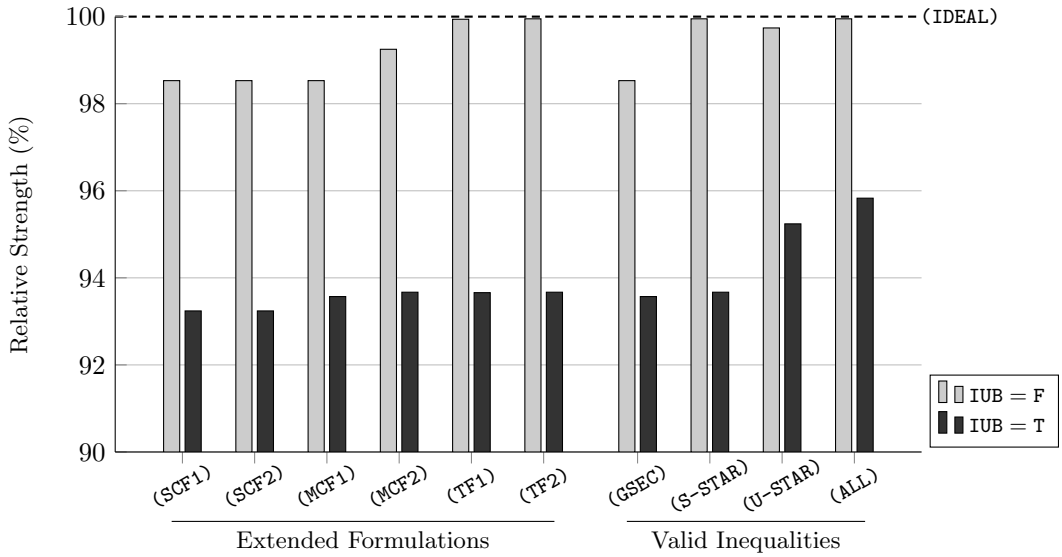


Figure 2.9: Test results on multi-period instances: LP strength

noted by z(IDEAL). To calculate z(IDEAL), we define a pattern-based formulation whose LP relaxation is solved using a column generation procedure. Because they are outside the scope, we do not provide detailed descriptions of the pattern-based formulation and column generation procedure in this section. They are provided in Appendix A. We denote the strength of a formulation (F) as $\frac{z(\text{F})}{z(\text{IDEAL})} \times 100$ and present the corresponding results in Figure 2.9. The detailed results are provided in Table B.4 in Appendix B.

Similar to the results of the single-period instances, the newly proposed formulations and inequalities are successful in providing tight bounds for the multi-period instances. On average, they provide only approximately 3.2% less tight bounds relative to (IDEAL). In particular, when IUB = F, they provide almost the same LP bound (99.95%) as that of the ideal formulation. When IUB = T, the advantage of (U-STAR) is emphasized as it provides the tightest LP bound among others.

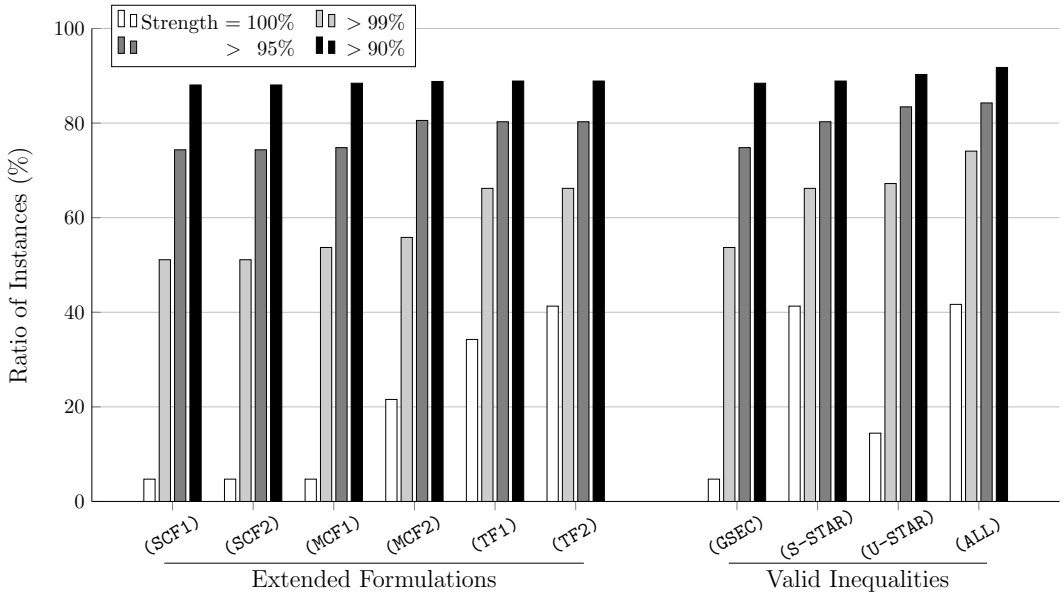


Figure 2.10: Test results on multi-period instances: Ratio of instances with LP strength above certain values

In addition, we report the ratio of instances whose LP strength is above certain values in Figure 2.10 and Table B.5. In approximately 40% of the instances, `(TF2)` and `(S-STAR)` provide the same LP bound as that of the ideal formulation. In addition, the difference is less than 1% for more than 65% of instances. There is little difference between the formulations and inequalities in terms of the number of instances with a strength greater than 90%.

In Table 2.5, we present the sizes of the formulations relative to the smallest one `(SCF1)`. As expected, the multi-commodity flow formulations are the largest. Particularly, when $I = 25$, `(MCF2)` requires 12.4 times more variables and 23.5 times more constraints than `(SCF1)` which may be prohibitively large. The average computation times for different problem dimensions are shown in Table 2.6. This result also demonstrates that multi-commodity formulations are not viable options

Table 2.5: Test results on multi-period instances: Relative formulation size

Factors		*Variables*			*Constraints*			
		(SCF1) =(TF1)	(SCF2) =(TF2)	(MCF1) =(MCF2)	(SCF1) =(TF1)	(SCF2) =(TF2)	(MCF1)	(MCF2)
	5	1.00	1.00	2.67	1.00	1.43	3.59	4.37
I	15	1.00	1.00	7.43	1.00	1.71	12.37	13.75
	25	1.00	1.00	12.36	1.00	1.81	21.95	23.53
	5	1.00	1.00	7.49	1.00	1.65	12.69	13.94
T	15	1.00	1.00	7.49	1.00	1.65	12.62	13.87
	25	1.00	1.00	7.49	1.00	1.65	12.61	13.85
Average		1.00	1.00	7.49	1.00	1.65	12.64	13.89

Table 2.6: Test results on multi-period instances: Computation time

Factors		*Computation Time (s)*					
		(SCF1)	(SCF2)	(TF1)	(TF2)	(MCF1)	(MCF2)
	5	0.01	0.02	0.02	0.02	0.03	0.05
I	15	0.12	0.18	0.16	0.26	1.40	4.25
	25	0.53	0.96	0.92	1.36	15.07	31.89
	5	0.04	0.06	0.06	0.10	0.62	1.90
T	15	0.21	0.36	0.36	0.55	6.10	10.47
	25	0.42	0.75	0.68	0.99	9.79	23.81
Average		0.22	0.39	0.37	0.55	5.50	12.06

when the problem dimension increases.

Regarding the valid inequalities, as reported in Table 2.7, the numbers of added inequalities of (S-STAR) and (U-STAR) are much larger than that of GSEC which can be regarded as a trade-off for the tighter bound. We observe that most of the S-STAR and U-STAR inequalities added after earlier iterations do not contribute significantly to the strengthening of the bounds. Nevertheless, owing to the original purpose of the

Table 2.7: Test results on multi-period instances: Number of added inequalities

Factors		*Number of added inequalities*			
		(GSEC)	(S-STAR)	(U-STAR)	(ALL)
I	5	4.9	26.9	27.8	66.9
	15	14.4	147.0	152.1	300.6
	25	25.5	296.9	382.3	574.5
T	5	5.6	69.1	77.0	125.4
	15	15.7	154.8	172.2	309.6
	25	23.6	247.0	313.1	507.0
UTIL	60	11.3	41.5	47.5	101.8
	80	16.2	67.7	69.9	152.0
	100	17.3	361.6	444.9	688.2
SC	50	18.3	153.3	180.7	309.4
	100	11.6	160.6	194.2	318.7
IUB	F	0.0	215.3	263.8	376.0
	T	29.9	98.6	111.1	252.0
Average		15.0	156.9	187.4	314.0

experiments, we did not terminate the separation which led to a number of additional inequalities, most of which were insignificant. In this regard, if these inequalities are used as cutting planes in tree-search algorithms, more detailed analysis and further studies on their effects are required.

2.7 Summary

In this chapter, we addressed the lot-sizing and scheduling problem with sequence-dependent setups and its single-period substructure. We presented S-STAR and U-STAR inequalities which are new families of valid inequalities for the problem.

The theoretical strength and separation complexity of the proposed inequalities are discussed and the facet-defining conditions are also identified. In addition, we presented polynomial-time separation procedures of these inequalities which use the maximum-flow algorithm. New extended formulations based on the time-flow variables are proposed and compared with the existing ones. Results of computational experiments on both single-period and multi-period instances demonstrated distinct advantages of the newly proposed inequalities and extended formulations in tightening the LP relaxation bounds.

Chapter 3

New Optimization Models for Lot-sizing and Scheduling Problem with Sequence-dependent Setups, Crossover, and Carryover

In this chapter, we propose new integer optimization models for the lot-sizing and scheduling problem with sequence-dependent setups, based on GLSP. To incorporate setup crossover and carryover which are extensions of the problem frequently considered when dealing with real-world industrial problems, we first propose a standard model that straightforwardly adapts a formulation technique from the literature. Then, as the main contribution, we propose a novel optimization model that incorporates the notion of time flow. We derive a family of valid inequalities with which to compare the tightness of the models' LP relaxations. The proposed models are further tightened using the well-known reformulation techniques. In addition, to obtain feasible solutions quickly, we devise an LP-based heuristic algorithm. Computational experiments are conducted to test the performance of the proposed models and heuristic algorithm. The test results indicate that the newly proposed time-flow model has considerable advantages compared with the standard model in terms of tightness and solvability. It is also shown that the proposed heuristic algorithm can provide a feasible solution within a short computation time.

3.1 Introduction

As reviewed in Chapter 1, there have been many studies on LSP over recent decades, resulting in substantial advancements. However, as real-world manufacturing processes have a variety of unique characteristics and the features to be considered have been evolving, there remain outstanding issues not yet completely resolved, and indeed, active research on optimization models and solution approaches tailored to solve real problems continues.

One of the main sources of the uniqueness of certain manufacturing processes is their setup activity. Regarding the characteristics of the manufacturing process, the corresponding setup occurs in various aspects. For instance, in process industries, the setup takes a relatively long time due to preventive maintenance activities. In such cases, the setups can occur across consecutive time buckets, which situation is called *setup crossover*. However, the traditional big bucket models for LSP assume that a setup must be completed in a single time period. Therefore, in cases where the setup takes a considerable amount of time, the traditional optimization models cannot capture the situation properly and need modifications to allow setups to be crossed over. In addition, *setup carryover*, which indicates the preservation of the setup state across time buckets, should be allowed in order to avoid unnecessary setups.

As shown by Fiorotto, Huaccha Neyra, et al. (2020), if the model does not incorporate setup crossover and carryover, the quality of the established production schedule can deteriorate significantly, or a feasible schedule may not be found even if it exists. In this regard, there exist some previous studies which have discussed setup crossover and carryover and their consideration within various LSP models (Suerie,

2006; Belo-Filho et al., 2014; Fiorotto, Jans, et al., 2017). Particularly, there has been much effort devoted to the development of big bucket models that can consider various characteristics such as nontriangular sequence-dependent setup as well as setup crossover and carryover, which has resulted in more complex models with a larger number of variables and constraints (e.g. Menezes et al., 2011; Clark et al., 2014; Mahdieh et al., 2018).

As pointed out by Almeder & Almada-Lobo (2011), GLSP-based models enable accurate modeling, as they incorporate the two-level time structure. For these models, unlike big bucket models, consideration of sequence-dependent setup is straightforward. Because the sequence of the microperiods is fixed, the production sequence within each macroperiod is naturally obtained (Camargo et al., 2012). Moreover, setup carryover can be expressed in such a way that the setup states of the last microperiod in the previous macroperiod and the first microperiod in the subsequent macroperiod are identical.

The modeling framework of the GLSP also is beneficial when the setup times are relatively long, because the maximum number of items that can be produced within a macroperiod is small, which leads to a smaller number of microperiods. Therefore, to deal with the long and sequence-dependent setups in this study, we focused on GLSP-based models. However, to the best of our knowledge, there is a lack of studies addressing setup crossover for GLSP-based models. Moreover, as pointed out by Almada-Lobo, Clark, et al. (2015), introducing setup crossover into GLSP-based models is not straightforward.

The remainder of this chapter is organized as follows. In Section 3.2, we review the literature related to both the optimization model and the solution algorithm

that we consider. In Section 3.3, we present the optimization models and describe the differences among them by deriving a family of valid inequalities. In Section 3.4, we propose an LP-based heuristic algorithm. In Section 3.5, we present the results of computational experiments. In Section 3.6, we summarize the chapter and make concluding remarks.

3.2 Literature Review

GLSP modeling framework was first proposed by Fleischmann & Meyr (1997). As referred to in Copil et al. (2017), the GLSP can be regarded as a generalization of various models of LSP, because of its two-level time structure consisting of macroperiods of fixed length and microperiods of variable length. Meyr (2000) proposed an extension of the GLSP that considers a sequence-dependent setup, namely, GLSPST. In particular, as indicated by Camargo et al. (2012), the sequence-dependent setups can be incorporated into GLSP naturally because the sequence of microperiods is fixed. In addition, the heuristic algorithm of Fleischmann & Meyr (1997) was improved using a dual reoptimization technique by Meyr (2000). Meyr (2002) and Meyr & Mann (2013) considered the GLSPST with parallel machines and solved it with heuristic algorithms.

Recently, several studies have shown that the GLSPST can be extended to more general settings. Alem et al. (2018) considered the LSP with demand uncertainty and proposed a robust optimization model based on the GLSPST. Alipour et al. (2020) addressed the LSP with perishable products in the context of the food industry. They modeled the problem based on the GLSPST and proposed MIP-based heuristic algorithms. Guimarães et al. (2014) pointed out that the GLSPST can

be tightened using the well-known network flow reformulation presented by Wolsey (1997). Starting from the tightened GLSPST presented in Guimarães et al. (2014), we herein propose new GLSP-based models that can consider sequence-dependent setup, setup crossover, and carryover.

In fact, for consideration of setup crossover and carryover, various models have been proposed. They are based on either big or small bucket models. Suerie & Stadtler (2003) proposed a big bucket model that incorporates setup carryover. The authors then derived an extended formulation and presented families of valid inequalities. Later, this model was extended by Mohan et al. (2012) to further consider setup crossover. Suerie (2006) proposed two small bucket models that consider setup crossover.

Almada-Lobo, Klabjan, et al. (2007) presented big bucket models with both sequence-dependent setup and setup carryover. These models treat a production sequence in a macroperiod as a connected path or cycle without any subtours. Based on these results, Menezes et al. (2011) further developed models to incorporate nontriangular setups by identifying certain types of admissible subtours. Further, the authors devised a new big bucket model that considers setup carryover and crossover simultaneously. It was shown by Fiorotto, Jans, et al. (2017) that this model performs better than that proposed by Mohan et al. (2012). Therefore, we applied the formulation technique of Menezes et al. (2011) to the model of Guimarães et al. (2014), resulting in our first, standard model `(ST)`.

3.3 Integer Optimization Models

Let $\mathcal{I} = \{1, \ldots, I\}$, $\mathcal{T} = \{1, \ldots, T\}$, $\mathcal{S} = \{1, \ldots, S\}$ be the sets of items, macroperiods, and microperiods, respectively. Throughout the exposition, $i, j \in \mathcal{I}$, $t \in \mathcal{T}$, $s \in \mathcal{S}$ are used for indices. Let hc_{it}, bc_{it}, pc_{it}, and d_{it} denote the unit inventory holding, backlogging, production cost, and demand for each item i and macroperiod t, respectively. Also, let sc_{ijt} be the cost incurred when setup occurs from item i to item j in macroperiod t. The production capacity for macroperiod t, given in time units, is denoted by K_t. The unit production time of item i is a_i, while the time required for setup from item i to item j is st_{ij}. We also define st_{ii} for item i and let the values be zero to represent the situation wherein the setup state is carried over. We let $\mathcal{S}_t \subset \mathcal{S}$ be the set of microperiods that are contained in macroperiod t. The first and the last microperiods within the macroperiod t are denoted by f^t and l^t, respectively. In other words, $\mathcal{S}_t = \{f^t, f^t + 1, \ldots, l^t\}$. Moreover, for microperiod s, we define $T(s)$ as the macroperiod that contains s; that is, $T(s) = t$ if and only if $s \in \mathcal{S}_t$. Let $start_t$ and end_t be the start/end time of macroperiod t; that is, $start_t = \sum_{k=1}^{t-1} K_t$ and $end_t = \sum_{k=1}^{t} K_t$ for all $t \in \mathcal{T}$.

Next, we define the decision variables. Let I_{it} and B_{it} be the inventory level and backlog amount of item i at the end of macroperiod t. The initial inventory and backlog for item i are denoted by I_{i0} and B_{i0}, which are assumed to be zero. The variable x_{is} represents the production amount of item i in microperiod s. Binary variable y_{is} is equal to one if item i is produced in microperiod s. For $i, j \in \mathcal{I}$ and $s \in \mathcal{S} \setminus \{1\}$, binary variable z_{ijs} is equal to one if the setup from item i to j occurs from microperiod $s - 1$ to s. Variable z_{iis} represents the setup carryover of item i from microperiod $s - 1$ to s (see Meyr & Mann, 2013). For notational convenience,

Table 3.1: Nomenclature

Sets and Indices	
$\mathcal{I}$	Set of items which are indexed by i and j; $i, j \in \mathcal{I} = \{1, \ldots, I\}$
$\mathcal{T}$	Set of macroperiods which are indexed by t; $t \in \mathcal{T} = \{1, \ldots, T\}$
$\mathcal{S}$	Set of microperiods which are indexed by s; $s \in \mathcal{S} = \{1, \ldots, S\}$
$\mathcal{S}_t$	Set of microperiods contained in a macroperiod t
Parameters	
hc_{it}	Inventory holding cost of item i in macroperiod t
bc_{it}	Backlog penalty cost of item i in macroperiod t
pc_{it}	Production cost of item i in macroperiod t
sc_{ijt}	Cost incurred when setup occurs from item i to item j in macroperiod t
d_{it}	Demand of item i in macroperiod t
K_t	Production capacity of macroperiod t given in time unit
a_i	Production time per unit of item i
st_{ij}	Time required for setup from item i to j. $st_{ii} = 0$
$T(s)$	Macroperiod that contains microperiod s, i.e., $T(s) = t \Leftrightarrow s \in \mathcal{S}_t$
f^t/l^t	First/last microperiod of macroperiod t, i.e., $\mathcal{S}_t = \{f^t, f^t + 1, \ldots, l^t\}$
$start_t$	Start time of macroperiod t, i.e., $start_t = \sum_{k=1}^{t-1} K_k$
end_t	End time of macroperiod t, i.e., $end_t = \sum_{k=1}^{t} K_k$
Variables	
I_{it}	Inventory level of item i at the end of macroperiod t, $I_{i0} = 0$
B_{it}	Backlog amount of item i at the end of macroperiod t, $B_{i0} = 0$
x_{is}	Production amount of item i in microperiod s
y_{is}	$= 1$ if item i is produced in microperiod s; $y_{is} = 0$, otherwise
z_{ijs}	$= 1$ if setup from item i to j occurs from microperiod $s-1$ to s; $z_{ijs} = 0$, otherwise
q_{ijt}	$= 1$ if setup from item i to j crosses over from macroperiod $t-1$ to t; $q_{ijt} = 0$, otherwise; $q_{ij1} = q_{ijT+1} = 0$
v_{ijt}	Setup time split into macroperiod $t-1$, if $q_{ijt} = 1$; $v_{ijt} = 0$, otherwise; $v_{ij1} = v_{ijT+1} = 0$

we additionally define z_{ij1} for all $i, j \in \mathcal{I}$ and let their values be zero. For $i, j \in \mathcal{I}$ and $t \in \mathcal{T} \setminus \{1\}$, the binary variable q_{ijt} is equal to one if the setup from item i to item j crosses over from macroperiod $t-1$ to t. In this case, the setup time is split into $t-1$ and t. The continuous variable v_{ijt} represents the amount of the corresponding setup time distributed to $t-1$. It can be interpreted as the extra time borrowed from macroperiod $t-1$ to t for the setup crossover. Although there are no possibilities of setup crossover at the beginning of the first macroperiod and at the end of the last macroperiod, we define q_{ij1}, q_{itT+1}, v_{ij1}, and v_{ijT+1} for notational convenience and let their values be zero. All of the notations are summarized in Table 3.1.

3.3.1 Standard Model (ST)

We first provide the standard model (ST). This model is a generalization of the model presented by Guimarães et al. (2014) that can further consider setup crossover. After providing the model, we illustrate how setup crossover and carryover can be represented with the GLSP-based model. Later, this model is used to demonstrate the strength of our novel time-flow model (TF). The (ST) is as follows:

$$\text{minimize} \quad \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\Bigg(hc_{it}I_{it} + bc_{it}B_{it} + \sum_{s\in\mathcal{S}_t}\Big(pc_{it}x_{is} + \sum_{j\in\mathcal{I}} sc_{ijt}z_{ijs}\Big)\Bigg) \tag{3.1a}$$

$$\text{subject to} \quad I_{it} - B_{it} + d_{it} = I_{it-1} - B_{it-1} + \sum_{s\in\mathcal{S}_t} x_{is} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{3.1b}$$

$$\sum_{i\in\mathcal{I}}\sum_{s\in\mathcal{S}_t}\big(a_i x_{is} + \sum_{j\in\mathcal{I}} st_{ji}z_{jis}\big) \leq K_t + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{I}}\big(v_{ijt} - v_{ijt+1}\big) \qquad \forall t \in \mathcal{T} \tag{3.1c}$$

$$a_i x_{is} \leq K_t y_{is} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S}_t \tag{3.1d}$$

$$\sum_{i \in \mathcal{I}} y_{i1} = 1 \tag{3.1e}$$

$$y_{is} = \sum_{j \in \mathcal{I}} z_{jis} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \tag{3.1f}$$

$$y_{is} = \sum_{j \in \mathcal{I}} z_{ijs+1} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{S\} \tag{3.1g}$$

$$v_{ijt} \leq st_{ij} q_{ijt} \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T} \tag{3.1h}$$

$$q_{ijt} \leq z_{ijf^t} \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T} \tag{3.1i}$$

$$I_{it},\ B_{it},\ x_{is},\ y_{is},\ v_{ijt} \geq 0 \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S} \tag{3.1j}$$

$$q_{ijt}, z_{ijs} \in \{0, 1\} \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S} \tag{3.1k}$$

The objective function (3.1a) is the sum of the inventory holding, backlog penalty, production, and setup costs, which total is to be minimized. Constraints (3.1b) are balance equations between the inventory, backlog, demand, and production amounts. Constraints (3.1c) ensure that the sum of the production and setup times is less than or equal to the available capacity. The available capacity in t is computed in consideration of the time required for setup crossover. Constraints (3.1d) indicate that an item can be produced only if the corresponding setup occurs. Constraint (3.1e) represents the start of production in the first microperiod. Constraints (3.1f) and (3.1g) logically link the binary variables. They ensure that a setup from item i to j in microperiod s occurs if and only if i is produced in $s-1$ and j is produced in s. In other words, $z_{ijs} = 1$ if and only if $y_{is-1} = y_{js} = 1$. Constraints (3.1h) ensure that the setup time can be split only if the corresponding setup crossover occurs. The amount of split time is limited by the setup time. Constraints (3.1i) indicate that if the setup crossover occurs from $t-1$ to t, the item setup for the first microperiod of t can be determined. Constraints (3.1j) and (3.1k) ensure the domains of the

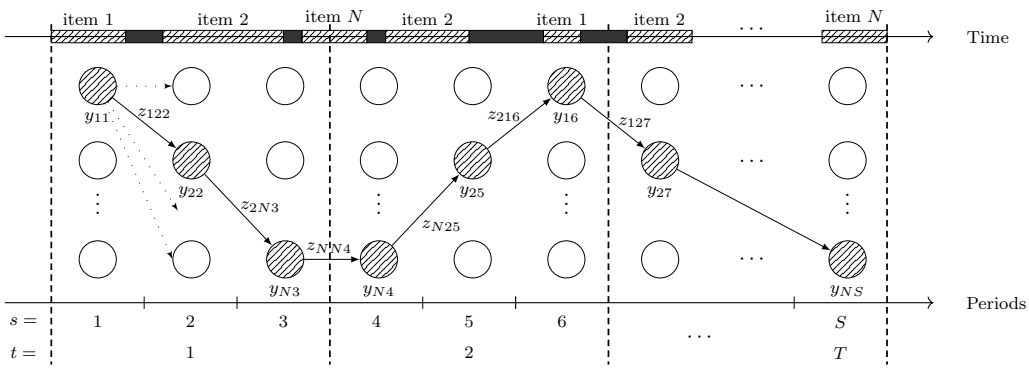


Figure 3.1: Illustration of the network flow corresponding to a production plan

variables. Note that the binary restriction of variable y is not necessary because it is implied by the constraints (3.1e) – (3.1g) and (3.1k).

With this model, a production plan can be represented as a network flow as shown in Figure 3.1. The bar shown in the upper part of Figure 3.1 represents a production plan wherein the productions and setups are indicated by the dashed and dark areas, respectively. This production plan is represented as a network flow in the lower part of Figure 3.1. The setup variables correspond to the arcs of the network, the flow balance equations of which correspond to the constraints (3.1e) – (3.1g). In this example, the setup state of item N is carried over from $s = 3$ to $s = 4$; that is, $z_{NN4} = 1$. Moreover, the setup between item 1 and item 2 is crossed over from $t = 2$ to $t = 3$; that is, $q_{123} = 1$.

3.3.2 Time-flow Model (TF)

We present the new time-flow model, which is denoted by (TF). For (TF), we introduce new variables and additional notations. Let us define a continuous variable w_{ijs} to represent the start time of the setup from item i to j in microperiod s, which can have a nonzero value only if $z_{ijs} = 1$. Note that, from the definition, $w_{ij1} = 0$

for all $i, j \in \mathcal{I}$. In addition, let r_{is} denote the idle time associated with item i in microperiod s. L_{ijs} and U_{ijs} denote the lower and upper bounds on the value of w_{ijs}, respectively. By adjusting these bounds, the model may or may not consider setup crossover, as shown later. The `(TF)` is as follows:

$$\text{minimize} \quad \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\Bigg(hc_{it}I_{it} + bc_{it}B_{it} + \sum_{s\in\mathcal{S}_t}\Big(pc_{it}x_{is} + \sum_{j\in\mathcal{I}} sc_{ijt}z_{ijs}\Big)\Bigg) \tag{3.2a}$$

$$\text{subject to} \quad I_{it} - B_{it} + d_{it} = I_{it-1} - B_{it-1} + \sum_{s\in\mathcal{S}_t} x_{is} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{3.2b}$$

$$\sum_{j\in\mathcal{I}}(w_{jis} + st_{ji}z_{jis}) + a_i x_{is} + r_{is} = \sum_{k\in\mathcal{I}} w_{iks+1} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \tag{3.2c}$$

$$a_i x_{is} + r_{is} \le K_t y_{is} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S}_t \tag{3.2d}$$

$$\sum_{i\in\mathcal{I}} y_{i1} = 1 \tag{3.2e}$$

$$y_{is} = \sum_{j\in\mathcal{I}} z_{jis} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \tag{3.2f}$$

$$y_{is} = \sum_{j\in\mathcal{I}} z_{ijs+1} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{S\} \tag{3.2g}$$

$$L_{ijs}z_{ijs} \le w_{ijs} \le U_{ijs}z_{ijs} \qquad \forall i, j \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \tag{3.2h}$$

$$I_{it},\ B_{it},\ x_{is},\ r_{is},\ y_{is},\ w_{ijs} \ge 0 \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S} \tag{3.2i}$$

$$z_{ijs} \in \{0, 1\} \qquad \forall i, j \in \mathcal{I}, s \in \mathcal{S} \tag{3.2j}$$

The objective function (3.2a) and other constraints (3.2b) and (3.2e) — (3.2g) are defined as in (3.1a), (3.1b), and (3.1e) – (3.1g) of `(ST)`, respectively. Constraints (3.2c) indicate that if an item i is set up in microperiod s from the previous item j ($z_{jis} = 1$), the sum of its start time (w_{jis}), setup time (st_{ji}), production time ($a_i x_{is}$), and idle time (r_{is}) is equal to the start time of the next setup to another

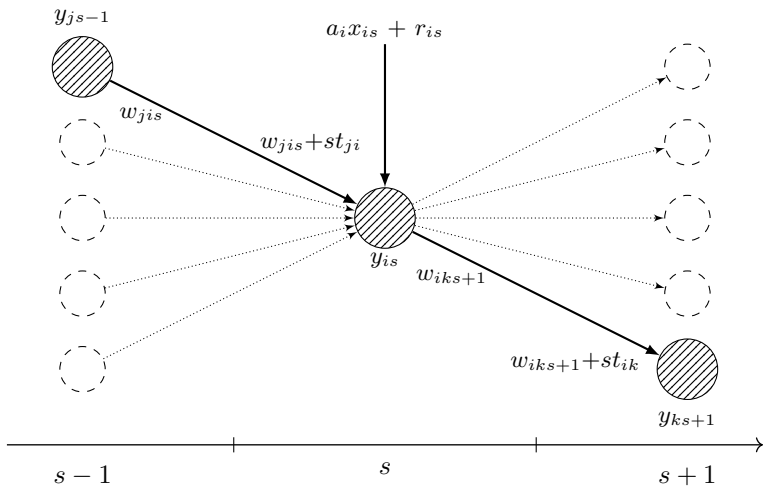


Figure 3.2: Illustration of the constraint (3.2c)

item k (w_{iks+1}). This relation is shown in Figure 3.2. As illustrated in the figure, constraints (3.2c) can be regarded as the time-flow balance equations. Constraints (3.2d) indicate that an item can be produced only if the corresponding setup occurs. Constraints (3.2h) ensure that w_{ijs} cannot have a nonzero value unless the corresponding setup occurs. If the corresponding setup occurs, its start time is bounded by L_{ijs} and U_{ijs}. Constraints (3.2i) and (3.2j) ensure the domains of the variables.

Now, we show how to set the values of L_{ijs} and U_{ijs}. If $s \in \mathcal{S} \setminus \{1\}$ and $s = f^{T(s)}$, the setup z_{ijs} can be crossed over. In this case, the setup start time w_{ijs} should be within the range $[start_{T(s-1)}, end_{T(s-1)}]$, while the setup end time $w_{ijs} + st_{ij}$ should be within the range $[start_{T(s)}, end_{T(s)}]$. Assuming that $st_{ij} \leq K_t$ for all $i, j \in \mathcal{I}$ and $t \in \mathcal{T}$, the following holds: $start_{T(s-1)} \leq start_{T(s)} - st_{ij}$ and $end_{T(s-1)} = start_{T(s)} \leq end_{T(s)} - st_{ij}$. Therefore, we can set $L_{ijs} = start_{T(s)} - st_{ij}$ and $U_{ijs} = start_{T(s)}$ for all $s \in \mathcal{S} \setminus \{1\}$ such that $s = f^{T(s)}$. On the contrary, when $s \neq f^{T(s)}$, there is no

chance of setup crossover as both $s-1$ and s belong to the same macroperiod. In this case, we can set $L_{ijs} = start_{T(s)}$ and $U_{ijs} = end_{T(s)} - st_{ij}$. In summary, by letting

$$L_{ijs} = \begin{cases} start_{T(s)} - st_{ij} & \text{if } s = f^{T(s)} \\ start_{T(s)} & \text{if } s \neq f^{T(s)} \end{cases} \qquad \forall i, j \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \tag{3.3a}$$

$$U_{ijs} = \begin{cases} start_{T(s)} & \text{if } s = f^{T(s)} \\ end_{T(s)} - st_{ij} & \text{if } s \neq f^{T(s)} \end{cases} \qquad \forall i, j \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \tag{3.3b}$$

the model can properly address setup crossover.

It is easy to modify both `(ST)` and `(TF)` so as not to allow setup crossover. For `(ST)`, we simply need to set $v_{ijt} = 0$ and $q_{ijt} = 0$. In this case, `(ST)` becomes equivalent to the model of Guimarães et al. (2014). For `(TF)`, it is sufficient to modify $L_{ijs} = U_{ijs} = start_{T(s)}$ for all $s = f^{T(s)}$ in equations (3.3), such that the first setup of a macroperiod starts at the very beginning of the macroperiod and there is no chance for setup crossover from the previous macroperiod.

Moreover, when the setup crossover is not allowed, it is possible to reduce the number of constraints in `(TF)` by slightly modifying the definition of the variable $\mathbf{w}$. Using the modified lower and upper bounds $L_{ijs} = U_{ijs} = start_{T(s)}$ for all the $s = f^{T(s)}$, one can define new variables w'_{ijs} in replacement of w_{ijs} which are defined as $w'_{ijs} = w_{ijs} - start_{T(s)} z_{ijs}$. The variable w'_{ijs} can be interpreted as the difference between the start time of the corresponding setup and the start time of the macroperiod $T(s)$, as illustrated in Figure 3.3. Then, by definition, $w'_{ijs} = 0$ for all $s = f^{T(s)}$ and constraints (3.2c) and (3.2h) are changed to (3.2c′) and (3.2h′),

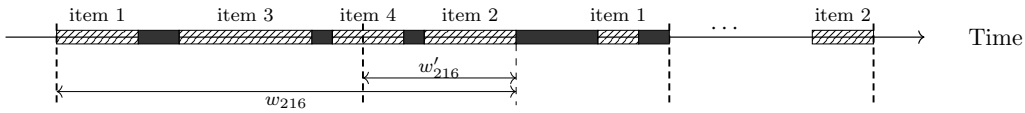


Figure 3.3: Definition of the variable w'_{ijs}

respectively, as follows:

$$\sum_{j\in\mathcal{I}}(w'_{jis} + st_{ji}z_{jis}) + a_i x_{is} + r_{is} = \begin{cases} \sum_{k\in\mathcal{I}} w'_{iks+1} & s \neq l^{T(s)} \\ \sum_{k\in\mathcal{I}} K_{T(s)} z_{iks+1} & s = l^{T(s)} \end{cases} \quad \forall i \in \mathcal{I}, s \in \mathcal{S} \quad (3.2c')$$

$$w'_{ijs} \leq \left(K_{T(s)} - st_{ij}\right) z_{ijs} \qquad \forall i, j \in \mathcal{I}, s \in \mathcal{S}, s \neq f^{T(s)} \qquad (3.2h')$$

With this simple modification of using variable $\mathbf{w}'$, the numbers of constraints and variables of `(TF)` can be reduced. Also, from the relation between the variables $\mathbf{w}'$ and $\mathbf{w}$, it is clear that, from any solution of one version, a solution of another version with the same objective value can be recovered.

In Guimarães et al. (2014), GLSP is classified as a product-oriented microperiod model. Therefore, following their classification scheme, `(ST)` and `(TF)` are also categorized as product-oriented microperiod models. In addition, Copil et al. (2017) classified GLSP as a generic model among various LSP models. Particularly, GLSP provides much flexibility in incorporating various extensions such as sequence-dependent or nontriangular setups as well as setup carryover, thanks to its two-level time structure. This is also true for `(ST)` and `(TF)` as they both are based on GLSP. Moreover,

our models can further consider setup crossover which allows for even more flexibility than the basic GLSP.

3.3.3 Comparison of (ST) and (TF)

In this section, we compare the tightness of the LP relaxations of (ST) and (TF). For comparison, let $\mathcal{P}^{\mathtt{ST}}$, $\mathcal{P}^{\mathtt{TF}}$ and $z^{\mathtt{ST}}_{\mathtt{LP}}$, $z^{\mathtt{TF}}_{\mathtt{LP}}$ be the sets of feasible solutions of the LP relaxations of (ST) and (TF) and their LP relaxation bounds, respectively.

Proposition 3.1. $z^{\mathtt{TF}}_{\mathtt{LP}} \geq z^{\mathtt{ST}}_{\mathtt{LP}}$.

Proof. We show that for any given solution $(\bar{\mathbf{I}}, \bar{\mathbf{B}}, \bar{\mathbf{x}}, \bar{\mathbf{r}}, \bar{\mathbf{y}}, \bar{\mathbf{w}}, \bar{\mathbf{z}}) \in \mathcal{P}^{\mathtt{TF}}$, the corresponding solution $(\mathbf{I}, \mathbf{B}, \mathbf{x}, \mathbf{y}, \mathbf{v}, \mathbf{q}, \mathbf{z}) \in \mathcal{P}^{\mathtt{ST}}$ with the same objective value can be constructed. Firstly, by letting $(\mathbf{I}, \mathbf{B}, \mathbf{x}, \mathbf{y}, \mathbf{z}) = (\bar{\mathbf{I}}, \bar{\mathbf{B}}, \bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}})$, the constraints that are common in both models are satisfied. In addition, we set $v_{ijt} = start_t \bar{z}_{ijf^t} - \bar{w}_{ijf^t}$ and $q_{ijt} = \bar{z}_{ijf^t}$, for all $i, j \in \mathcal{I}$ and $t \in \mathcal{T}$. Then, constraints (3.1i) clearly hold. Moreover, constraints (3.1h) are satisfied by the bound constraints (3.2h). For a given $t \in \mathcal{T}$, the following result can be obtained by summing constraints (3.2c) for all $i \in \mathcal{I}$ and $s \in \mathcal{S}_t$:

$$
\begin{aligned}
\sum_{i \in \mathcal{I}} \sum_{s \in \mathcal{S}_t} \Big(\sum_{j \in \mathcal{I}} & st_{ji} \bar{z}_{jis} + a_i \bar{x}_{is} + \bar{r}_{is} \Big) \\
&= \sum_{i,j \in \mathcal{I}} \big(\bar{w}_{ijf^{t+1}} - \bar{w}_{ijf^t} \big) \\
&= \sum_{i,j \in \mathcal{I}} \Big(\big(start_{t+1} \bar{z}_{ijf^{t+1}} - v_{ijt+1} \big) - \big(start_t \bar{z}_{ijf^t} - v_{ijt} \big) \Big) \\
&= start_{t+1} - start_t - \sum_{i,j \in \mathcal{I}} \big(v_{ijt+1} - v_{ijt} \big)
\end{aligned}
$$

$$= K_t - \sum_{i,j \in \mathcal{I}} \big(v_{ijt+1} - v_{ijt}\big).$$

The second equality holds by the relation between variables v_{ijt} and $\bar{w}_{ijft}$ constructed above, while the third holds as $\sum_{i,j \in \mathcal{I}} \bar{z}_{ijft} = 1$ for all $t \in \mathcal{T} \setminus \{1\}$ and $start_1 = 0$. Therefore, constraints (3.1c) of (ST) are also satisfied. Finally, it is clear that if constraints (3.2d) are satisfied, then constraints (3.1d) are satisfied. As a result, from any given solution in $\mathcal{P}^{\mathtt{TF}}$, the corresponding solution that satisfies the set of constraints defining $\mathcal{P}^{\mathtt{ST}}$ can be constructed. Moreover, because the objective functions are identical, their values are the same. □

The above proposition shows that the LP bound of (TF) is at least as strong as that of (ST), which means that $\mathcal{P}^{\mathtt{TF}}$ is at least as tight as $\mathcal{P}^{\mathtt{ST}}$. We then show that $\mathcal{P}^{\mathtt{TF}}$ can be tighter than $\mathcal{P}^{\mathtt{ST}}$. To this end, we derive a family of valid inequalities for (ST) using (TF). Let $\mathcal{X} := \{(i,s) : i \in \mathcal{I}, s \in \mathcal{S}\}$ be the set of pairs of an item and a microperiod that coincides with the nodes in Figure 3.1. Similarly, for a given macroperiod $t \in \mathcal{T}$, let $\mathcal{X}_t := \{(i,s) : i \in \mathcal{I}, s \in \mathcal{S}_t\}$. For ease of notation, we denote an arc $\big((i,s-1),(j,s)\big)$ as (i,j,s). For $X_1, X_2 \subseteq \mathcal{X}$, let $E(X_1,X_2)$ be the set of arcs (i,j,s) such that $(i,s-1) \in X_1$ and $(j,s) \in X_2$. In addition, for $X \subseteq \mathcal{X}$, let $E(X) := E(X,X)$, $\delta^+(X) := E(X,X^C)$, and $\delta^-(X) := E(X^C,X)$.

We firstly provide a generic form of the inequalities. For a given $X \in \mathcal{X}$, by summing the constraints (3.2c) over all $(i,s) \in X$ and eliminating the common terms of both sides, we obtain

$$\sum_{(i,s) \in X} \Big(a_i x_{is} + r_{is} + \sum_{j \in \mathcal{I}} st_{ji} z_{jis}\Big) = \sum_{(i,k,s+1) \in \delta^+(X)} w_{iks+1} - \sum_{(j,i,s) \in \delta^-(X)} w_{jis}.$$

From the bound constraints (3.2h), the following inequality can be attained:

$$\sum_{(i,s)\in X}\Big(a_i x_{is}+\sum_{j\in\mathcal{I}} st_{ji}z_{jis}\Big)\le \sum_{(i,k,s+1)\in\delta^+(X)} U_{iks+1}z_{iks+1}-\sum_{(j,i,s)\in\delta^-(X)} L_{jis}z_{jis}. \tag{3.4}$$

The right-hand side coefficients depend on the bounds $\mathbf{L}$ and $\mathbf{U}$ of the variable $\mathbf{w}$. With the bounds defined in (3.3), the following proposition can be derived.

Proposition 3.2. *For a given subset of nodes $X\subseteq\mathcal{X}$, the inequality*

$$\sum_{(i,s)\in X} a_i x_{is}+\sum_{(j,i,s)\in E(X)} st_{ji}z_{jis}+\sum_{\substack{(j,i,s)\in\delta^-(X):\\ s\neq f^{T(s)}}} st_{ji}z_{jis}+\sum_{\substack{(i,k,s+1)\in\delta^+(X):\\ s\neq l^{T(s)}}} st_{ik}z_{iks+1}$$
$$\le \sum_{(i,k,s+1)\in\delta^+(X)} end_{T(s)}z_{iks+1}-\sum_{(j,i,s)\in\delta^-(X)} start_{T(s)}z_{jis}\quad \forall X\subseteq\mathcal{X} \tag{3.5}$$

is valid for (ST). *Moreover, if we restrict X to be a subset of $\mathcal{X}_t$ for a given $t\in\mathcal{T}$, inequality* (3.5) *is presented in a more concise form:*

$$\sum_{(i,s)\in X} a_i x_{is}+\sum_{(j,i,s)\in E(X)} st_{ji}z_{jis}+\sum_{\substack{(j,i,s)\in\delta^-(X):\\ s\neq f^t}} st_{ji}z_{jis}+\sum_{\substack{(i,j,s+1)\in\delta^+(X):\\ s\neq l^t}} st_{ik}z_{iks+1}$$
$$\le K_t\sum_{(j,i,s)\in\delta^-(X)} z_{jis}\qquad \forall X\subseteq\mathcal{X}_t \tag{3.6}$$

Proof. From the bound (3.3), we have

$$\sum_{(i,s)\in X}\Big(a_i x_{is}+\sum_{j\in\mathcal{I}} st_{ji}z_{jis}\Big)\le \sum_{(i,k,s+1)\in\delta^+(X)} U_{iks+1}z_{iks+1}-\sum_{(j,i,s)\in\delta^-(X)} L_{jis}z_{jis}$$

$$= \sum_{\substack{(i,k,s+1)\in\delta^+(X):\\ s+1=f^{T(s+1)}}} start_{T(s+1)} z_{iks+1} + \sum_{\substack{(i,k,s+1)\in\delta^+(X):\\ s+1\neq f^{T(s+1)}}} (end_{T(s+1)} - st_{ik}) z_{iks+1}$$

$$- \sum_{\substack{(j,i,s)\in\delta^-(X):\\ s=f^{T(s)}}} (start_{T(s)} - st_{ji}) z_{jis} - \sum_{\substack{(j,i,s)\in\delta^-(X):\\ s\neq f^{T(s)}}} start_{T(s)} z_{jis}.$$

As $start_{T(s+1)} = end_{T(s)}$ when $s+1 = f^{T(s+1)}$ and $end_{T(s+1)} = end_{T(s)}$ when $s+1 \neq f^{T(s+1)}$, the right-hand side term is equal to

$$\sum_{(i,k,s+1)\in\delta^+(X)} end_{T(s)} z_{iks+1} - \sum_{(j,i,s)\in\delta^-(X)} start_{T(s)} z_{jis}$$

$$- \sum_{\substack{(i,k,s+1)\in\delta^+(X):\\ s+1\neq f^{T(s+1)}}} st_{ik} z_{iks+1} + \sum_{\substack{(j,i,s)\in\delta^-(X):\\ s=f^{T(s)}}} st_{ji} z_{jis}.$$

After the rearrangement of the terms, the inequalities (3.5) can be derived. In addition, when X is restricted to a subset of $\mathcal{X}_t$, $T(s) = t$. Moreover,

$$\sum_{(i,k,s+1)\in\delta^+(X)} z_{iks+1} = \sum_{(j,i,s)\in\delta^-(X)} z_{jis}.$$

Therefore, inequality (3.6) can be obtained. □

The meaning of inequality (3.6) is as follows: When $\sum_{(j,i,s)\in\delta^-(X)} z_{jis} = 0$, there are no incoming arcs to X; therefore, the nodes in X and the adjacent arcs cannot be visited. When $\sum_{(j,i,s)\in\delta^-(X)} z_{jis} \geq 1$, the right-hand side is greater than or equal to K_t. As the setup arcs that may cross over are not included in the summations of the left-hand side, the inequalities are satisfied owing to the capacity constraints. Next, we show that inequalities (3.5) are not implied in (ST), which means that $\mathcal{P}^{\mathrm{TF}}$

can be tighter than $\mathcal{P}^{\mathtt{ST}}$.

Proposition 3.3. *The inequalities* (3.5) *are not dominated by the constraints of* *(ST)*.

Proof. As inequalities (3.6) belong to a family of valid inequalities (3.5), it is sufficient to show that inequalities (3.6) can cut off some fractional solutions of (ST). We consider the fractional solution of (ST) illustrated in Figure 3.4. This solution is constructed as follows: For a given $t \in \mathcal{T}$, let $f := f^t$ and $l := l^t$. Then, for a given $i \in \mathcal{I}$, let $y_{is} = 1$ for $s \in \mathcal{S}$ such that $s \leq f$ or $s \geq l$. Furthermore, let $z_{iif+1} = z_{iif+2} = \cdots = z_{iil} = \frac{1}{K_t}$ and $z_{ikf+1} = z_{kkf+2} = \cdots = z_{kkl-1} = z_{kil} = \frac{K_t-1}{K_t}$ for any $k \in \mathcal{I}$. We can easily observe that the constructed partial solution is feasible for $\mathcal{P}^{\mathtt{ST}}$.

By letting $X = \{(i, f+1), (i, f+2)\}$, $\sum_{(j,i,s)\in\delta^-(X)} z_{jis} = \frac{1}{K_t}$, and therefore, the right-hand side of the corresponding inequality (3.6) is equal to one. Meanwhile, due to the setup constraints $a_i x_{if+1} \leq K_t y_{if+1}$ and $a_i x_{if+2} \leq K_t y_{if+2}$, we can let $x_{if+1} = x_{if+2} = \frac{1}{a_i}$. In this case, the left-hand side of inequality (3.6) is at

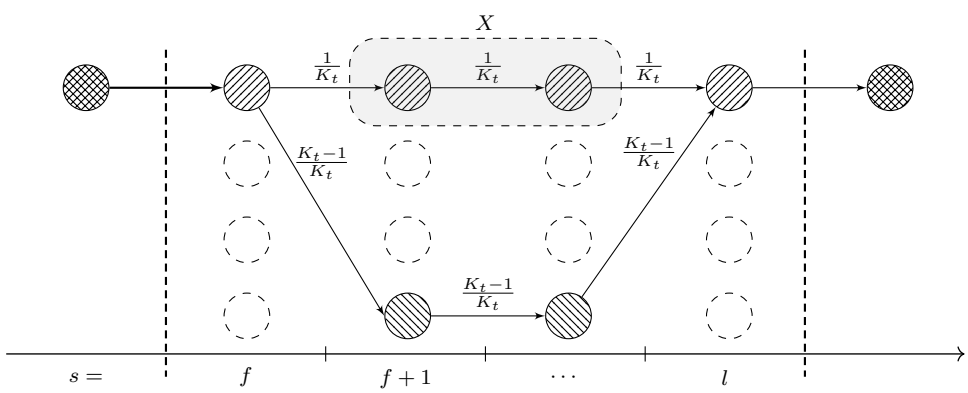


Figure 3.4: Illustration of the fractional solution of (ST)

least greater than two. Therefore, inequality (3.6) is violated, which means that the fractional solution is cut off by it. □

With a slight abuse of notation, we define $\mathcal{P}^{\mathtt{ST}+(3.5)}$ be the feasible set of the LP relaxation of (ST) after adding the family of inequalities (3.5). In addition, for $\mathtt{F} \in \{\mathtt{ST}, \mathtt{ST}+(3.5), \mathtt{TF}\}$, let $\mathcal{Q}^{\mathtt{F}} := proj_{(\mathbf{I},\mathbf{B},\mathbf{x},\mathbf{y},\mathbf{z})}\mathcal{P}^{\mathtt{F}}$; that is, the projection of $\mathcal{P}^{\mathtt{F}}$ onto the space of variables $(\mathbf{I}, \mathbf{B}, \mathbf{x}, \mathbf{y}, \mathbf{z})$. The above results yield the following corollary, which implies that $z^{\mathtt{TF}}_{\mathtt{LP}}$ can be greater than $z^{\mathtt{ST}}_{\mathtt{LP}}$.

Corollary 3.4. $\mathcal{Q}^{\mathtt{TF}} \subseteq \mathcal{Q}^{\mathtt{ST}+(3.5)} \subsetneq \mathcal{Q}^{\mathtt{ST}}$.

By restricting the problem settings, one can make the above inclusion relationship holds as equality. Particularly, we consider the situation where the setup crossover decisions are already fixed. In other words, we are given the fixed values of setup crossover variables $\bar{\mathbf{v}}$ and $\bar{\mathbf{q}}$ for (ST). Then, it is possible to set the bounds $\bar{\mathbf{L}}$ and $\bar{\mathbf{U}}$ in (TF) corresponding to $(\bar{\mathbf{v}}, \bar{\mathbf{q}})$. Specifically, by changing $\bar{L}_{ijs} = \bar{U}_{ijs} = start_{T(s)} - \bar{v}_{ijT(s)}$ for all $s = f^{T(s)}$, the same consequence is obtained. With these bounds, the generic inequality (3.4) is rewritten as follows:

$$\begin{aligned}
&\sum_{(i,s)\in X} a_i x_{is} + \sum_{(j,i,s)\in E(X)\cup\delta^-(X)} st_{ji} z_{jis} + \sum_{\substack{(i,k,s+1)\in\delta^+(X):\\ s\neq l^{T(s)}}} st_{ik} z_{iks+1} \\
&\qquad \leq \sum_{(i,k,s+1)\in\delta^+(X)} end_{T(s)} z_{iks+1} - \sum_{(j,i,s)\in\delta^-(x)} start_{T(s)} z_{jis} \\
&\quad + \sum_{\substack{(j,i,s)\in\delta^-(X):\\ s=f^{T(s)}}} \bar{v}_{jiT(s)} z_{jis} - \sum_{\substack{(i,k,s+1)\in\delta^+(X):\\ s=l^{T(s)}}} \bar{v}_{ikT(s+1)} z_{iks+1} \qquad \forall X \subseteq \mathcal{X} \quad (3.7)
\end{aligned}$$

For a given $(\bar{\mathbf{v}}, \bar{\mathbf{q}})$, let us define $\mathcal{P}^{\mathtt{ST}}(\bar{\mathbf{v}}, \bar{\mathbf{q}}) := \big\{(\mathbf{I}, \mathbf{B}, \mathbf{x}, \mathbf{y}, \mathbf{v}, \mathbf{q}, \mathbf{z}) \in \mathcal{P}^{\mathtt{ST}} \cap \{\mathbf{v} =$

$\bar{\mathbf{v}}, \mathbf{q} = \bar{\mathbf{q}}\}\}$, that is, the polyhedron defined by (ST) with fixed $(\bar{\mathbf{v}}, \bar{\mathbf{q}})$ values. The corresponding LP relaxation bound is denoted by $z_{\text{LP}}^{\text{ST}}(\bar{\mathbf{v}}, \bar{\mathbf{q}})$. Also, for the bounds $(\bar{\mathbf{L}}, \bar{\mathbf{U}})$ corresponding to $(\bar{\mathbf{v}}, \bar{\mathbf{q}})$, $\mathcal{P}^{\text{TF}}(\bar{\mathbf{L}}, \bar{\mathbf{U}})$ and $z_{\text{LP}}^{\text{TF}}(\bar{\mathbf{L}}, \bar{\mathbf{U}})$ can be defined similarly.

From Proposition 3.1, it is easy to observe that $z_{\text{LP}}^{\text{TF}}(\bar{\mathbf{L}}, \bar{\mathbf{U}}) \geq z_{\text{LP}}^{\text{ST}}(\bar{\mathbf{v}}, \bar{\mathbf{q}})$. We will show that after adding inequalities (3.7) to (ST), the same LP relaxation bound with (TF) can be obtained. We define $z_{\text{LP}}^{\text{ST}+(3.7)}(\bar{\mathbf{v}}, \bar{\mathbf{q}})$ be the LP relaxation bound of (ST) with the fixed $(\bar{\mathbf{v}}, \bar{\mathbf{q}})$ after adding the family of inequalities (3.7).

Proposition 3.5. *For a given fixed variables* $(\bar{\mathbf{v}}, \bar{\mathbf{q}})$ *of the formulation (ST) and the corresponding bounds* $(\bar{\mathbf{L}}, \bar{\mathbf{U}})$ *of the formulation (TF),*

$$z_{\text{LP}}^{\text{ST}+(3.7)}(\bar{\mathbf{v}}, \bar{\mathbf{q}}) = z_{\text{LP}}^{\text{TF}}(\bar{\mathbf{L}}, \bar{\mathbf{U}}).$$

Proof. By restricting X to be the subset of $\mathcal{X}_t$ for a given t for inequalities (3.7), we attain

$$\sum_{(i,s)\in X} a_i x_{is} + \sum_{(j,i,s)\in E(X)\cup\delta^-(X)} st_{ji} z_{jis} + \sum_{\substack{(i,k,s+1)\in\delta^+(X):\\ s\neq l^t}} st_{ik} z_{iks+1}$$
$$\leq K_t \sum_{(j,i,s)\in\delta^-(X)} z_{jis} + \sum_{\substack{(j,i,s)\in\delta^-(X):\\ s=f^t}} \bar{v}_{jit} z_{jis} - \sum_{\substack{(i,k,s+1)\in\delta^+(X):\\ s=l^t}} \bar{v}_{ikt+1} z_{iks+1} \quad (3.8)$$

for all $X \subseteq \mathcal{X}_t$. We firstly show that, although the inequalities (3.8) are the restricted version of the inequalities (3.7), they are sufficient to replace the inequalities (3.7), that is, $z_{\text{LP}}^{\text{ST}+(3.7)}(\bar{\mathbf{v}}, \bar{\mathbf{q}}) = z_{\text{LP}}^{\text{ST}+(3.8)}(\bar{\mathbf{v}}, \bar{\mathbf{q}})$. Without loss of generality, we consider a node set X such that $X = X_t \cup X_{t+1}$ where $X_t \subseteq \mathcal{X}_t$ and $X_{t+1} \subseteq \mathcal{X}_{t+1}$. We show that the sum of inequalities (3.8) defined by X_t and X_{t+1}, respectively, is identical

to the inequality (3.7) defined by X. The left-hand side of the sum of the two inequalities is as follows:

$$\sum_{(i,s)\in X_t\cup Q_{t+1}} a_i x_{is} + \sum_{\substack{(j,i,s)\in E(X_t)\cup\delta^-(X_t)\\ \cup E(X_{t+1})\cup\delta^-(X_{t+1})}} st_{ji} z_{jis}$$
$$+ \sum_{\substack{(i,k,s+1)\in\delta^+(X_t):\\ s\neq l^t}} st_{ik} z_{iks+1} + \sum_{\substack{(i,k,s+1)\in\delta^+(X_{t+1}):\\ s\neq l^{t+1}}} st_{ik} z_{iks+1}.$$

Note that

$$\begin{aligned}
&E(X_t)\cup\delta^-(X_t)\cup E(X_{t+1})\cup\delta^-(X_{t+1})\\
&\quad= E(X_t)\cup\delta^-(X_t)\cup E(X_{t+1})\cup E(X_t,X_{t+1})\cup E\big(\mathcal{X}\setminus X, X_{t+1}\big)\\
&\quad= E(X)\cup\delta^-(X_t)\cup E\big(\mathcal{X}\setminus X, X_{t+1}\big)\\
&\quad= E(X)\cup\delta^-(X).
\end{aligned}$$

Also, when s is not the last microperiod of $T(s)$, $\delta^+(X_t)\cup\delta^+(X_{t+1}) = \delta^+(X)$. Therefore, the left-hand side turns out to be the same as that of the inequality (3.7) defined by X. Regarding the right-hand side,

$$\begin{aligned}
&\sum_{(j,i,s)\in\delta^-(X_t)} K_t z_{jis} + \sum_{(j,i,s)\in\delta^-(X_{t+1})} K_{t+1} z_{jis}\\
&\quad= \sum_{(i,k,s+1)\in\delta^+(X_t)} end_t z_{iks+1} - \sum_{(j,i,s)\in\delta^-(X_t)} start_t z_{jis}\\
&\qquad+ \sum_{(i,k,s+1)\in\delta^+(X_{t+1})} end_{t+1} z_{iks+1} - \sum_{(j,i,s)\in\delta^-(X_{t+1})} start_{t+1} z_{jis}\\
&\quad= \sum_{(i,k,s+1)\in\delta^+(X)} end_{T(s)} z_{iks+1} - \sum_{(j,i,s)\in\delta^-(X)} start_{T(s)} z_{jis}
\end{aligned}$$

because $\sum_{(j,i,s)\in\delta^-(X_t)} z_{jis} = \sum_{(i,k,s+1)\in\delta^+(X_t)} z_{iks+1}$ and $start_{t+1} = end_t$. From this, the right-hand side is shown to be the same with that of the inequality (3.7) defined by X. As a result, we can conclude that the inequalities (3.8) are sufficient to replace the inequalities (3.7).

Next, we prove that $z_{\mathtt{LP}}^{\mathtt{ST}+(3.8)}(\bar{\mathbf{v}}, \bar{\mathbf{q}}) = z_{\mathtt{LP}}^{\mathtt{TF}}(\bar{\mathbf{L}}, \bar{\mathbf{U}})$ by showing that for any given fractional solution in $\mathcal{P}^{\mathtt{ST}+(3.8)}(\bar{\mathbf{v}}, \bar{\mathbf{q}})$, the corresponding assignment for the variable $\mathbf{w}$ that satisfies (3.2c) and (3.2h) always exists. To show this, we use Hoffman's circulation theorem which is stated as follows.

Lemma 3.6 (Hoffman (1976))**.** *Let $G = (V, E)$ be a directed graph and let $l, u : E \to \mathbb{R}^+$ satisfy $l(e) \le u(e)$ for all $e \in E$. Then there exists a feasible flow $f : E \to \mathbb{R}^+$ with $l(e) \le f(e) \le u(e)$ if and only if*

$$\sum_{e\in\delta^-(X)} u(e) \ge \sum_{e\in\delta^+(X)} l(e)$$

for all $X \subseteq V$.

For a macroperiod t, let us construct a digraph $G = (V, E)$ with $V = \mathcal{X}_t \cup \{0\}$ where 0 is a dummy node and $E = E_1 \cup E_2 \cup E_3 \cup E_4$. Given a fractional solution $(\bar{\mathbf{I}}, \bar{\mathbf{B}}, \bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}}, \bar{\mathbf{v}}, \bar{\mathbf{q}})$, the arc sets are defined as follows.

- $E_1 = \big\{\big(0, (i, f^t)\big) : (i, f^t) \in \mathcal{X}_t\big\}$
  with $l\big(0, (i, f^t)\big) = u\big(0, (i, f^t)\big) = \sum_{j\in\mathcal{I}}(start_t - \bar{v}_{jit} + st_{ji})\bar{z}_{jif^t} + a_i\bar{x}_{is}$

- $E_2 = \big\{\big(0, (i, s)\big) : (i, s) \in \mathcal{X}_t, s \ne f^t\big\}$
  with $l\big(0, (i, s)\big) = u\big(0, (i, s)\big) = \sum_{j\in\mathcal{I}} st_{ji}\bar{z}_{jis} + a_i\bar{x}_{is}$

- $E_3 = \big\{\big((i,l^t),0\big) : (i,l^t) \in \mathcal{X}_t\big\}$
  with $l\big((i,l^t),0\big) = u\big((i,l^t),0\big) = \sum_{k\in\mathcal{I}}(start_{t+1} - \bar{v}_{ikt+1})\bar{z}_{ikf^{t+1}}$

- $E_4 = \big\{(j,i,s) : (j,s-1),(i,s) \in \mathcal{X}_t\big\}$
  with $l(j,i,s) = start_t\bar{z}_{jis}$ and $u(j,i,s) = (end_t - st_{ji})\bar{z}_{jis}$.

From the construction of the graph, the question "For a given fractional solution in $\mathcal{P}^{\mathtt{ST}+(3.8)}(\bar{\mathbf{v}},\bar{\mathbf{q}})$, does feasible $\mathbf{w}$ that satisfies (3.2c) and (3.2h) exist?" is equivalent to "Does there exist a feasible flow f on G?". We show that there always exists a feasible flow using Lemma 3.6. We consider the following cases:

1. If $0 \in X$, we have

$$\begin{aligned}
\sum_{e\in\delta^-(X)} u(e) &= \sum_{(i,l^t)\in X^C}\sum_{j\in\mathcal{I}}\big(start_{t+1} - \bar{v}_{ijt+1}\big)\bar{z}_{ijf^{t+1}} \\
&\quad + \sum_{(j,i,s)\in E_4\cap\delta^-(X)} (end_t - st_{ji})\bar{z}_{jis}, \\
\sum_{e\in\delta^+(X)} l(e) &= \sum_{(i,f^t)\in X^C}\sum_{j\in\mathcal{I}}\big(start_t - \bar{v}_{jit}\big)\bar{z}_{jif^t} + \sum_{(i,s)\in X^C} a_i\bar{x}_{is} \\
&\quad + \sum_{j\in\mathcal{I}}\sum_{(i,s)\in X^c} st_{ji}\bar{z}_{jis} + \sum_{(j,i,s)\in E_4\cap\delta^+(X)} start_t\bar{z}_{jis}.
\end{aligned}$$

   Therefore,

$$\begin{aligned}
&\sum_{e\in\delta^-(X)} u(e) \geq \sum_{e\in\delta^+(X)} l(e) \\
(\Leftrightarrow)\quad &\sum_{(i,l^t)\in X^C}\sum_{j\in\mathcal{I}}\big(start_{t+1} - \bar{v}_{ijt+1}\big)\bar{z}_{ijf^{t+1}} + \sum_{(j,i,s)\in E_4\cap\delta^-(X)} (end_t - st_{ji})\bar{z}_{jis} \\
&\quad \geq \sum_{(i,f^t)\in X^C}\sum_{j\in\mathcal{I}}\big(start_t - \bar{v}_{jit}\big)\bar{z}_{jif^t} + \sum_{(i,s)\in X^C} a_i\bar{x}_{is}
\end{aligned}$$

$$+\sum_{j\in\mathcal{I}}\sum_{(i,s)\in X^c} st_{ji}\bar{z}_{jis} + \sum_{(j,i,s)\in E_4\cap\delta^+(X)} start_t\bar{z}_{jis}$$

$$(\Leftrightarrow)\quad \sum_{(j,i,s)\in\delta^+(X^C)} end_t\bar{z}_{jis} - \sum_{(j,i,s)\in\delta^-(X^C)} start_t\bar{z}_{jis}$$

$$+\sum_{(j,i,f^t)\in\delta^-(X^C)} \bar{v}_{jit}\bar{z}_{jif^t} - \sum_{(i,j,f^{t+1})\in\delta^+(X^C)} \bar{v}_{ijt+1}\bar{z}_{ijf^{t+1}}$$

$$\geq \sum_{(i,s)\in X^C} a_i\bar{x}_{is} + \sum_{(j,i,s)\in E(X^C)\cup\delta^-(X^C)} st_{ji}\bar{z}_{jis} + \sum_{\substack{(j,i,s+1)\in\delta^+(X^c):\\ s\neq l^t}} st_{ji}\bar{z}_{ji,s+1}$$

which coincides with the inequality (3.8) defined by X^C. Because the given solution satisfies the inequalities (3.8), the above inequality holds. As a result, there exists a feasible flow.

2. If $0 \notin X$, we have

$$\sum_{e\in\delta^-(X)} u(e) = \sum_{(i,f^t)\in X}\sum_{j\in\mathcal{I}} \big(start_t - \bar{v}_{jit}\big)\bar{z}_{jif^t} + \sum_{(i,s)\in X} a_i\bar{x}_{is}$$

$$+\sum_{j\in\mathcal{I}}\sum_{(i,s)\in X} st_{ji}\bar{z}_{jis} + \sum_{(j,i,s)\in E_4\cap\delta^-(X)} (end_t - st_{ji})\bar{z}_{jis},$$

$$\sum_{e\in\delta^+(X)} l(e) = \sum_{(i,l^t)\in X}\sum_{k\in\mathcal{I}} \big(start_{t+1} - \bar{v}_{ikt+1}\big)\bar{z}_{ikf^{t+1}} + \sum_{(j,i,s)\in E_4\cap\delta^+(X)} start_t\bar{z}_{jis}.$$

Therefore,

$$\sum_{e\in\delta^-(X)} u(e) \geq \sum_{e\in\delta^+(X)} l(e)$$

$$(\Leftrightarrow)\quad \sum_{(i,f^t)\in X}\sum_{j\in\mathcal{I}} \big(start_t - \bar{v}_{jit}\big)\bar{z}_{jif^t} + \sum_{(i,s)\in X} a_i\bar{x}_{is} + \sum_{j\in\mathcal{I}}\sum_{(i,s)\in X} st_{ji}\bar{z}_{jis}$$

$$+\sum_{(j,i,s)\in E_4\cap\delta^-(X)} (end_t - st_{ji})\bar{z}_{jis} - \sum_{(i,l^t)\in X}\sum_{k\in\mathcal{I}} \big(start_{t+1} - \bar{v}_{ikt+1}\big)\bar{z}_{ikf^{t+1}}$$

$$- \sum_{(j,i,s)\in E_4\cap\delta^+(X)} start_t \bar{z}_{jis} \geq 0. \tag{3.9}$$

We will show that the inequality (3.9) holds. Firstly,

$$\begin{aligned}
&\sum_{(j,i,f^t)\in\delta^-(X)} start_t \bar{z}_{jif^t} + \sum_{(j,i,s)\in E_4\cap\delta^-(X)} end_t \bar{z}_{jis} \\
&\qquad - \sum_{(i,l^t)\in X} \sum_{k\in\mathcal{I}} start_{t+1} \bar{z}_{ikf^{t+1}} - \sum_{(j,i,s)\in E_4\cap\delta^+(X)} start_t \bar{z}_{jis} \\
&= \Big(\sum_{(j,i,s)\in\delta^-(X)} end_t \bar{z}_{jis} - K_t \sum_{(j,i,f^t)\in\delta^-(X)} \bar{z}_{jif^t} \Big) \\
&\qquad - \Big(\sum_{(j,i,s)\in\delta^+(X)} end_t \bar{z}_{jis} - K_t \sum_{(j,i,s)\in E_4\cap\delta^+(X)} \bar{z}_{jis} \Big) \\
&= K_t \Big(\sum_{(j,i,s)\in E_4\cap\delta^+(X)} \bar{z}_{jis} - \sum_{(j,i,f^t)\in\delta^-(X)} \bar{z}_{jif^t} \Big) \\
&= K_t \Big(\sum_{(j,i,s)\in E_4\cap\delta^-(X^C)} \bar{z}_{jis} - (1 - \sum_{(j,i,f^t)\in\delta^-(X^C)} \bar{z}_{jif^t}) \Big) \\
&= K_t \Big(\sum_{(j,i,s)\in\delta^-(X^C)} \bar{z}_{jis} - 1 \Big).
\end{aligned}$$

In addition,

$$\begin{aligned}
&\sum_{(i,s)\in X} a_i \bar{x}_{is} + \sum_{j\in\mathcal{I}} \sum_{(i,s)\in X} st_{ji} \bar{z}_{jis} - \sum_{(j,i,s)\in E_4\cap\delta^-(X)} st_{ji} \bar{z}_{jis} \\
&\qquad = \sum_{(i,s)\in X} a_i \bar{x}_{is} + \sum_{(j,i,s)\in E(X)} st_{ji} \bar{z}_{jis} + \sum_{(j,i,f^t)\in\delta^-(X)} st_{ji} \bar{z}_{jis}.
\end{aligned}$$

Therefore, the inequality (3.9) is equivalent to

$$K_t \sum_{(j,i,s)\in\delta^-(X^C)} z_{jis} \geq K_t + \sum_{(j,i,f^t)\in\delta^-(X)} \bar{v}_{jit} z_{jif^t} - \sum_{(i,s)\in X} a_i x_{is}$$

$$
\begin{aligned}
&- \sum_{(i,k,f^{t+1})\in\delta^+(X)} \bar{v}_{ikt+1} z_{ikf^{t+1}} - \sum_{(j,i,s)\in E(X)} st_{ji} z_{jis} - \sum_{(j,i,f^t)\in\delta^-(X)} st_{ji} z_{jis} \\
\geq &\sum_{(i,k,f^{t+1})\in\delta^+(X^C)} \bar{v}_{ikt+1} z_{ikf^{t+1}} - \sum_{(j,i,f^t)\in\delta^-(X^C)} \bar{v}_{jit} z_{jif^t} \\
&+ \sum_{(i,s)\in X^C} a_i x_{is} + \sum_{(j,i,s)\in E(X^C)\cup\delta^-(X^C)} st_{ji} z_{jis} + \sum_{\substack{(j,i,s)\in\delta^+(X^C): \\ s\neq l^t}} st_{ji} z_{jis}
\end{aligned}
$$

where the second inequality holds due to the capacity constraints (3.1c). The above inequality coincides with the inequality (3.8) defined by X^C. As a result, it is shown that there exists a feasible flow.

□

This proposition shows that, for example, when the setup crossover is not allowed, `(ST)` with inequalities (3.7) gives the same LP bound with `(TF)`.

3.3.4 Facility Location Reformulation

After firstly being introduced by Krarup & Bilde (1977), the facility location (`FL`) reformulation technique has been widely used for various LSP models. It is part of the folklore, so to speak, that this reformulation significantly improves the quality of the LP bound; see, for example, Pochet & Wolsey (2006). Our models also can be further tightened by the `FL` reformulation. The basic idea is to use variables representing the fraction of demand for a specific period satisfied by the production in another period. Let α_{ist} be the variable representing the fraction of the demand of item i in macroperiod t satisfied by the production in microperiod s. As backlogging is allowed, the demand in t can be served from any microperiod s. In addition, we use a dummy microperiod $S+1$ to represent the demand that is not satisfied during

the entire planning horizon; for example, α_{iS+1t} represents the fraction of demand of item i in macroperiod t that is not satisfied. The following relations hold between the original variables $(\mathbf{x}, \mathbf{I}, \mathbf{B})$ and the new variable $\boldsymbol{\alpha}$.

$$
\begin{aligned}
x_{is} &= \sum_{t \in \mathcal{T}} d_{it} \alpha_{ist} && \forall i \in \mathcal{I}, s \in \mathcal{S}, \\
I_{it} &= \sum_{l=t+1}^{T} \sum_{s=1}^{l^t} d_{il} \alpha_{isl} && \forall i \in \mathcal{I}, t \in \mathcal{T}, \\
B_{it} &= \sum_{l=1}^{t} \sum_{s=f^{t+1}}^{S+1} d_{il} \alpha_{isl} && \forall i \in \mathcal{I}, t \in \mathcal{T}.
\end{aligned}
$$

Using the above relations, `(ST)` and `(TF)` can be reformulated as `(ST-FL)` and `(TF-FL)`, respectively. The following additional variables and notations are introduced for the reformulations `(ST-FL)` and `(TF-FL)`. We define a dummy microperiod $S+1$ to represent the demand that is not satisfied. Let $T(S+1) = T+1$ and $\mathcal{S}^0 = \mathcal{S} \cup \{S+1\}$. Then, we define variable α_{ist} as the fraction of demand of item i in macroperiod t satisfied by the production in microperiod s. If $s = S+1$, it represents the fraction of demand of item i in macroperiod t that is not satisfied.

Regarding the cost, let $H_{ist} = \sum_{k=T(s)}^{t-1} hc_{ik} d_{it}$ when $T(s) < t$; otherwise, $H_{ist} = 0$. Similarly, let $B_{ist} = \sum_{k=t}^{T(s)-1} bc_{ik} d_{it}$ when $T(s) > t$; otherwise, $B_{ist} = 0$. The cost coefficient of α_{ist} is defined as $C_{ist} := H_{ist} + B_{ist} + pc_{iT(s)} d_{it}$. The `(ST-FL)` is as follows:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \Big(\sum_{s \in \mathcal{S}^0} C_{ist} \alpha_{ist} + \sum_{j \in \mathcal{I}} \sum_{s \in \mathcal{S}_t} sc_{ijt} z_{ijs} \Big) && && \text{(3.10a)} \\
\text{subject to} \quad & \sum_{s \in \mathcal{S}^0} \alpha_{ist} = 1 && \forall i \in \mathcal{I}, t \in \mathcal{T} && \text{(3.10b)} \\
& \sum_{i \in \mathcal{I}} \sum_{s \in \mathcal{S}_t} \Big(\sum_{l \in \mathcal{T}} a_i d_{il} \alpha_{isl} + \sum_{j \in \mathcal{I}} st_{ij} z_{ijs} \Big)
\end{aligned}
$$

$$\leq K_t + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{I}}(v_{ijt} - v_{ijt+1}) \qquad \forall t \in \mathcal{T} \quad (3.10c)$$

$$\alpha_{ist} \leq y_{is} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S}, t \in \mathcal{T} \quad (3.10d)$$

$$\sum_{t\in\mathcal{T}} a_i d_{it} \alpha_{ist} \leq K_{T(s)} y_{is} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \quad (3.10e)$$

$$\sum_{i\in\mathcal{I}} y_{i1} = 1 \quad (3.10f)$$

$$y_{is} = \sum_{j\in\mathcal{I}} z_{jis} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \quad (3.10g)$$

$$y_{is} = \sum_{j\in\mathcal{I}} z_{ijs+1} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{S\} \quad (3.10h)$$

$$v_{ijt} \leq st_{ij} q_{ijt} \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T} \quad (3.10i)$$

$$q_{ijt} \leq z_{ijf^t} \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T} \quad (3.10j)$$

$$\alpha_{ist},\ y_{is},\ v_{ijt} \geq 0 \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S}^0 \quad (3.10k)$$

$$q_{ijt}, z_{ijs} \in \{0, 1\} \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S} \quad (3.10l)$$

The objective function (3.10a) is the total cost. Constraints (3.10b) are the demand constraints. Constraints (3.10c) are the capacity constraints. Constraints (3.10d) indicate that an item can be produced only if the corresponding setup occurs. Constraints (3.10e) represent the upper bound constraints for the amount of an item produced in a microperiod. Constraints (3.10f) – (3.10l) are defined in the same manner as the constraints (3.1e) – (3.1k) of `(ST)`, except for the domains of variable $\boldsymbol{\alpha}$. Similarly, the `(TF-FL)` is as follows:

$$\text{minimize} \quad \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\Big(\sum_{s\in\mathcal{S}^0} C_{ist}\alpha_{ist} + \sum_{j\in\mathcal{I}}\sum_{s\in\mathcal{S}_t} sc_{ijt} z_{ijs}\Big) \quad (3.11a)$$

$$\text{subject to} \quad \sum_{s\in\mathcal{S}^0} \alpha_{ist} = 1 \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \quad (3.11b)$$

$$\sum_{j\in\mathcal{I}}(w_{jis} + st_{ji} z_{jis}) + \sum_{l\in\mathcal{T}} a_i d_{il} \alpha_{isl} + r_{is}$$

$$= \sum_{k \in \mathcal{I}} w_{iks+1} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \quad (3.11c)$$

$$\alpha_{ist} \leq y_{is} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S}, t \in \mathcal{T} \quad (3.11d)$$

$$r_{is} + \sum_{t \in \mathcal{T}} a_i d_{it} \alpha_{ist} \leq K_{T(s)} y_{is} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \quad (3.11e)$$

$$\sum_{i \in \mathcal{I}} y_{i1} = 1 \quad (3.11f)$$

$$y_{is} = \sum_{j \in \mathcal{I}} z_{jis} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \quad (3.11g)$$

$$y_{is} = \sum_{j \in \mathcal{I}} z_{ijs+1} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{S\} \quad (3.11h)$$

$$L_{ijs} z_{ijs} \leq w_{ijs} \leq U_{ijs} z_{ijs} \qquad \forall i, j \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \quad (3.11i)$$

$$\alpha_{ist},\ y_{is},\ w_{ijs} \geq 0 \qquad \forall i \in \mathcal{I}, s \in \mathcal{S}^0, t \in \mathcal{T} \quad (3.11j)$$

$$z_{ijs} \in \{0, 1\} \qquad \forall i, j \in \mathcal{I}, s \in \mathcal{S} \quad (3.11k)$$

The objective function (3.11a) and other constraints (3.11b) and (3.11d) – (3.11k) are defined in the same manner as those for `(ST-FL)`. Constraints (3.11c) are the same as the time flow balance equations of `(TF)`, except that variable x_{is} is replaced with $\sum_{l \in \mathcal{T}} d_{il} \alpha_{isl}$.

3.4 LP-based Naive Fixing Heuristic Algorithm

In this section, we provide an LP-based naive fixing heuristic (LPNF) algorithm that is based on the iterative procedure of solving LP problems. This algorithm, also known as integer rounding heuristic or LP-rounding heuristic, has been popularly used in solving hard MIP problems including LSPs. For instance, Maes et al. (1991) proposed several variants of the heuristic and compared their performance for multi-level problems. Alfieri et al. (2002) applied the LP-rounding heuristic to several

LSP models and analyzed the differences. Denizel & Süral (2006) used the LP-rounding heuristic for the lot-sizing problem with setup times. The performance of the LP-based heuristic algorithms heavily relies on the tightness of the LP relaxation bounds. When the LP relaxation provides tighter bounds, the solution of the LP relaxation is more likely to be closer to the optimal solution. In this regard, we devise an LPNF algorithm to exploit the advantage of the tight bound obtained by the proposed time-flow models.

The LPNF algorithm for an instance $\mathcal{P}$ is described in Algorithm 3.1. In the

Algorithm 3.1: LPNF($\mathcal{P}$, γ^{up}, γ^{down}, δ, $\mathcal{Y}_0$, $\mathcal{Y}_1$)

$flag \leftarrow true$; // set the flag
$(obj, \mathbf{y}^{LP}) \leftarrow solveLP(\mathcal{P}, \mathcal{Y}_0, \mathcal{Y}_1)$ // get obj. value and sol. value of variable **y**
repeat
  if $\mathbf{y}^{LP}$ *all_binary* **then** // if all y variables are binary,
    $(UB^{LPNF}, \mathbf{y}^{LPNF}) \leftarrow (obj, \mathbf{y}^{LP})$; // return UB and feasible sol.
    break; // escape
  end
  forall $(i, s) \in (\mathcal{I} \times \mathcal{S}) \setminus \{\mathcal{Y}_0 \cup \mathcal{Y}_1\}$ **do**
    if $y_{is}^{LP} > \gamma^{up}$ **then** // fix to one
      $\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{(i, s)\}$;
      $flag \leftarrow false$;
    else if $y_{is}^{LP} < \gamma^{down}$ **then** // fix to zero
      $\mathcal{Y}_0 \leftarrow \mathcal{Y}_0 \cup \{(i, s)\}$;
      $flag \leftarrow false$;
    end
  end
  if *flag* **then** // if no variables are fixed in current iteration
    pick δ least fractional variables and put them in $\mathcal{Y}_0$ or $\mathcal{Y}_1$; // fix δ variables
  end
  $flag \leftarrow true$; // reset the flag
  $(obj, \mathbf{y}^{LP}) \leftarrow solveLP(\mathcal{P}, \mathcal{Y}_0, \mathcal{Y}_1)$;
until *false*;
return $(UB^{LPNF}, \mathbf{y}^{LPNF})$

algorithm, LP relaxation is recursively solved (lines 2, 21) with the subroutine $solveLP(\mathcal{P}, \mathcal{Y}_0, \mathcal{Y}_1)$ which provides the solution and the objective value of the LP relaxation of $\mathcal{P}$. The input parameters $\mathcal{Y}_0$ and $\mathcal{Y}_1$ are the index sets of variable $\mathbf{y}$, which are fixed to zero and one, respectively. After obtaining the fractional solution, variables whose values are greater than an upper threshold γ^{up} ($0.5 \leq \gamma^{up} < 1$) are fixed to one, while variables whose values are less than a lower threshold γ^{down} ($0 < \gamma^{down} < 0.5$) are fixed to zero (lines 8–16). This procedure is repeated until all binary variables are fixed to zero or one (lines 4–7). We only consider the binary condition of variable $\mathbf{y}$, because if it is fixed to binary values, the variable $\mathbf{z}$ is also restricted to binary values.

If all fractional values of the nonfixed binary variables are neither below the lower threshold nor above the upper threshold, during the iterations, the algorithm can run permanently. To avoid this, if there are no variables to be fixed in an iteration, the least fractional variables are forcibly fixed to zero or one, whichever is closer (lines 17–19). When applying this alternative fixing procedure, we considered the *relative* value of the variables, rather than the absolute value of them. To be more specific, when evaluating the value of variable y_{is}^{LP}, we take into account of the number of variables regarding the microperiod s which are already fixed to zero (when there is a variable which is fixed to one, other variables in microperiod s are fixed to zero automatically). Therefore, we use the following value to assess the relative value of the solution:

$$\widehat{y}_{is}^{LP} = y_{is}^{LP} \times (I - |\mathcal{Y}_{0s}|)$$

where $\mathcal{Y}_{0s}$ represents the subset of variables in microperiod s which are already fixed to zero. This alternative fixing procedure is illustrated in the following example.

Example 3.1. *Let us consider a fractional solution of the problem with five items illustrated in Figure 3.5. The variables already fixed to zero are represented by nodes with a dashed border, whereas those fixed to one are represented by nodes filled in gray. The slashed nodes represent the variables with fractional solution values.*

In microperiod 2, item 3 has the largest solution value (0.5). Because two variables are already fixed to zero, $\widehat{y}_{32}^{LP} = 0.5 \times 3 = 1.5$. *On the other hand, in microperiod 3, no variables are fixed yet. Item 1 has the largest solution value of* 0.4 *and the corresponding relative solution value is* $0.4 \times 5 = 2$. *Therefore, although the absolute solution value of variable* y_{32} *is greater than that of* y_{13}, *the variable* y_{13} *is fixed to one because it has a greater relative solution value.*

When considering the variables to be fixed to zero, the smallest relative solution value is obtained with $\widehat{y}_{52}^{LP} = 0.2 \times 3 = 0.6$ *although the absolute solution values of item 2 through 5 in microperiod 3 are 0.15, smaller than 0.2.*

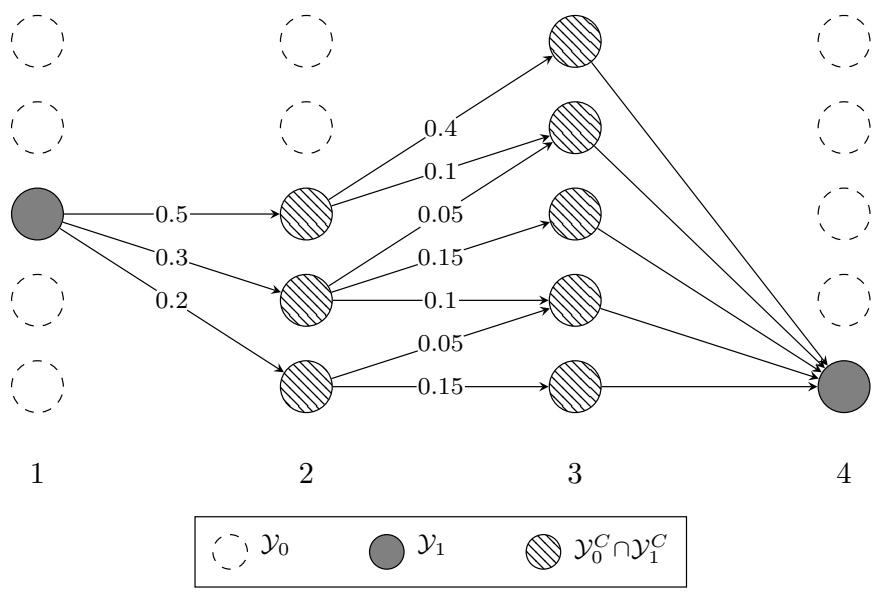


Figure 3.5: Illustration of the fixing procedure

The number of variables that can be forcibly fixed in one iteration is indicated by an integer parameter δ. Note that the LPNF algorithm for a given instance $\mathcal{P}$ can use any of the previously presented models as a base model: `(ST)`, `(TF)`, `(ST-FL)`, or `(TF-FL)`. We conducted experiments on the performance of the LPNF algorithm with different base models.

3.5 Computational Experiments

3.5.1 Test Instances

The set of instances is generated considering the characteristics of the flat-panel display manufacturing environment introduced in Lee & Lee (2020). In this manufacturing process, the setup takes a significant amount of time, and thus, it should not occur frequently in each macroperiod. In this regard, we set $|\mathcal{S}_t| = 3$ for all $t \in \mathcal{T}$. The length of each macroperiod is set to 100 (i.e., $K_t = K = 100$), and the unit production time a_i is set to 1 for all instances. The dimension of an instance is defined by the number of items (I) and the number of macroperiods (T). We use five dimensions: $(5, 10)$, $(5, 15)$, $(10, 10)$, $(10, 15)$, and $(15, 15)$.

In order to generate demand data similar to the demand patterns of this manufacturing environment, we use two parameters ρ and f, which denote the production capacity utilization level and the demand frequency, respectively. Specifically, $\rho = \frac{\sum_{i\in\mathcal{I},t\in\mathcal{T}} d_{it}}{K\cdot T}$ and $f = \frac{\sum_{i\in\mathcal{I},t\in\mathcal{T}} \mathbb{I}_{\{d_{it}>0\}}}{I\cdot T}$ where $\mathbb{I}_{\{d_{it}>0\}} = 1$ if nonzero demand occurs for item i in macroperiod t. From these parameters, the average amount of nonzero demand can be determined; that is, $d_{avg} = \frac{K\cdot\rho}{I\cdot f}$. Among the elements of the set $\mathcal{I}\times\mathcal{T}$, we randomly choose $f \cdot I \cdot T$ demand points while ensuring that at least one item has nonzero demand in the last macroperiod and that there is no item with zero demand

during the entire planning horizon. For each selected demand point, the amount of demand is generated from a discrete uniform distribution $[0.8 \cdot d_{avg}, 1.2 \cdot d_{avg}]$. We use the parameters $\rho \in \{0.8, 1\}$, $f \in \{1/3, 1/2\}$.

To determine the influence of the length of the setup times, we use three different classes: S, M, and L, whose (st^{min}, st^{max}) are $(0.05, 0.15)$, $(0.1, 0.3)$, and $(0.4, 0.6)$, respectively. Then, the setup times are generated from a discrete uniform distribution $DU[st^{min} \cdot K, st^{max} \cdot K]$. The inventory holding cost hc_{it} is set to 1, whereas the backlogging cost bc_{it} is generated from $DU[1, 5]$. The setup costs are set to be the same as the setup times; that is, $sc_{ijt} = st_{ij}$.

There are $5 \times 3 \times 4 = 60$ different combinations of factors, and we generate five instances for each combination, such that 300 instances are presented in total. The experimental results represented in this section are averaged over these five instances of a particular combination. The detailed test results are reported in Appendix C. All of our experiments were conducted on an Intel Core 3.10 GHz PC with 16 GB RAM under Windows 10 Pro. The proposed algorithms were implemented in C++. FICO Xpress 8.9 with its default parameter settings was used as the LP/MIP solver. We set the time limit for solving an MIP problem to 600 s.

3.5.2 LP Bound

We first present the test results regarding the strength of the LP relaxation bound in Figures 3.6 and 3.7 which present the *LP gap* and *Gap closed*, respectively. *LP gap* is computed as $\frac{(MIP\ Best\ Sol)-(LPB)}{MIP\ Best\ Sol} \times 100\%$, where *MIP Best Sol* is the best solution among those obtained by solving the four models using the MIP solver. *Gap Closed* of each model represents the ratio of the closed gap with respect to the *LPB* of `ST`;

that is, $\frac{(LPB \text{ of the model}) - (LPB \text{ of } \texttt{ST})}{(MIP\ Best\ Sol) - (LPB \text{ of } \texttt{ST})} \times 100\%$.

As shown in Figures 3.6 and 3.7, the `FL` reformulation has a significant impact on tightening the *LPB* for both `(ST)` and `(TF)`. The average LP gap of the models with the `FL` reformulation is about 65%, whereas that of the models without the `FL` reformulation is above 80%. From these results, we can see that, similarly to other LSP models, our models can benefit significantly from `FL` reformulation.

In addition, there is an obvious improvement in *LPB* if `(TF)` is used instead of `(ST)`. The *LP gap* is reduced by approximately 8% on average. This also holds between `(ST-FL)` and `(TF-FL)`. The *LPB* provided by `(TF-FL)` is tighter than that by `(ST-FL)`. In summary, `(TF-FL)` is the tightest model, and approximately 25% of the gap is closed compared to `(ST)`. The following relationship is shown between the

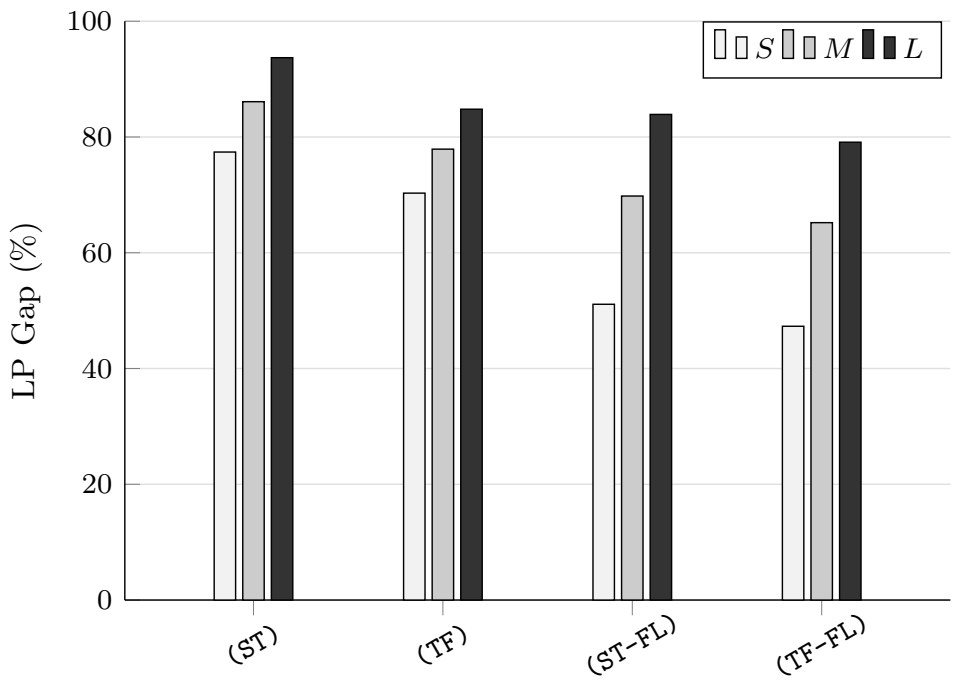


Figure 3.6: Comparison of the LP bound of the models: LP Gap (%)

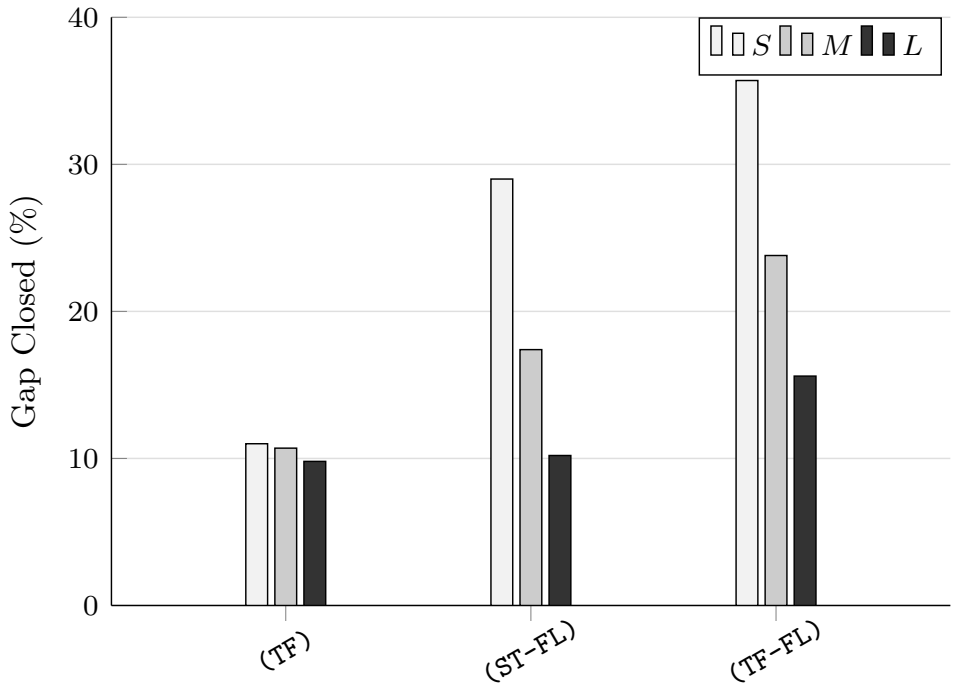


Figure 3.7: Comparison of the LP bound of the models: Gap Closed (%)

tightness of the models: (TF-FL) > (ST-FL) > (TF) > (ST). Detailed results of the tightness of the LP bound are presented in Table C.1 which summarizes the results for each factor, as indicated in the first column.

3.5.3 Computational Performance with MIP Solver

Now, we compare the solution quality and computational burden when the MIP solver is used. The test results are shown in Table 3.2 and Figure 3.8. Each row in Table 3.2 shows the summarized results. *Gap* indicates the final gap between the best solution and the best lower bound found by the solver within the time limit; for example, a zero gap indicates that the optimal solution is found. The ratio of the instances that found the optimal solution among all corresponding instances is

Table 3.2: Summary of the computational performance of the models

Model	ST	TF	ST-FL	TF-FL
Gap (%)	32.7	24.8	36.7	30.5
Opt (%)	29.3	39.0	24.7	32.7
Time (s)	459.7	410.7	488.2	445.6
# Node	292835	13536	71281	6691

represented in row *Opt*. The next rows *#Node* and *Time* represent the average number of nodes visited while running the branch-and-bound algorithm and the average computation time, respectively. The detailed test results are given in Tables C.2 and C.3 in Appendix C.

The main observation is that (TF) and (TF-FL) are more successful in solving problem instances than are (ST) and (ST-FL). The average gap of (ST) and (ST-FL)

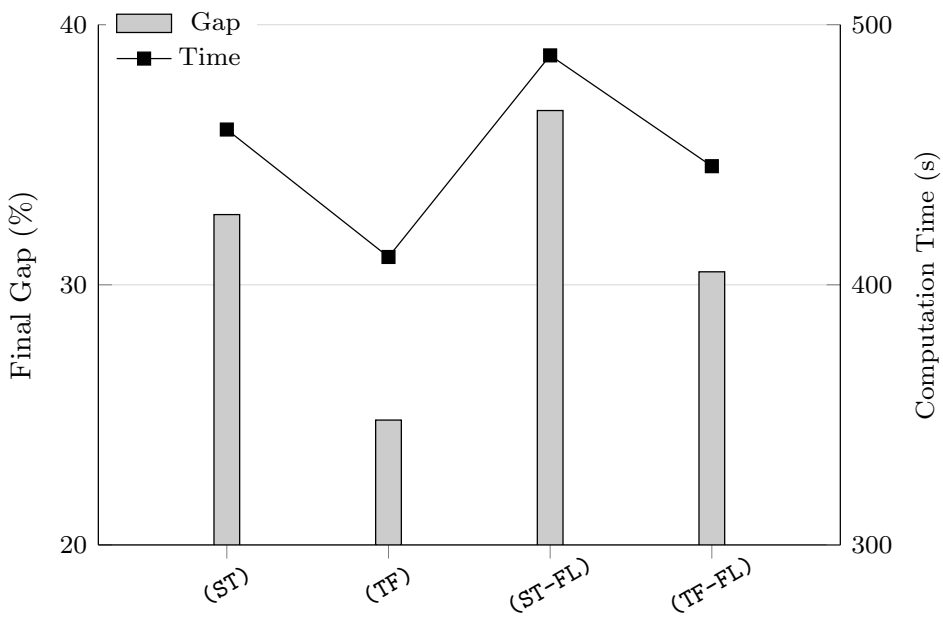


Figure 3.8: Comparison of the computational performance of the models

is approximately 35%, whereas that of (TF) and (TF-FL) is approximately 27%. Meanwhile, even if the FL reformulation is effective in reducing the LP gap, it does not seem to help much in improving the MIP solvability. With the reformulation, the final gap increases, and the number of optimal solutions found diminishes. This is due to the fact that the disadvantage of the increased model size is greater than the gain from the tighter LP bound. In conclusion, (TF) shows the best performance among the others.

The number of nodes visited by (TF) and (TF-FL) to close the gap is much smaller than those by (ST) and (ST-FL). Furthermore, they require less computation time. In summary, the proposed idea of the time-flow model seems quite useful. For the largest instances of dimensions $(15, 15)$, however, the performance is not satisfactory, because the average gap is above 50%. Therefore, we applied our solution approaches to those instances.

3.5.4 Performance of LPNF Algorithm

Lastly, we test the performance of the LPNF algorithm. Specifically, we investigate the effects of the tightness of the base model and the different parameter settings. We compare the different parameters $\gamma^{up} \in \{0.7, 0.9\}$ and $\delta \in \{1, 2, 3, 4, 5\}$, while γ^{down} is fixed to zero. Figure 3.9 presents the relative solution values and computation times where the relative solution value is defined as $\frac{\text{LPNF solution value}}{\textit{MIP Best Sol}} \times 100\%$. In Figure 3.9 we present the results of (ST-FL) and (TF-FL) for the sake of clarity. The detailed test results including the results of (ST) and (TF) are presented in Table C.4 of Appendix C.

As shown in Figure 3.9, there are significant differences in the algorithms' perfor-

mance with the different models. The quality of the solution obtained using (TF-FL) is much better than that obtained using the others. The average deviation from the *MIP Best Sol* is approximately 80% with (ST-FL), whereas it is about only 24% with (TF-FL). These results show that the tightness of the base model greatly affects the solution quality, which is natural, as the algorithm is highly dependent on the quality of the LP bound. Among all of the models, (TF-FL), the tightest, shows the best solution quality. The computation time for (TF-FL) is approximately twice that of (ST-FL), but is short enough to be used in practice. Regarding the parameters, the value of γ^{up} does not affect either the solution quality or the computation time

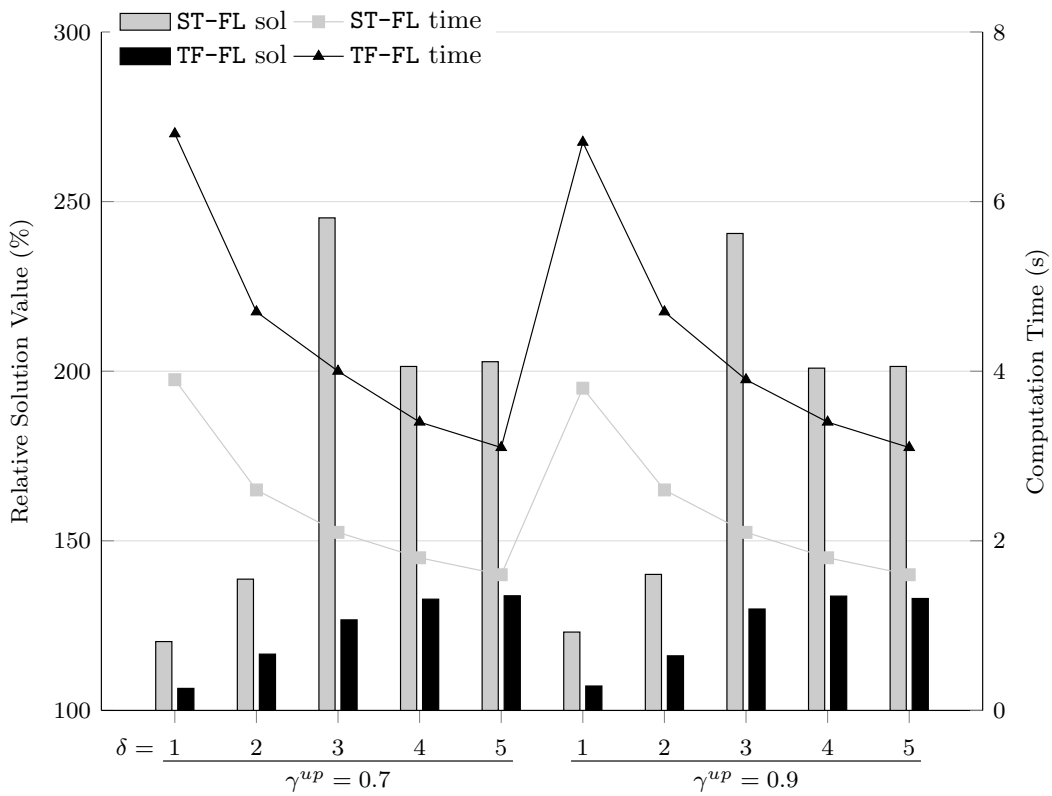


Figure 3.9: Test results for the LPNF algorithm

considerably. When the larger δ value is used, which means the more variables can be fixed in each iteration, the shorter computation time is required. On the other hand, the solution quality does not change significantly according to δ.

3.6 Summary

In this chapter, we introduced new integer optimization models for the LSP with sequence-dependent setups that can consider setup crossover and carryover. It was shown that the newly proposed time-flow models `(TF)` and `(TF-FL)` have certain benefits compared with the standard GLSP-based models in terms of the tightness of the LP bound and solvability with the MIP solver. Moreover, it was demonstrated that the tightness of the model not only affects the solvability with the MIP solver, but also has a significant impact on the performance of the proposed LP-based heuristic algorithm through the computational experiment results.

Chapter 4

Approximate Dynamic Programming Algorithm for Lot-sizing and Scheduling Problem with Sequence-dependent Setups

In this chapter, we devise an ADP algorithm for LSP with sequence-dependent setups to alleviate the drawback of traditional DP approaches, that is, the number of states can easily explode as the problem dimension increases. The proposed ADP algorithm estimates the value of states using their lower and upper bounds obtained by an optimization model. Specifically, the lower bound is obtained by the LP relaxation of the optimization model, whereas the upper bound is acquired from the LPNF algorithm provided in the previous chapter. Particularly, the time-flow model presented in Chapter 3 is employed as a base model. We conduct computational experiments to demonstrate the competitiveness of the proposed ADP algorithm. The ADP algorithm shows clear benefits over the standard MIP solver. Moreover, its performance is revealed to be competitive with a state-of-the-art big bucket model.

4.1 Introduction

4.1.1 Markov Decision Process

DP algorithms have been essential solution approaches for many optimization problems including LSP. To apply DP algorithms to typical optimization problems, they should be reformulated as dynamic programs which are also referred to as *Markov decision processes* (MDP). In this section, we firstly introduce MDP which provides a general modeling framework for the class of sequential decision making problems. Then, we briefly review DP-based solution approaches proposed for LSP. We adapt the general definition of MDP taken from textbooks by Puterman (2014) and Powell (2007) with some simplification to fit our problem. Readers are referred to Puterman (2014) and Powell (2007) for a comprehensive and detailed introduction of MDP.

MDP is defined with a given planning horizon which, in our discussion, is assumed to be finite and discrete with a set of periods $t = 1, \ldots, T$. At the beginning of each period, called as a *decision epoch* or simply a *stage*, decisions for the period should be made based on the current state. The current status of the MDP at stage t is represented by a state variable S_t. Based on the state information, we choose an action x_t which should be chosen from a set of feasible actions defined by the current state, that is, from $\mathcal{X}_t(S_t)$. The action x_t taken at the state S_t are evaluated by a contribution function $C_t(S_t, x_t)$. Depending on the context, the contribution function can be either the cost or profit. Because our problem is to minimize the total costs, the contribution function is interpreted as a cost that occurs when we choose a specific action given a specific state. After making a decision, the state transition from the current state to the next one is made according to a transition function

S^M, that is, $S_{t+1} = S^M(S_t, x_t)$. Because we do not consider the data uncertainty, the transition function is assumed to be deterministic.

The objective of the problem is to determine a sequence of optimal actions $\mathbf{x}^* = (x_1^*, \ldots, x_T^*)$ which minimizes the total costs that occur within the entire planning horizon, given an initial state S_1, that is,

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} \left\{ \sum_{t=1}^{T} C_t(S_t, x_t) \middle| S_1 \right\} \tag{4.1}$$

where $\mathcal{X} = \Pi_{t=1}^{T} \mathcal{X}_t(S_t)$. The optimal action at stage t, given the state S_t, can be written as

$$x_t^*(S_t) = \arg\min_{x_t\in\mathcal{X}_t(S_t)} \left(C_t(S_t, x_t) + \left\{ \sum_{t'=t+1}^{T} C_{t'}(S_{t'}, x_{t'}^*(S_{t'})) \middle| S_{t+1} \right\} \right) \tag{4.2}$$

where $S_{t+1} = S^M(S_t, x_t)$. In the above equation, the term inside the curly brackets forms another smaller problem. Defining a value function $V_t(S_t)$, the value of being at the state S_t, the recursion can be concisely represented as follows:

$$x_t^*(S_t) = \arg\min_{x_t\in\mathcal{X}_t} \Big(C_t(S_t, x_t) + V_{t+1}(S_{t+1}) \Big). \tag{4.3}$$

Therefore, the above equation can be written as a recursion of the value functions:

$$V_t(S_t) = \min_{x_t\in\mathcal{X}_t} \Big(C_t(S_t, x_t) + V_{t+1}(S_{t+1}) \Big). \tag{4.4}$$

The above equation (4.4) is well-known as the Bellman equation (Bellman, 1957). Bellman (1957) introduced the concept of the *principle of optimality* which enables

the optimal solution of the problem to be obtained by solving smaller subproblems recursively. In this regard, the class of solution approaches based on the forward or backward recursion procedure using the principle of optimality is called a DP approach. After the introduction of DP approaches, numerous optimization problems from vast research areas including optimal control theory, operations research, and sequential decision making have been tackled by DP approaches. See Bertsekas (2012) for detail.

Like other optimization problems, the LSP can be reformulated as MDP and solved via DP algorithm. In fact, DP algorithms have been essential solution approaches for many production planning problems including lot-sizing and scheduling problems. Since first being addressed by Wagner & Whitin (1958), many DP algorithms have been proposed to solve lot-sizing problems efficiently (e.g., Zangwill, 1966; Florian et al., 1980; Federgruen & Tzur, 1991; Wagelmans et al., 1992; Aggarwal & Park, 1993; Van Hoesel & Wagelmans, 1996, to name a few influential early works). For a well-organized summary of the various algorithms for the class of LSP, including DP, readers are referred to Pochet & Wolsey (2006).

In addition to the standalone use of DP algorithms, they are often combined with other optimization approaches or heuristic algorithms to solve LSP. For example, Hartman et al. (2010) proposed valid inequalities for single-item capacitated lot-sizing problem derived from a truncated forward DP algorithm. The proposed valid inequalities are used as cutting planes in a branch-and-bound algorithm. Subsequently, Büyüktahtakın, Smith, et al. (2018) proposed valid inequalities for multi-item capacitated lot-sizing problem which are derived by a partial DP algorithm. The inequalities were strengthened by a lifting procedure and used as cutting planes.

Recently, Cunha & Melo (2021) proposed a DP-based heuristic algorithm to solve multi-level uncapacitated lot-sizing problem which uses a multi-start randomized procedure to obtain feasible solutions quickly.

4.1.2 Approximate Dynamic Programming Algorithms

Except for certain special cases such as the uncapacitated or single-item problem, however, most real-world LSPs are difficult to solve via DP algorithms, because they often have a prohibitively large number of states and actions which increases exponentially with the problem instance size. Particularly, if there exist some practical constraints or characteristics such as sequence-dependent setups, the dimension of the states also can drastically increase. Therefore, obtaining an optimal solution for practical large instances using exact DP approaches seems unpromising.

In order to avoid this issue, known as the "curse of dimensionality'' (Powell, 2007), ADP algorithms have been widely employed. Most of the ADP approaches often use a policy, which is defined as "a rule (or function) that determines a feasible decision given the available information in a state" (Powell, 2019), based on a specific approximation strategy. There is a great deal of freedom in designing the policy, and a great variety of options exist. Powell categorized various approximation strategies in the ADP framework into four broad classes: *policy function approximation*, *cost function approximation*, *direct lookahead approximation*, and *value function approximation*.

In this study, among the four strategies, we adapt value function approximation to devise an ADP algorithm for LSP with sequence-dependent setups. Using the value function approximation strategy, the true value functions of states are re-

placed with the approximated value functions. Therefore, the equation (4.3) can be rewritten as follows:

$$x_t^{ADP}(S_t) = \arg\min_{x_t \in \mathcal{X}_t} \left(C_t(S_t, x_t) + \widehat{V}_{t+1}(S_{t+1}) \right). \tag{4.5}$$

Then the corresponding policy is to choose actions greedily with respect to the approximated value function without the recursive evaluation of an exponential number of future states.

There are many possibilities in designing the approximated value function $\widehat{V}_t(S_t)$. One of the widely used methods is to assume specific function structures. When determining the function structure, one should consider the trade-off between simplicity and expressivity. As the function structure gets simpler, it becomes easier to handle, whereas it is less likely to capture the structure of the true value function. One of the simplest structures is a linear function. The function $\widehat{V}_t(S_t)$ is often assumed as a linear function of the state variable S_t with the corresponding coefficient vector v_t, i.e., $\widehat{V}_t(S_t) = v_t S_t$. Because of its simplicity, linearly approximated functions can be naturally incorporated into large optimization problems. However, it cannot provide a good approximation of the true value due to its simplicity except for some special cases.

Alternatively, one can adapt piecewise-linear value function structures which have much more expressivity. Especially, when convexity or concavity can be assumed, the value function also can be incorporated into optimization problems easily. There are types of problems where the convexity or concavity assumption is natural and plausible such as dynamic resource allocation problems. For instance, Topaloglu

& Powell (2006) addressed stochastic time-staged integer multi-commodity flow problems and proposed ADP algorithms. They proposed three different approximation strategies: linear, concave piecewise-linear, and hybrid of them. Using the characteristics of the network flow problem, they provide efficient estimation and update procedures for function coefficients. Toriello et al. (2010) provided an ADP algorithm for deterministic inventory routing problems. The authors used a separable piecewise-linear concave function to approximate the value function. They provided a fitting procedure that preserves the concavity of the function. Papageorgiou et al. (2015) proposed an ADP algorithm for maritime inventory routing problems with a long planning horizon. They also used a piecewise-linear value function, but did not necessarily assume concavity or convexity.

Alternatively, there exist value function approximation approaches which directly estimate the value without any assumption on the structure. For instance, to solve multi-dimensional knapsack problems, Bertsimas & Demir (2002) used both the primal and dual bounds of the true state value, which were obtained by simple heuristic algorithms and LP relaxation, respectively. In addition, the authors proposed ADP algorithms with parametric and nonparametric approximations. See Powell (2007) and Powell (2016) for several other successful applications of ADP.

There exists some research where the value function approximation is applied to solve LSPs. Büyüktahtakın & Liu (2016) proposed various ADP algorithms for a single-item capacitated lot-sizing problem. Using the characteristics of the inventory cost function, they devised a direct-connection algorithm and a slope-check algorithm based on the sampling of the states. To apply the value function approximation approach proposed by Büyüktahtakın & Liu (2016) to the problem with

multiple items, however, one should assume the separability of the value function, that is, the value function of the problem can be represented as the sum of value functions defined for each item. However, there exist interactions between the items. For instance, multiple items compete for the restricted capacity to be produced which implies that the true value function is nonseparable. In addition, when the setups occur in a sequence-dependent manner, the inter-dependency between the items further increases and cannot be neglected. Therefore, the ADP algorithm proposed in Büyüktahtakın & Liu (2016) is hard to be applied to our problem. Instead, we present an ADP algorithm that is similar to the approach presented by Bertsimas & Demir (2002) who made use of the bound information.

The remainder of this chapter is organized as follows. In Section 4.2, we reformulate our problem as an MDP. In Section 4.3, we present our ADP algorithm and the value function approximation procedure. In Section 4.4, we present the results of computational experiments. In Section 4.5, we summarize the chapter and make concluding remarks.

4.2 Markov Decision Process Reformulation

Before presenting the ADP algorithm, we first present our problem as an MDP based on the optimization model addressed in Chapter 3. We regard microperiods $s = 1, \ldots, S$ as stages. The state of the system at stage s, denoted by R_s, is defined as a sequence of items that are set up to microperiod s. Specifically, let $R_s = (i_1, \ldots, i_s)$, where i_k denotes an index of the item that is set up in microperiod k. Thus, the state space at stage s is defined as $\mathcal{R}_s = \{(i_1, \ldots, i_s) | \, i_k \in \mathcal{I}, \, \forall k = 1, \ldots, s\}$. The set of possible states at stage s, given a previous state $\bar{R}_{s-1} = (\bar{i}_1, \ldots, \bar{i}_{s-1}) \in \mathcal{R}_{s-1}$, is

defined as

$$\mathcal{R}_s(\bar{R}_{s-1}) = \{(\bar{i}_1, \ldots, \bar{i}_{s-1}, i) |\, i \in \mathcal{I}\} = \{(\bar{R}_{s-1}, i) |\, i \in \mathcal{I}\}.$$

This state definition is different from those commonly used in other studies where the states are defined by inventory level or cumulative production amounts (Florian et al., 1980; Hartman et al., 2010; Büyüktahtakın & Liu, 2016).

Let us define the value function $V_s(R_s)$ as the minimum total cost, except for the setup cost from 1 to s, under a given fixed sequence R_s. Then, the value function of the last stage S for a given R_S, $V_S(R_S)$, can be easily computed by solving an LP, because the entire setup sequence is already determined. The problem is then to calculate the value function $V_0(R_0)$, where R_0 is a null sequence in which nothing is fixed. When a state R_{s-1} transitions to R_s, the corresponding state transition cost occurs. Specifically, if $R_{s-1} = (i_1, \ldots, i_{s-1})$ and $R_s = (i_1, \ldots, i_{s-1}, i_s)$, the transition cost $C_s(R_{s-1}, R_s)$ is the setup cost between i_{s-1} and i_s; that is, $sc_{i_{s-1},i_s,T(s)}$. We assume that $C_1(R_0, \cdot) = 0$. With this definition, the DP recursion can be expressed as

$$\begin{aligned} V_{s-1}(R_{s-1}) &= \min_{R_s \in \mathcal{R}_s(R_{s-1})} \big\{C_s(R_{s-1}, R_s) + V_s(R_s) |\, R_{s-1} = (i_1, \ldots, i_{s-1})\big\} \\ &= \min_{i \in \mathcal{I}} \big\{sc_{i_{s-1},i,T(s)} + V_s(i_1, \ldots, i_{s-1}, i)\big\}, \quad (4.6) \end{aligned}$$

for $s = 1, \ldots, S$. The last stage value function $V_S(\cdot)$ is easy to compute, as previously mentioned. Starting with $s = S$, $V_0(R_0)$ can be calculated using the backward recursion of equation (4.6). Subsequently, the optimal production sequence can be

identified using

$$R_s^* = (R_{s-1}^*, i_s^*) = (i_1^*, \ldots, i_s^*) \text{ where } i_s^* = \arg\min_{i \in \mathcal{I}} \{sc_{i_{s-1}^*, i, T(s)} + V_s(i_1^*, \ldots, i_{s-1}^*, i)\}$$

for $s = 1, \ldots, S$ and $R_0^* = R_0$.

The DP recursion is illustrated in Figure 4.1. As shown in the figure, however, the number of total states is $\mathcal{O}(I^S)$ which is exponentially large. Therefore, the number of states easily explodes when the number of items and periods increases. Therefore, it is not practical to solve the DP recursion exactly. In the following section, we present an ADP algorithm to avoid the issue of the curse of dimensionality.

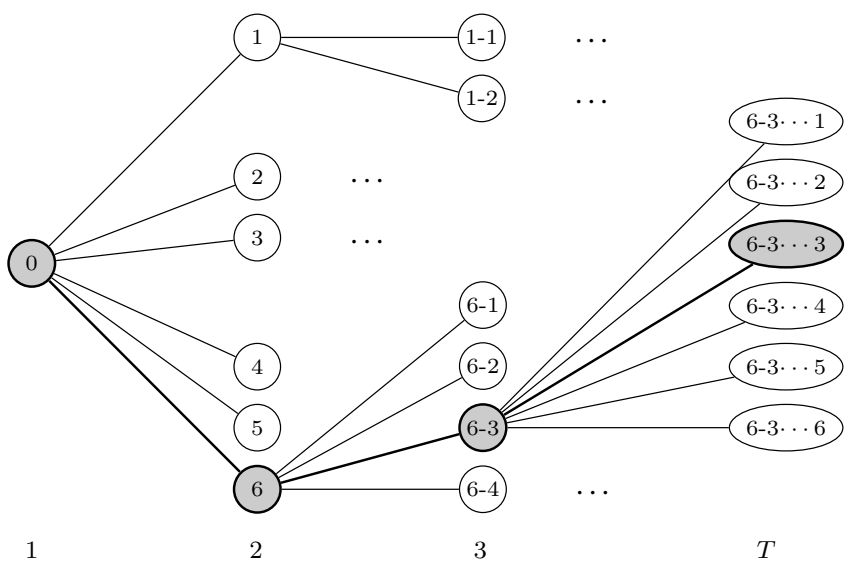


Figure 4.1: Illustration of the dynamic programming recursion

4.3 Approximate Dynamic Programming Algorithm

In our ADP algorithm, we use the approximated value function $\widehat{V}_s(R_s)$, which approximates $C_s(R_{s-1}, R_s) + V_s(R_s)$. Then, the recursion (4.6) is modified as

$$\widetilde{V}_{s-1}(R_{s-1}) = \min_{R_s \in \mathcal{R}_s(R_{s-1})} \left\{ \widehat{V}_s(R_s) |\, R_{s-1} \right\} \tag{4.7}$$

where $\widetilde{V}_{s-1}(R_{s-1})$ is an estimate of the value of being at R_{s-1}. The approximated value function $\widehat{V}_s(R_s)$ is constructed using both the lower bound $LB(R_s)$ and the upper bound $UB(R_s)$ of the true value of $C_s(R_{s-1}, R_s) + V_s(R_s)$. There are many different ways to obtain these bounds. One can expect that if tighter $LB(R_s)$ and $UB(R_s)$ are used, the approximation will be more accurate and the solution quality will be improved. In addition, as these bounds need to be obtained repetitively in the ADP algorithm, it is important to do so in a short time. Considering this trade-off, acquiring good bounds in a short time is a key aspect of the ADP algorithm. In this regard, to obtain $LB(R_s)$, we solve the LP relaxation of the problem with the partially fixed production sequence $R_s = (i_1, \ldots, i_s)$; that is, $y_{i_k k} = 1$ for all $1 \leq k \leq s$. To obtain $UB(R_s)$ in the ADP algorithm, we use LPNF algorithm which is proposed in Chapter 3 as a subroutine.

With $LB(R_s)$ and $UB(R_s)$, calculated by the LP relaxation and the LPNF algorithm, respectively, we evaluate $\widehat{V}_s(R_s)$ using the equation

$$\widehat{V}_s(R_s) = (1 - \varepsilon) \cdot LB(R_s) + \varepsilon \cdot UB(R_s)$$

where the parameter ε indicates the ratio of the upper bound when estimating the

value of state ($0 \leq \varepsilon \leq 1$). If we set $\varepsilon = 1$ ($\varepsilon = 0$), the value is approximated using only the upper bound (lower bound). Using this approximation, the production sequence can be identified using

$$\widehat{R}_s = (\widehat{R}_{s-1}, \widehat{i}_s) = (\widehat{i}_1, \ldots, \widehat{i}_s) \text{ where } \widehat{i}_s = \arg\min_{i \in \mathcal{I}} \{\widehat{V}_s(\widehat{i}_1, \ldots, \widehat{i}_{s-1}, i)\}$$

for $s = 1, \ldots, S$ and $\widehat{R}_0 = R_0$. Moreover, given a state, it is possible to consider only some promising states among all possible next states so as to avoid wasting computational time on evaluating prospectless states. The set of promising candidate states, denoted by $\mathcal{CAND}$, is chosen as those with a large LP fractional solution value. An integer parameter $1 \leq \lambda \leq I$ is used to indicate the number of states to be evaluated. By setting $\lambda = I$, every possible future state is evaluated. If $\lambda = 1$, only one future state, which is to say, that with the largest LP fractional solution value, is evaluated.

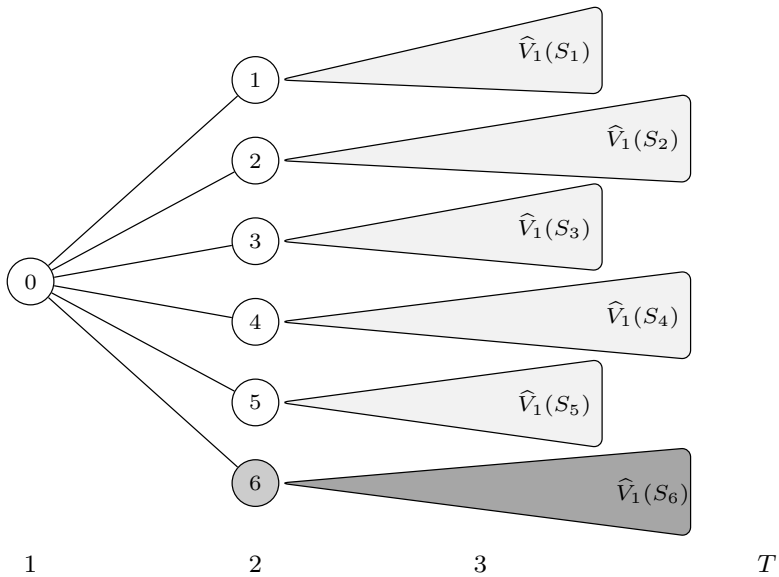
The ADP algorithm for an instance $\mathcal{P}$ is given in Algorithm 4.1. Starting from stage 1, at each stage, the algorithm fixes the state with the best approximated value until it reaches the last stage and returns the solution $\mathbf{y}^{ADP}$ and its objective value UB^{ADP}. While executing the algorithm, UB^{ADP} is updated whenever a better solution is obtained. In this algorithm, a subroutine $getCand(\mathbf{y}^{LP}, \lambda, s)$, which selects candidate states to be evaluated at stage s, is used. From the fractional solution $\mathbf{y}^{LP}$, this subroutine compares the y_{is}^{LP} values for $i \in \mathcal{I}$ and returns a set of indices of λ largest fractional values: $\mathcal{CAND}(s) = \{(i^1, s), \ldots, (i^\lambda, s)\}$.

The overall procedure of the ADP algorithm is illustrated in Figure 4.2. At state 0 in stage 1, the next state is decided based on the approximated value function

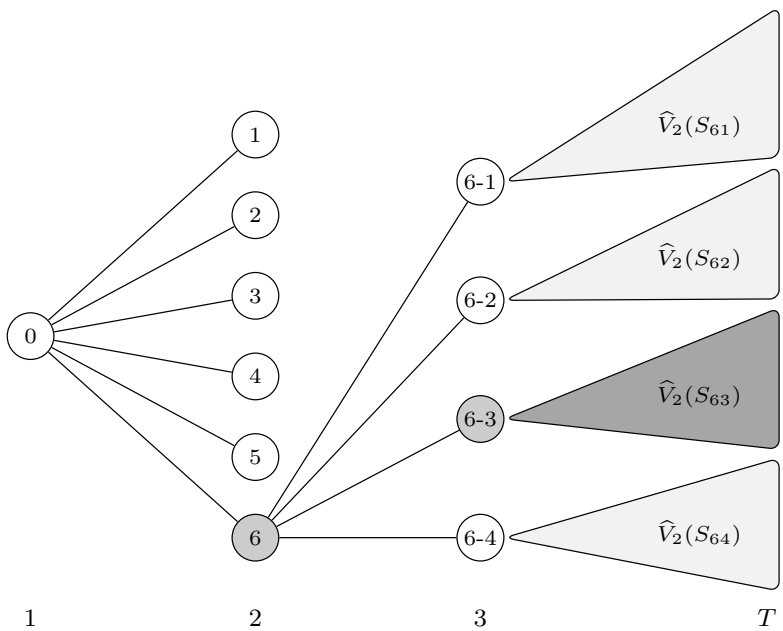
Algorithm 4.1: ADP$(\mathcal{P}, \varepsilon, \lambda, \gamma^{up}, \gamma^{down}, \delta)$

$\mathcal{Y}_1 \leftarrow \emptyset$, $\mathcal{Y}_0 \leftarrow \emptyset$, $s \leftarrow 1$, $UB^{ADP} \leftarrow \infty$; `// initialization`
$(obj, \mathbf{y}^{LP}) \leftarrow solveLP(\mathcal{P}, \mathcal{Y}_0, \mathcal{Y}_1)$; `// solve LP relaxation`
$\mathcal{CAND}(s) \leftarrow getCand(\mathbf{y}^{LP}, \lambda, s)$; `// select candidates to be evaluated at stage` s
do
  forall $c \in \mathcal{CAND}(s)$ **do** `// evaluate each candidate state`
    $\big(LB(c), \mathbf{y}^{LP}(c)\big) \leftarrow solveLP(\mathcal{P}, \mathcal{Y}_0, \mathcal{Y}_1 \cup \{c\})$;
    $\big(UB(c), \mathbf{y}^{LPNF}(c)\big) \leftarrow LPNF(\mathcal{P}, \gamma^{up}, \gamma^{down}, \delta, \mathcal{Y}_0, \mathcal{Y}_1 \cup \{c\})$;
    if $UB(c) < UB^{ADP}$ **then** `// if better solution found`
      $(UB^{ADP}, \mathbf{y}^{ADP}) \leftarrow (UB(c), \mathbf{y}^{LPNF}(c))$
    end
    $\widehat{V}(c) \leftarrow (1-\varepsilon) \cdot LB(c) + \varepsilon \cdot UB(c)$;
  end
  $c^* \leftarrow \underset{c \in \mathcal{CAND}(s)}{\arg\min}\ \widehat{V}(c)$;
  $\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{c^*\}$;
  $\mathcal{CAND}(s+1) \leftarrow getCand(\mathbf{y}^{LP}(c^*), \lambda, s+1)$;
  $s \leftarrow s + 1$;
while $s < S$;
$(obj, \mathbf{y}^{LP}) \leftarrow solveLP(\mathcal{P}, \mathcal{Y}_0, \mathcal{Y}_1)$;
if $obj < UB^{ADP}$ **then**
  $(UB^{ADP}, \mathbf{y}^{ADP}) \leftarrow (obj, \mathbf{y}^{LP})$
end
return $(UB^{ADP}, \mathbf{y}^{ADP})$

$\widehat{V}_1(R_1)$ which is calculated by the lower and upper bound values of the state R_1. After choosing the most promising state with the highest approximated value, state 6 in Figure 4.2(a), the states in the next stage are evaluated as shown in 4.2(b), and so on.



(a) Illustration of ADP algorithm: At stage 1



(b) Illustration of ADP algorithm: At stage 2

Figure 4.2: Illustration of ADP algorithm

4.4 Computational Experiments

We conducted two sets of computational experiments. The first aims to compare the performance of the presented time-flow model and the ADP algorithm using a set of instances similar to a specific real-world manufacturing environment. In the second experiment, the time-flow model and ADP algorithm are further compared with a recent big bucket model using a set of instances from the literature.

4.4.1 Comparison with (TF-FL) Model

The set of instances used for the first experiment is identical to that used in Chapter 3 which is generated considering the characteristics of the manufacturing environment introduced in Lee & Lee (2020). Among the instances with different problem dimensions, we use the largest dimension, $(I, T) = (15, 15)$ which was shown to be a hard instance by the test results in Chapter 3. Again, all of our experiments were conducted on an Intel Core 3.10 GHz PC with 16 GB RAM under Windows 10 Pro. The proposed algorithms were implemented in C++. FICO Xpress 8.9 with its default parameter settings was used as the LP/MIP solver. We set the time limit for solving a MIP problem to 600 s.

For the ADP algorithm, we fix the threshold parameters $(\gamma^{down}, \gamma^{up})$ to $(0, 0.9)$ and use (TF-FL) as the base model which provides tight LP bounds as well as shows good performance with the LPNF algorithm. In addition, fixing parameters $\delta \in \{1, 3, 5\}$, candidate parameters $\lambda \in \{2, 3, 5\}$, and weighting parameters $\epsilon \in \{0, 0.2, 0.5, 0.8, 1\}$ are used for the ADP algorithm. Note that when $\epsilon = 0$, the parameter δ has no meaning because the LPNF algorithm is not used.

Tables 4.1, 4.2, and 4.3 report the relative solution values, the ratio of instances

whose ADP solution is better than the *MIP Best Sol*, and the computation time for $\lambda = 2$, 3, and 5, respectively. The relative solution values and computation time are also illustrated in Figures 4.3 and 4.4, respectively. In Figure 4.4, the computation time for the ADP algorithms with ϵ greater than zero are averaged and grouped together because there are no meaningful differences between them.

Overall, the performance of the ADP algorithm seems satisfactory. As shown in Tables 4.1 – 4.3, the largest average deviation from *MIP Best Sol* is only about 4% when the LP bound is not used ($\epsilon = 1$). When both the LP bound and the LPNF algorithm are used, the ADP algorithms succeed in finding even better solutions under most of the parameter settings. For instance, when $(\lambda, \delta, \epsilon) = (5, 1, 0.2)$ is used, the solution quality improves by more than 10% compared with *MIP Best*

Table 4.1: Test results for the ADP algorithm: $\lambda = 2$

λ	δ	ϵ	*Relative Solution Value* (%)	*Better Solution* (%)	*Time* (s)
2	-	0	93.54	80.00	66.5
	1	0.2	92.46	81.67	263.6
		0.5	94.81	76.67	259.4
		0.8	95.78	76.67	251.1
		1	95.84	75.00	272.0
	3	0.2	96.28	66.67	174.8
		0.5	98.94	61.67	170.1
		0.8	99.03	61.67	164.8
		1	99.38	56.67	183.6
	5	0.2	98.01	65.00	152.6
		0.5	101.08	55.00	163.2
		0.8	101.97	50.00	142.1
		1	102.45	48.33	170.7
Average			97.66	65.77	187.3

Table 4.2: Test results for the ADP algorithm: $\lambda = 3$

λ	δ	ϵ	*Relative Solution Value* (%)	*Better Solution* (%)	*Time* (s)
3	-	0	91.65	83.33	99.8
	1	0.2	90.59	85.00	420.1
		0.5	95.17	75.00	384.2
		0.8	96.05	70.00	366.3
		1	96.09	70.00	407.4
	3	0.2	96.89	66.67	260.5
		0.5	100.15	56.67	249.1
		0.8	101.12	55.00	236.1
		1	101.50	55.00	270.1
	5	0.2	96.92	63.33	224.3
		0.5	100.92	60.00	237.1
		0.8	103.43	40.00	206.4
		1	103.68	48.33	252.2
	Average		98.01	64.49	278.0

Sol. Under this setting, the ADP algorithm obtains better solutions for 90% of the instances.

Regarding the parameters, $\delta = 1$ and $\epsilon = 0.2$ show, as indicated in Figure 4.3, the best results with respect to the relative solution value. Smaller δ leads to better solutions at the expense of increased computation times. The average computation time is also proportional to the value of λ. Moreover, we observe that a larger λ does not always lead to better solution quality. For ϵ values with relatively good performance, a larger λ value helps improve the solution quality. On the contrary, for ϵ values with poor performance, evaluating additional states makes it even poorer.

Table 4.3: Test results for the ADP algorithm: $\lambda = 5$

λ	δ	ϵ	*Relative Solution Value* (%)	*Better Solution* (%)	*Time* (s)
5	-	0	89.82	91.67	158.2
	1	0.2	88.98	90.00	555.0
		0.5	94.72	71.67	569.3
		0.8	95.59	71.67	549.6
		1	96.20	70.00	590.8
	3	0.2	95.68	70.00	431.2
		0.5	100.79	60.00	407.6
		0.8	101.86	55.00	371.4
		1	101.86	55.00	474.2
	5	0.2	97.86	63.33	373.8
		0.5	102.62	55.00	373.1
		0.8	103.48	53.33	318.0
		1	103.93	51.67	412.3
	Average		97.95	66.03	429.6

4.4.2 Comparison with Big Bucket Model

In our second experiment, we further investigate the performance of the `(TF)` and ADP algorithm relative to a big bucket model recently proposed by Mahdieh et al. (2018). This model, denoted as `(MCB)`, uses a multi-commodity flow formulation to capture the sequence of production within each bucket. As far as we know, `(MCB)` is the latest LSP model that can incorporate the sequence-dependent setup, setup crossover, and setup carryover. Note that the models presented in Guimarães et al. (2014) cannot incorporate them all simultaneously. Moreover, `(MCB)` is an improved version of the models previously presented in Menezes et al. (2011) and Clark et al. (2014).

For the comparison, we create a set of instances according to the instance gener-

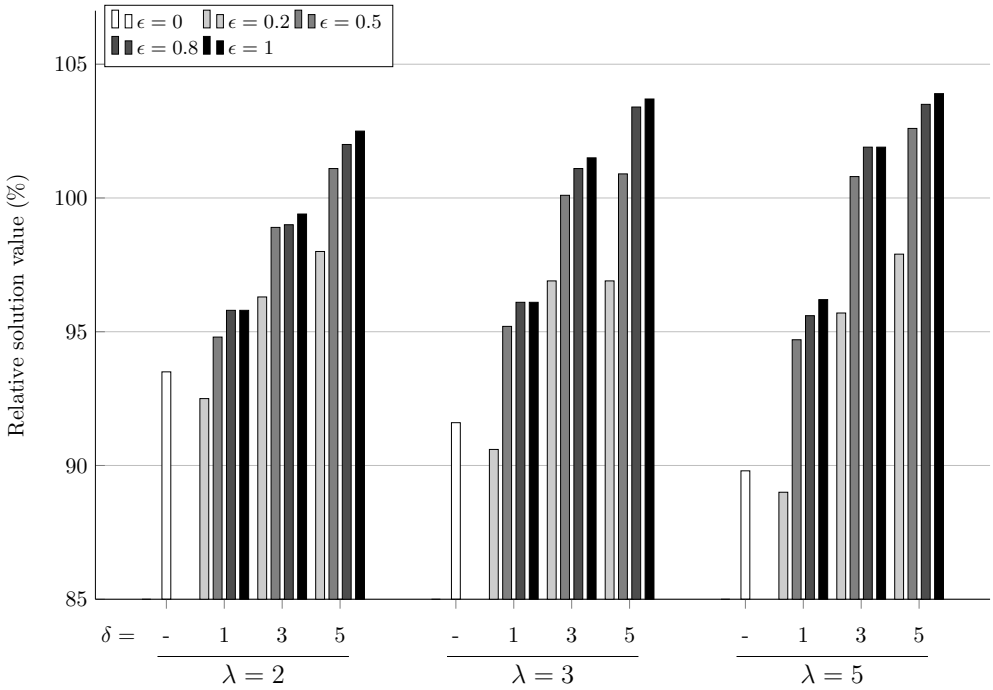


Figure 4.3: Test results for the ADP algorithm: Solution quality

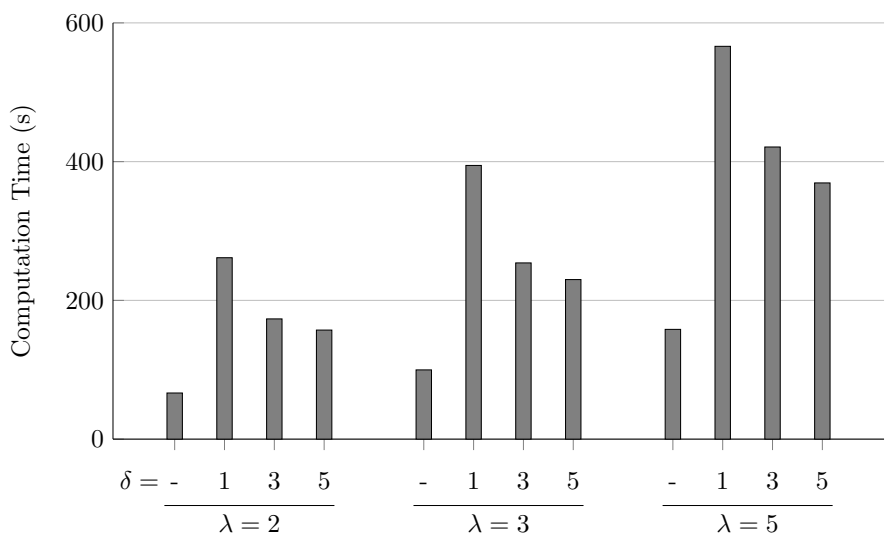


Figure 4.4: Test results for the ADP algorithm: Computation time

ation scheme in Almada-Lobo, Klabjan, et al. (2007), which is frequently used in the literature (e.g. James & Almada-Lobo, 2011; Guimarães et al., 2014). The instance type is defined by the combination of problem dimensions ($I \times T$), capacity utilization (Cut), and setup cost factor (θ). See Almada-Lobo, Klabjan, et al. (2007) for further details. We additionally introduce one more parameter to control the average length of the setup time *(Setup Time)*, as in the instances of the first experiment. We use the following parameters: $I \in \{10, 15\}$, $T \in \{10, 15\}$, $Cut \in \{0.6, 0.8, 1\}$, $\theta \in \{50, 100\}$, and $Setup\ Time \in \{S, M, L\}$. The (st^{min}, st^{max}) of classes S, M, and L are $(0.2, 0.4)$, $(0.4, 0.6)$, and $(0.6, 0.8)$, respectively. For each combination, we generate five instances, resulting in a total of 360 instances. For these instances, we set the time limit for solving a MIP problem to 1800 s, and for the ADP algorithm, 900 s.

We compare the results obtained by solving `(TF)` and `(MCB)` with the MIP solver and those obtained by our ADP and LPNF algorithms. For the algorithms, we use `(TF-FL)` as a base model which provides the tightest bound among other models and set the parameters as $(\gamma^{down}, \gamma^{up}, \delta, \lambda) = (0, 0.9, 3, 3)$. Table 4.4 represents the relative solution quality obtained from different approaches when we let the solution values of `(TF)` as 100. The subscript of the ADP algorithm indicates the types of bounds used; that is, $\texttt{ADP}_{\texttt{LU}}$ uses both the lower and upper bounds (using $\epsilon = 0.2$), while $\texttt{ADP}_{\texttt{L}}$ uses only the lower bound ($\epsilon = 0$).

In the comparison of the two models, `(TF)` performs worse than `(MCB)` on average, though it is better in some instances. This can be explained by the fact that, due to the microperiods, the size of `(TF)` is larger than that of `(MCB)`, which makes it harder to solve. The performance of `(TF)` gets better as the setup time increases, which

appears to be because the maximum number of items that can be produced within a macroperiod decreases, resulting in fewer microperiods. This result indicates that (TF) can be beneficial when the number of setups that can be conducted within a macroperiod is limited. Although (MCB) shows relatively better performance than (TF), it is not successful in solving instances within the time limit.

The solution provided by $\texttt{ADP}_{\texttt{LU}}$ is 3% better than that of (TF) on average. Moreover, it is even better than the solution of (MCB), though the difference is quite small. Similar to the results of the first experiment, $\texttt{ADP}_{\texttt{LU}}$ performs better than

Table 4.4: Comparison of the models and the solution algorithms

Factors	*Relative Solution Value* (%)					*Time* (s)			
	TF	MCB	$\texttt{ADP}_{\texttt{LU}}$	$\texttt{ADP}_{\texttt{L}}$	LPNF	TF/MCB	$\texttt{ADP}_{\texttt{LU}}$	$\texttt{ADP}_{\texttt{L}}$	LPNF
Dim									
10×10	100.0	96.61	101.88	105.26	112.15	1800	120.8	24.6	2.1
10×15	100.0	98.40	98.57	103.00	109.60	1800	403.8	92.3	6.7
15×10	100.0	96.34	95.94	97.44	105.52	1800	256.0	72.5	4.8
15×15	100.0	98.04	91.43	91.92	100.29	1800	537.7	242.0	14.2
Cut									
0.6	100.0	95.78	96.43	99.24	109.63	1800	337.4	97.5	6.4
0.8	100.0	97.68	96.28	98.91	105.67	1800	332.4	113.1	7.0
1	100.0	98.59	98.14	100.07	105.37	1800	318.9	113.0	7.4
θ									
50	100.0	97.24	96.40	98.49	106.73	1800	333.4	112.4	7.1
100	100.0	97.45	97.51	100.32	107.05	1800	325.7	103.4	6.8
Setup Time									
S	100.0	91.82	93.60	97.60	104.64	1800	658.7	227.3	13.6
M	100.0	98.40	98.15	99.89	107.34	1800	247.4	69.3	5.0
L	100.0	101.83	99.11	100.74	108.69	1800	82.6	27.0	2.3
Total	100.0	97.35	96.95	99.41	106.89	1800	329.6	107.9	7.0

$\texttt{ADP}_{\texttt{L}}$ with a longer computation time. Notably, the relative performance of both $\texttt{ADP}_{\texttt{LU}}$ and $\texttt{ADP}_{\texttt{L}}$ gets better as the size of instances increases. This indicates that the ADP algorithms can be more beneficial for larger instances. The computation time required for the ADP algorithms is much shorter than 1800 s.

However, the performance of the ADP algorithms relies on the choice of the parameters. Especially, in contrast to other parameters whose effects are quite predictable, choosing the value of the weighting parameter ϵ may require several trials. In spite of the additional effort for choosing appropriate parameters, considering its advantages in terms of computation time, the ADP algorithm can be one viable option for solving LSPs with sequence-dependent setups.

4.5 Summary

In this chapter, we devised an ADP algorithm that can alleviate the curse-of-dimensionality problem of the traditional DP approaches. The proposed algorithm uses the value function approximation procedure that estimates the value of states using both the lower and upper bounds of them. In the first experiment, our ADP algorithm shows some benefits over the MIP solver; that is, it can find a better solution within a shorter computation time. In addition, in the second experiment, the ADP algorithms show competitive performance in comparison with a state-of-the-art big bucket model in the literature.

Chapter 5

Conclusion

5.1 Summary and Contributions

In this dissertation, we proposed integer optimization and ADP approaches to solve LSP with sequence-dependent setups. In Chapter 2, we addressed the single-period substructures of the big bucket models of LSP with sequence-dependent setups which, contrary to the problems with sequence-independent setups, have not been investigated in previous research. We conducted polyhedral analysis on the single-period substructures and derived new families of valid inequalities, that is, `S-STAR` and `U-STAR` inequalities. The proposed inequalities are demonstrated to define facets of the single-period substructures under some conditions. In addition, we presented polynomial-time separation procedures of these inequalities which use the maximum-flow algorithm. Then, we provided a new type of extended formulation which is shown to provide the same lower bound as that of the original formulation with all `S-STAR` added. The practical effectiveness of the newly proposed inequalities and extended formulations was compared with the existing formulations by conducting computational experiments. The results of computational experiments on both single-period and multi-period instances demonstrated distinct advantages of the newly proposed inequalities and formulations in tightening the LP relaxation

bounds.

In Chapter 3, we provided new optimization models for LSP that can incorporate the setup crossover and carryover, which are important extensions of the problem encountered frequently in the real-world manufacturing processes. The proposed models are called time-flow models which were constructed based on the results of Chapter 2. After further tightening the proposed models using the well-known reformulation techniques, we compared the strength of the proposed models with the existing models. The time-flow models were demonstrated to provide a much tighter LP relaxation bound than that of the standard GLSP-based models. Furthermore, we proposed an LP-based heuristic algorithm based on the time-flow model which can provide feasible solutions quickly. The performances of the proposed models and heuristic algorithm were tested through computational experiments using instances from the real-world industry. From the computational experiments, it was demonstrated that the proposed models have advantages compared with standard models in terms of tightness and solvability.

In Chapter 4, we devised an ADP algorithm for LSP with sequence-dependent setups to mitigate the so-called "curse-of-dimensionality" issue of the traditional DP-based solution approaches, that is, the number of states can easily explode as the problem dimension increases (Powell, 2007). It becomes hard to evaluate the value of exponentially many states with the traditional DP recursion approaches. To alleviate this drawback, our ADP algorithm adapts the value function approximation approach to approximate the value of the states. The proposed value function approximation approach estimates the value of each state using both the lower and upper bound values without recursive evaluation of future states. Therefore, the pro-

posed ADP algorithm enjoys the advantages of both the tight lower bound values obtained by the time-flow model and the upper bound value which can be obtained in a short time by the LP-based heuristic algorithm proposed in Chapter 3. To examine the performance of the proposed algorithm, we conducted two sets of computational experiments using instances from the real-world industry and the previous literature, respectively. The results of the first experiment indicated that the proposed ADP algorithm has clear benefits over the algorithm provided by the MIP solver, that is, it can find a better solution within a shorter computation time. Moreover, in the second experiment, the ADP algorithm showed competitive performance in comparison with a state-of-the-art big bucket model in the literature.

5.2 Future Research Directions

The results of the dissertation offer several future research directions. The valid inequalities proposed in Chapter 2 can be utilized to devise efficient solution algorithms such as branch-and-cut algorithms for LSP with sequence-dependent setups. Specifically, when used as cuts in the branch-and-cut algorithm, detailed algorithmic components such as the number of cuts added in one iteration, the frequency of adding cuts during the tree search, and the order in which cuts are added, affect the algorithm performance. Therefore, considering the algorithmic elements, additional extensive computational experiments are required in future research. In addition, the known inequalities provided in previous studies such as the (l, S)-inequality can be used together with the proposed `S-STAR` or `U-STAR` inequalities. Analyzing the interaction between these inequalities is also an interesting future research topic.

The polyhedral results can also be extended to other optimization problems

such as TSP, CVRP, IRP, and their variants due to the similarity between the single-period substructures and these problems. As we mentioned, the proposed inequalities are closely related to the known results for those problems, for instance, generalized multistar inequalities for CVRP and disjoint route inequalities for IRP. Therefore, theoretical analysis in this dissertation can be utilized for those problems to derive new results.

In addition, the proposed time-flow formulation can be adapted to matheuristics, that is, heuristic algorithms based on mathematical programming methodologies, to solve real-world problems. Other than the proposed LP-based fixing heuristic, various matheuristics such as *relax-and-fix* and *fix-and-optimize* algorithms have been popularly used to solve LSP with sequence-dependent setups that occurs in various industries. The performance of these heuristics is significantly affected by the tightness of the base formulation. At the same time, the formulation with a large number of variables and constraints might be prohibitive because it may have to be solved many times repeatedly. Considering this trade-off between the formulation size and tightness, the proposed time-flow formulation can be utilized to improve the performance of the various matheuristics which can be another possible future research direction.

As another research direction, there are some possible improvements to the proposed ADP algorithm. For instance, different schemes could be used to obtain primal and dual bound values instead of the LPNF algorithm and LP relaxation, respectively. In particular, as the size of the real-world problems to be solved has been continually increasing, even the proposed ADP algorithm can take too much computation time to evaluate the approximated value of states. In this case, the approx-

imation procedure can be made faster by sacrificing the tightness of the bounds. In fact, even whether they are valid bounds or not is unimportant when approximating the values. It is sufficient if the true value can be approximated appropriately. Further, one can newly define the state in each stage, such as the inventory position of the items or the cumulative production amounts. With these states, the value function can be approximated using various techniques such as piece-wise linear function fitting, regression, or machine-learning-based techniques. In addition, the value function approximation scheme using lower and upper bounds can also be adapted to solution frameworks other than the DP-based approaches. For instance, during tree search algorithms one can choose nodes to be explored based on their bound information.

Finally, the ADP algorithm can be extended for application to cases in which uncertainty is present in data (e.g., demand or setup time). In fact, the ADP algorithm has been popularly used as a tool for sequential decision making under uncertainty where the uncertainty reveals over time. Then, decision makers should choose action in each stage considering the revealed uncertainty until the stage and then observe the realization of uncertain parameters of the next stage. In such cases, the expected value of each state should be approximated, which can be achieved by various methods such as sampling or scenario grouping. This is another interesting direction for future research.

Bibliography

Aggarwal, Alok & James K Park (1993). "Improved algorithms for economic lot size problems". *Operations research* 41.3, pp. 549–571.

Akartunalı, Kerem, Ioannis Fragkos, Andrew J. Miller & Tao Wu (2016). "Local Cuts and Two-Period Convex Hull Closures for Big-Bucket Lot-Sizing Problems". *INFORMS Journal on Computing* 28.4, pp. 766–780.

Alem, Douglas, Eduardo Curcio, Pedro Amorim & Bernardo Almada-Lobo (2018). "A computational study of the general lot-sizing and scheduling model under demand uncertainty via robust and stochastic approaches". *Computers & Operations Research* 90, pp. 125–141.

Alfieri, A., P. Brandimarte & S. D'Orazio (2002). "LP-based heuristics for the capacitated lot-sizing problem: The interaction of model formulation and solution algorithm". *International Journal of Production Research* 40.2, pp. 441–458.

Alipour, Zohreh, Fariborz Jolai, Ehsan Monabbati & Nima Zaerpour (2020). "General lot-sizing and scheduling for perishable food products". *RAIRO - Operations Research* 54.3, pp. 913–931.

Almada-Lobo, Bernardo, Alistair Clark, Luis Guimarães, Gonçalo Figueira & Pedro Amorim (2015). "Industrial insights into lot sizing and scheduling modeling". *Pesquisa Operacional* 35.3, pp. 439–464.

Almada-Lobo, Bernardo, Diego Klabjan, Maria Antónia Carravilla & José F. Oliveira (2007). "Single machine multi-product capacitated lot sizing with sequence-dependent setups". *International Journal of Production Research* 45.20, pp. 4873–4894.

Almeder, Christian & Bernardo Almada-Lobo (2011). "Synchronisation of scarce resources for a parallel machine lotsizing problem". *International Journal of Production Research* 49.24, pp. 7315–7335.

Alves, Fernanda F, Thiago H Nogueira, Mauricio C de Souza & Martın G Ravetti (2021). "Approaches for the joint resolution of lot-sizing and scheduling with infeasibilities occurrences". *Computers & Industrial Engineering* 155, p. 107176.

Avella, Pasquale, Maurizio Boccia & Laurence A Wolsey (2018). "Single-period cutting planes for inventory routing problems". *Transportation Science* 52.3, pp. 497–508.

Barany, Imre, Tony J Van Roy & Laurence A Wolsey (1984). "Strong formulations for multi-item capacitated lot sizing". *Management Science* 30.10, pp. 1255–1261.

Bellman, R.E. (1957). *Dynamic Programming*. Rand Corporation research study. Princeton University Press.

Belo-Filho, Márcio A F, Franklina M B Toledo & Bernardo Almada-Lobo (2014). "Models for capacitated lot-sizing problem with backlogging, setup carryover and crossover". *Journal of the Operational Research Society* 65.11, pp. 1735–1747.

Bertsekas, Dimitri (2012). *Dynamic programming and optimal control: Volume I*. Vol. 1. Athena scientific.

Bertsimas, Dimitris & Ramazan Demir (2002). "An approximate dynamic programming approach to multidimensional knapsack problems". *Management Science* 48.4, pp. 550–565.

Bitran, Gabriel R & Horacio H Yanasse (1982). "Computational complexity of the capacitated lot size problem". *Management Science* 28.10, pp. 1174–1186.

Büyüktahtakın, İ Esra & Ning Liu (2016). "Dynamic programming approximation algorithms for the capacitated lot-sizing problem". *Journal of Global Optimization* 65.2, pp. 231–259.

Büyüktahtakın, I Esra, J Cole Smith & Joseph C Hartman (2018). "Partial objective inequalities for the multi-item capacitated lot-sizing problem". *Computers & Operations Research* 91, pp. 132–144.

Camargo, V. C B, F. M B Toledo & B. Almada-Lobo (2012). "Three time-based scale formulations for the two-stage lot sizing and scheduling in process industries". *Journal of the Operational Research Society* 63.11, pp. 1613–1630.

Carvalho, Desiree M & Mariá CV Nascimento (2022). "Hybrid matheuristics to solve the integrated lot sizing and scheduling problem on parallel machines with sequence-dependent and non-triangular setup". *European Journal of Operational Research* 296.1, pp. 158–173.

Cervantes-Sanmiguel, KI, MJ Vargas-Flores & OJ Ibarra-Rojas (2021). "A two-stage sequential approach for scheduling with lot-sizing decisions in the context of plastic injection systems". *Computers & Industrial Engineering* 151, p. 106969.

Chen, W-H & J-M Thizy (1990). "Analysis of relaxations for the multi-item capacitated lot-sizing problem". *Annals of operations Research* 26.1, pp. 29–72.

Chiang, Tsung-Che & Li-Chen Fu (2009). "Using a family of critical ratio-based approaches to minimize the number of tardy jobs in the job shop with sequence dependent setup times". *European Journal of Operational Research* 196.1, pp. 78–92.

Christopher, Martin (2016). *Logistics & supply chain management*. Pearson UK.

Claassen, GDH, Johanna C Gerdessen, Eligius MT Hendrix & Jack GAJ van der Vorst (2016). "On production planning and scheduling in food processing industry: Modelling non-triangular setups andproduct decay". *Computers & Operations Research* 76, pp. 147–154.

Clark, Alistair, Masoumeh Mahdieh & Socorro Rangel (2014). "Production lot sizing and scheduling with non-triangular sequence-dependent setup times". *International Journal of Production Research* 52.8, pp. 2490–2503.

Constantino, Miguel (1996). "A cutting plane approach to capacitated lot-sizing with start-up costs". *Mathematical Programming* 75.3, pp. 353–376.

Copil, Karina, Martin Wörbelauer, Herbert Meyr & Horst Tempelmeier (2017). "Simultaneous lotsizing and scheduling problems: a classification and review of models". *OR Spectrum* 39.1, pp. 1–64.

Cunha, Jesus O & Rafael A Melo (2021). "Valid inequalities, preprocessing, and an effective heuristic for the uncapacitated three-level lot-sizing and replenishment problem with a distribution structure". *European Journal of Operational Research* 295.3, pp. 874–892.

Dauzere-Peres, Stephane & Jean-Bernard Lasserre (1994). "Integration of lotsizing and scheduling decisions in a job-shop". *European Journal of Operational Research* 75.2, pp. 413–426.

Denizel, Meltem & Haldun Süral (2006). "On alternative mixed integer programming formulations and LP-based heuristics for lot-sizing with setup times". *Journal of the Operational Research Society* 57.4, pp. 389–399.

Desaulniers, Guy, Jacques Desrosiers & Marius M Solomon (2006). *Column generation*. Vol. 5. Springer Science & Business Media.

Doostmohammadi, Mahdi & Kerem Akartunalı (2018). "Valid inequalities for two-period relaxations of big-bucket lot-sizing problems: Zero setup case". *European Journal of Operational Research* 267.1, pp. 86–95.

Drexl, Andreas & Knut Haase (1995). "Proportional lotsizing and scheduling". *International Journal of Production Economics* 40.1, pp. 73–87.

Drexl, Andreas & Alf Kimms (1997). "Lot sizing and scheduling—survey and extensions". *European Journal of operational research* 99.2, pp. 221–235.

Eppen, Gary D & R Kipp Martin (1987). "Solving multi-item capacitated lot-sizing problems using variable redefinition". *Operations Research* 35.6, pp. 832–848.

Federgruen, Awi & Michal Tzur (1991). "A simple forward algorithm to solve general dynamic lot sizing models with n periods in $O(n \log n)$ or $O(n)$ time". *Management Science* 37.8, pp. 909–925.

Ferreira, Deisemara, Reinaldo Morabito & Socorro Rangel (2010). "Relax and fix heuristics to solve one-stage one-machine lot-scheduling models for small-scale soft drink plants". *Computers & Operations Research* 37.4, pp. 684–691.

Fiorotto, Diego Jacinto, Jackeline del Carmen Huaccha Neyra & Silvio Alexandre de Araujo (2020). "Impact analysis of setup carryover and crossover on lot sizing problems". *International Journal of Production Research* 58.20, pp. 6350–6369.

Fiorotto, Diego Jacinto, Raf Jans & Silvio Alexandre de Araujo (2017). "An analysis of formulations for the capacitated lot sizing problem with setup crossover". *Computers & Industrial Engineering* 106, pp. 338–350.

Fleischmann, Bernhard (1994). "The discrete lot-sizing and scheduling problem with sequence-dependent setup costs". *European Journal of Operational Research* 75.2, pp. 395–404.

Fleischmann, Bernhard & Herbert Meyr (1997). "The general lotsizing and scheduling problem". *Operations-Research-Spektrum* 19.1, pp. 11–21.

Florian, Michael, Jan Karel Lenstra & AHG Rinnooy Kan (1980). "Deterministic production planning: Algorithms and complexity". *Management science* 26.7, pp. 669–679.

Gavish, Bezalel & Stephen C Graves (1978). "The travelling salesman problem and related problems".

Gicquel, Céline, Abdel Lisser & Michel Minoux (2014). "An evaluation of semidefinite programming based approaches for discrete lot-sizing problems". *European Journal of Operational Research* 237.2, pp. 498–507.

Gicquel, Céline & Michel Minoux (2015). "Multi-product valid inequalities for the discrete lot-sizing and scheduling problem". *Computers & Operations Research* 54, pp. 12–20.

Gouveia, Luis (1995). "A result on projection for the vehicle routing problem". *European Journal of Operational Research* 85.3, pp. 610–624.

Guimarães, Luis, Diego Klabjan & Bernardo Almada-Lobo (2013). "Pricing, relaxing and fixing under lot sizing and scheduling". *European Journal of Operational Research* 230.2, pp. 399–411.

— (2014). “Modeling lotsizing and scheduling problems with sequence dependent setups”. *European Journal of Operational Research* 239.3, pp. 644–662.

Gupta, Diwakar & Thorkell Magnusson (2005). “The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times”. *Computers and Operations Research* 32.4, pp. 727–747.

Haase, Knut (1996). “Capacitated lot-sizing with sequence dependent setup costs”. *Operations-Research-Spektrum* 18.1, pp. 51–59.

Haase, Knut & Alf Kimms (2000). “Lot sizing and scheduling with sequence-dependent setup costs and times and efficient rescheduling opportunities”. *International Journal of Production Economics* 66.2, pp. 159–169.

Harris, Ford W (1913). “How many parts to make at once”. *Operations Research* 38.6, pp. 947–950.

Hartman, Joseph C, I Esra Büyüktahtakin & J Cole Smith (2010). “Dynamic-programming-based inequalities for the capacitated lot-sizing problem”. *IIE Transactions* 42.12, pp. 915–930.

Hoffman, Alan J (1976). “Total unimodularity and combinatorial theorems”. *Linear Algebra and its Applications* 13.1-2, pp. 103–108.

James, Ross JW & Bernardo Almada-Lobo (2011). “Single and parallel machine capacitated lotsizing and scheduling: New iterative MIP-based neighborhood search heuristics”. *Computers & Operations Research* 38.12, pp. 1816–1825.

Karmarkar, Uday S & Linus Schrage (1985). “The deterministic dynamic product cycling problem”. *Operations Research* 33.2, pp. 326–345.

Koch, Cyril, Taha Arbaoui, Yassine Ouazene, Farouk Yalaoui, Humbert De Brunier, Nicolas Jaunet & Antoine De Wulf (2022). “A Matheuristic Approach for Solving

a Simultaneous Lot Sizing and Scheduling Problem with Client Prioritization in Tire Industry". *Computers & Industrial Engineering*, p. 107932.

Koçlar, Ayşe & Haldun Süral (2005). "A note on "The general lot sizing and scheduling problem"". *OR Spectrum* 27.1, pp. 145–146.

Krarup, Jakob & Ole Bilde (1977). "Plant location, set covering and economic lot size: An $O(mn)$-algorithm for structured problems". *Numerische Methoden bei Optimierungsaufgaben Band 3*.

Küçükyavuz, Simge & Yves Pochet (2009). "Uncapacitated lot sizing with backlogging: The convex hull". *Mathematical Programming* 118.1, pp. 151–175.

Larroche, François, Odile Bellenguez & Guillaume Massonnet (2021). "Clustering-based solution approach for a capacitated lot-sizing problem on parallel machines with sequence-dependent setups". *International Journal of Production Research*, pp. 1–24.

Lasserre, Jean-Bernard (1992). "An integrated model for job-shop planning and scheduling". *Management Science* 38.8, pp. 1201–1211.

Lee, Younsoo & Kyungsik Lee (2020). "Lot-sizing and scheduling in flat-panel display manufacturing process". *Omega* 93, p. 102036.

Letchford, Adam N, Richard W Eglese & Jens Lysgaard (2002). "Multistars, partial multistars and the capacitated vehicle routing problem". *Mathematical Programming* 94.1, pp. 21–40.

Letchford, Adam N. & Juan José Salazar-González (2006). "Projection results for vehicle routing". *Mathematical Programming* 105.2-3, pp. 251–274.

— (2015). “Stronger multi-commodity flow formulations of the Capacitated Vehicle Routing Problem”. *European Journal of Operational Research* 244.3, pp. 730–738.

Leung, Janny MY, Thomas L Magnanti & Rita Vachani (1989). “Facets and algorithms for capacitated lot sizing”. *Mathematical programming* 45.1, pp. 331–359.

Maes, Johan, John O. McClain & Luk N. Van Wassenhove (1991). “Multilevel capacitated lotsizing complexity and LP-based heuristics”. *European Journal of Operational Research* 53.2, pp. 131–148.

Mahdieh, Masoumeh, Alistair Clark & Mehdi Bijari (2018). “A novel flexible model for lot sizing and scheduling with non-triangular, period overlapping and carry-over setups in different machine configurations”. *Flexible Services and Manufacturing Journal* 30.4, pp. 884–923.

Manne, Alan S (1958). “Programming of economic lot sizes”. *Management science* 4.2, pp. 115–135.

Maravelias, Christos T & Charles Sung (2009). “Integration of production planning and scheduling: Overview, challenges and opportunities”. *Computers & Chemical Engineering* 33.12, pp. 1919–1930.

Melega, Gislaine Mara, Silvio Alexandre de Araujo & Reinaldo Morabito (2020). “Mathematical model and solution approaches for integrated lot-sizing, scheduling and cutting stock problems”. *Annals of Operations Research* 295.2, pp. 695–736.

Menezes, António Aroso, Alistair Clark & Bernardo Almada-Lobo (2011). "Capacitated lot-sizing and scheduling with sequence-dependent, period-overlapping and non-triangular setups". *Journal of Scheduling* 14.2, pp. 209–219.

Meyr, Herbert (2000). "Simultaneous lotsizing and scheduling by combining local search with dual reoptimization". *European Journal of Operational Research* 120.2, pp. 311–326.

— (2002). "Simultaneous lotsizing and scheduling on parallel machines". *European Journal of Operational Research* 139.2, pp. 277–292.

Meyr, Herbert & Matthias Mann (2013). "A decomposition approach for the general lotsizing and scheduling problem for parallel production lines". *European Journal of Operational Research* 229.3, pp. 718–731.

Miller, Andrew J, George L Nemhauser & Martin WP Savelsbergh (2003a). "A multi-item production planning model with setup times: algorithms, reformulations, and polyhedral characterizations for a special case". *Mathematical programming* 95.1, pp. 71–90.

— (2003b). "On the polyhedral structure of a multi–item production planning model with setup times". *Mathematical Programming* 94.2, pp. 375–405.

Miller, Clair E, Albert W Tucker & Richard A Zemlin (1960). "Integer programming formulation of traveling salesman problems". *Journal of the ACM (JACM)* 7.4, pp. 326–329.

Mohan, Srimathy, Mohan Gopalakrishnan, Rahul Marathe & Ashwin Rajan (2012). "A note on modelling the capacitated lot-sizing problem with set-up carry-over and set-up splitting". *International Journal of Production Research* 50.19, pp. 5538–5543.

Nemhauser, George & Laurence Wolsey (1988). *Integer and Combinatorial Optimization*. John Wiley & Sons, Ltd. Chap. III.1, pp. 533–607.

Oyebolu, Folarin B, Jeroen van Lidth de Jeude, Cyrus Siganporia, Suzanne S Farid, Richard Allmendinger & Juergen Branke (2017). "A new lot sizing and scheduling heuristic for multi-site biopharmaceutical production". *Journal of Heuristics* 23.4, pp. 231–256.

Padberg, Manfred & Ting-Yi Sung (1991). "An analytical comparison of different formulations of the travelling salesman problem". *Mathematical Programming* 52.1, pp. 315–357.

Papageorgiou, Dimitri J, Myun-Seok Cheon, George Nemhauser & Joel Sokol (2015). "Approximate dynamic programming for a class of long-horizon maritime inventory routing problems". *Transportation Science* 49.4, pp. 870–885.

Pinedo, Michael L (2012). *Scheduling*. Vol. 29. Springer.

Pochet, Yves & Laurence A Wolsey (1994). "Polyhedra for lot-sizing with Wagner—Whitin costs". *Mathematical Programming* 67.1, pp. 297–323.

— (2006). *Production planning by mixed integer programming*. Springer Science & Business Media.

Potts, Chris N & Luk N Van Wassenhove (1992). "Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity". *Journal of the Operational Research Society* 43.5, pp. 395–406.

Powell, Warren B (2007). *Approximate dynamic programming: Solving the curses of dimensionality*. John Wiley & Sons.

— (2016). "Perspectives of approximate dynamic programming". *Annals of Operations Research* 241.1-2, pp. 319–356.

Powell, Warren B (2019). "A unified framework for stochastic optimization". *European Journal of Operational Research* 275.3, pp. 795–821.

Puterman, Martin L (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Rıos-Solıs, Yasmın Á, Omar J Ibarra-Rojas, Marta Cabo & Edgar Possani (2020). "A heuristic based on mathematical programming for a lot-sizing and scheduling problem in mold-injection production". *European Journal of Operational Research* 284.3, pp. 861–873.

Rogers, Jack (1958). "A computational approach to the economic lot scheduling problem". *Management science* 4.3, pp. 264–291.

Salomon, Marc, Marius M Solomon, Luk N Van Wassenhove, Yvan Dumas & Stephane Dauzère-Pérès (1997). "Solving the discrete lotsizing and scheduling problem with sequence dependent set-up costs and set-up times using the travelling salesman problem with time windows". *European Journal of Operational Research* 100.3, pp. 494–513.

Sarin, Subhash C, Hanif D Sherali & Liming Yao (2011). "New formulation for the high multiplicity asymmetric traveling salesman problem with application to the Chesapeake problem". *Optimization Letters* 5.2, pp. 259–272.

Stadtler, Hartmut, Christoph Kilger & Herbert Meyr (2015). *Supply chain management and advanced planning: concepts, models, software, and case studies*. Springer.

Suerie, Christopher (2006). "Modeling of period overlapping setup times". *European Journal of Operational Research* 174.2, pp. 874–886.

Suerie, Christopher & Hartmut Stadtler (2003). "The capacitated lot-sizing problem with linked lot sizes". *Management Science* 49.8, pp. 1039–1054.

Sung, Charles & Christos T Maravelias (2008). "A mixed-integer programming formulation for the general capacitated lot-sizing problem". *Computers & Chemical Engineering* 32.1-2, pp. 244–259.

Topaloglu, Huseyin & Warren B Powell (2006). "Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems". *INFORMS Journal on Computing* 18.1, pp. 31–42.

Toriello, Alejandro, George Nemhauser & Martin Savelsbergh (2010). "Decomposing inventory routing problems with approximate value functions". *Naval Research Logistics* 57.8, pp. 718–727.

Toscano, Alyne, Deisemara Ferreira & Reinaldo Morabito (2019). "A decomposition heuristic to solve the two-stage lot sizing and scheduling problem with temporal cleaning". *Flexible Services and Manufacturing Journal* 31.1, pp. 142–173.

— (2020). "Formulation and MIP-heuristics for the lot sizing and scheduling problem with temporal cleanings". *Computers & Chemical Engineering* 142, p. 107038.

Toth, Paolo & Daniele Vigo (2002). *The vehicle routing problem*. SIAM.

Trigeiro, William W, L Joseph Thomas & John O McClain (1989). "Capacitated lot sizing with setup times". *Management science* 35.3, pp. 353–366.

Van Hoesel, CPM & Albert Peter Marie Wagelmans (1996). "An $O(T^3)$ algorithm for the economic lot-sizing problem with constant capacities". *Management science* 42.1, pp. 142–150.

Van Vyve, Mathieu (2007). "Algorithms for single-item lot-sizing problems with constant batch size". *Mathematics of Operations Research* 32.3, pp. 594–613.

Wagelmans, Albert, Stan Van Hoesel & Antoon Kolen (1992). "Economic lot sizing: An $O(n \log n)$ algorithm that runs in linear time in the Wagner-Whitin case". *Operations Research* 40.1-supplement-1, S145–S156.

Wagner, Harvey M & Thomson M Whitin (1958). "Dynamic version of the economic lot size model". *Management science* 5.1, pp. 89–96.

Wolsey, Laurence A (1989). "Uncapacitated lot-sizing problems with start-up costs". *Operations Research* 37.5, pp. 741–747.

Wolsey, Laurence A. (1997). "MIP modelling of changeovers in production planning and scheduling problems". *European Journal of Operational Research* 99.1, pp. 154–165.

Wolsey, Laurence A (2020). *Integer programming*. John Wiley & Sons.

Xiao, Jing, Huasheng Yang, Canrong Zhang, Li Zheng & Jatinder N.D. Gupta (2015). "A hybrid Lagrangian-simulated annealing-based heuristic for the parallel-machine capacitated lot-sizing and scheduling problem with sequence-dependent setup times". *Computers and Operations Research* 63, pp. 72–82.

Zangwill, Willard I (1966). "A deterministic multi-period production scheduling model with backlogging". *Management Science* 13.1, pp. 105–119.

Zhu, Xiaoyan & Wilbert E. Wilhelm (2006). "Scheduling and lot sizing with sequence-dependent setup: A literature review". *IIE Transactions* 38.11, pp. 987–1007.

Appendix A

Pattern-based Formulation in Chapter 2

We present a pattern-based formulation which is obtained by applying Dantzig-Wolfe decomposition for the original LSP-SQ. In particular, we apply a period-wise decomposition, that is, each pattern corresponds to a production plan of a single period. Let $\mathcal{P}_t$ be the set of possible production schedules in period t. For each pattern $p \in \mathcal{P}_t$, the following associated parameters are defined:

- $\bar{x}_{it}^p$: Production amount of item i in pattern $p \in \mathcal{P}_t$.
- $\bar{y}_{it}^p = 1$ if item i is produced in pattern $p \in \mathcal{P}_t$.
- $\bar{z}_{ijt}^p = 1$ if setup from item i to j occurs in pattern $p \in \mathcal{P}_t$.
- C_{tp}: Sum of the production and setup costs of pattern $p \in \mathcal{P}_t$, corresponding to $(\bar{\mathbf{x}}^p, \bar{\mathbf{y}}^p, \bar{\mathbf{z}}^p)$.

Let the pattern variable $\lambda_{tp} = 1$ if the pattern $p \in \mathcal{P}_t$ is selected in period t. The master problem (MP) is represented as follows:

$$\text{(MP)} \quad \text{minimize} \quad \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}(hc_{it}s_{it} + bc_{it}b_{it}) + \sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}_t} C_{tp}\lambda_{tp} \tag{A.1a}$$

$$\text{subject to} \quad s_{it-1} - b_{it-1} + \sum_{p \in \mathcal{P}_t} \bar{x}_{it}^{p} \lambda_{tp}$$

$$= d_{it} + s_{it} - b_{it} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \quad \text{(A.1b)}$$

$$\sum_{p \in \mathcal{P}_t} \bar{z}_{0it}^{p} \lambda_{tp} = \sum_{p \in \mathcal{P}_{t+1}} \bar{z}_{i0t+1}^{p} \lambda_{t+1p} \quad \forall i \in \mathcal{I}, t \in \mathcal{T} \setminus \{T\} \quad \text{(A.1c)}$$

$$\sum_{p \in \mathcal{P}_t} \lambda_{tp} = 1 \qquad \forall t \in \mathcal{T} \quad \text{(A.1d)}$$

$$s_{it},\ b_{it} \geq 0 \qquad \forall i \in \mathcal{I},\ t \in \mathcal{T} \quad \text{(A.1e)}$$

$$\lambda_{tp} \in \{0, 1\} \qquad \forall t \in \mathcal{T}, p \in \mathcal{P}_t \quad \text{(A.1f)}$$

Demand constraints (2.1b) and setup carryover constraints (2.1e) of the original LSP-SQ (2.1) are kept in `(MP)` as linking constraints (A.1b)–(A.1c). Other constraints are presented in subproblems. It is known that, as the pattern incorporates all the decisions regarding a single period, the LP relaxation bound of `(MP)` is equivalent to that which is obtained when the convex hull of the single-period solution set $\mathcal{X}$ is known (Wolsey, 2020).

The LP relaxation of `(MP)` is solved by a column generation procedure which recursively adds profitable columns by solving pricing subproblems until no one is found (Desaulniers et al., 2006). Let μ_{it}, π_{it}, and σ_t be the dual variables of constraints (A.1b)–(A.1d), respectively. Then, the pricing subproblem for period t which tries to find profitable patterns by minimizing the reduced cost can be constructed as follows:

$$(\mathtt{SP}_t) \quad \text{minimize} \quad \sum_{i \in \mathcal{I}} \Big((pc_{it} - \mu_{it}) x_{it} - \pi_{it} z_{0it} + \pi_{it-1} z_{i0t} \Big)$$

$$+ \sum_{(i,j)\in\mathcal{A}} sc_{ijt} z_{ijt} \tag{A.2a}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{I}} x_{it} + \sum_{(i,j)\in\mathcal{A}} st_{ijt} z_{ijt} \le K_t \tag{A.2b}$$

$$x_{it} \le u_{it} y_{it} \qquad \forall i \in \mathcal{I} \tag{A.2c}$$

$$\sum_{i\in\mathcal{I}} z_{0it} = 1 \tag{A.2d}$$

$$\sum_{j\in\mathcal{I}_0} z_{jit} = \sum_{j\in\mathcal{I}_0} z_{ijt} = y_{it} \qquad \forall i \in \mathcal{I} \tag{A.2e}$$

$$\sum_{i\in\mathcal{I}} f_{0it} = \sum_{i\in\mathcal{I}} y_{it} \tag{A.2f}$$

$$\sum_{j\in\mathcal{I}_0\setminus\{i\}} f_{jit} - \sum_{j\in\mathcal{I}_0\setminus\{i\}} f_{ijt} = y_{it} \qquad \forall i \in \mathcal{I} \tag{A.2g}$$

$$f_{ijt} \le I z_{ijt} \qquad \forall (i,j) \in \mathcal{A}_0 \tag{A.2h}$$

$$x_{it} \ge 0,\ y_{it} \in \{0,1\} \qquad \forall i \in \mathcal{I} \tag{A.2i}$$

$$f_{ijt} \ge 0,\ z_{ijt} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}_0 \tag{A.2j}$$

We use single-commodity flow formulation (A.2f)–(A.2h) to ensure the validity of cycles. If the optimal objective value of ($\mathtt{SP}_t$) is smaller than σ_t, that is, if a pattern with the negative reduced cost is found, the pattern corresponding to the optimal solution of ($\mathtt{SP}_t$) is added to (MP). This procedure is repeated until no pattern is generated by ($\mathtt{SP}_t$) for all $t \in \mathcal{T}$.

Appendix B

Detailed Test Results in Chapter 2

In Appendix B, we report the detailed computational test results in Chapter 2.

Table B.1: Test results on single-period instances: LP gap (%)

Factors		*Extended Formulations*						*Valid Inequalities*			
		(SCF1)	(SCF2)	(MCF1)	(MCF2)	(TF1)	(TF2)	(GSEC)	(S-STAR)	(U-STAR)	(ALL)
I	5	29.72	29.61	29.21	23.14	22.12	22.12	29.21	22.12	23.58	21.60
	15	33.66	33.65	33.42	26.52	21.77	21.77	33.42	21.77	23.14	21.04
	25	33.87	33.87	33.73	27.46	20.93	20.92	33.73	20.92	22.43	20.23
	35	34.68	34.68	34.59	28.65	21.02	21.02	34.59	21.02	22.54	20.31
UTIL	60	32.55	32.52	32.31	26.10	21.24	21.23	32.31	21.23	22.70	20.60
	80	33.05	33.02	32.82	26.48	21.48	21.47	32.82	21.47	22.95	20.80
	100	33.35	33.32	33.09	26.75	21.67	21.67	33.09	21.67	23.12	20.98
SC	50	49.66	49.63	49.42	43.49	36.49	36.48	49.42	36.48	37.99	35.28
	100	16.30	16.28	16.05	9.40	6.43	6.43	16.05	6.43	7.86	6.31
IUB	F	48.49	48.45	48.25	36.94	29.44	29.44	48.25	29.44	33.20	29.44
	T	17.48	17.46	17.23	15.95	13.48	13.47	17.23	13.47	12.65	12.15
Average		32.98	32.95	32.74	26.44	21.46	21.46	32.74	21.46	22.92	20.79

Table B.2: Test results on single-period instances: Closed gap (%)

Factors		*Extended Formulations*						*Valid Inequalities*			
		(SCF1)	(SCF2)	(MCF1)	(MCF2)	(TF1)	(TF2)	GSEC	S-STAR	U-STAR	(ALL)
I	5	2.27	2.84	4.97	27.80	30.27	30.30	4.97	30.30	24.52	32.49
	15	0.59	0.63	3.19	28.25	40.47	40.50	3.19	40.50	38.49	43.99
	25	0.17	0.18	2.05	24.99	43.74	43.77	2.05	43.77	41.61	47.24
	35	0.11	0.11	1.55	22.73	45.42	45.45	1.55	45.45	43.51	48.96
UTIL	60	0.82	0.95	2.89	26.16	39.79	39.81	2.89	39.81	36.84	42.90
	80	0.65	0.80	2.91	25.29	39.56	39.59	2.91	39.59	36.67	42.86
	100	0.89	1.07	3.03	26.39	40.58	40.60	3.03	40.60	37.59	43.74
	50	0.24	0.33	1.03	10.94	24.80	24.84	1.03	24.84	24.05	28.94
	100	1.33	1.55	4.85	40.95	55.15	55.17	4.85	55.17	50.02	57.39
IUB	F	0.51	0.67	1.46	31.66	47.03	47.03	1.46	47.03	37.94	47.03
	T	1.06	1.21	4.41	20.23	32.92	32.98	4.41	32.98	36.13	39.31
Average		0.78	0.94	2.94	25.94	39.97	40.00	2.94	40.00	37.03	43.17

Table B.3: Test results on single-period instances: Ratio of instances with LP gap below certain value

LP Gap	*Extended Formulations*						*Valid Inequalities*			
	(SCF1)	(SCF2)	(MCF1)	(MCF2)	(TF1)	(TF2)	(GSEC)	(S-STAR)	(U-STAR)	(ALL)
$= 0\%$	6.04	6.04	6.35	8.00	8.56	8.58	6.35	8.58	8.27	9.10
$< 1\%$	7.00	7.06	7.56	10.10	12.06	12.06	7.56	12.06	11.96	13.06
$< 5\%$	15.17	15.21	16.06	21.73	26.02	26.02	16.06	26.02	24.38	26.58
$< 10\%$	24.54	24.54	25.04	29.81	38.23	38.25	25.04	38.25	32.73	38.73

Table B.4: Test results on multi-period instances: LP strength

Factors		*Extended Formulations*						*Valid Inequalities*			
		(SCF1)	(SCF2)	(MCF1)	(MCF2)	(TF1)	(TF2)	(GSEC)	(S-STAR)	(U-STAR)	(ALL)
I	5	92.98	92.98	93.27	94.12	94.57	94.59	93.27	94.59	96.24	97.00
	15	97.31	97.31	97.43	97.69	98.06	98.07	97.43	98.07	98.35	98.63
	25	97.36	97.36	97.45	97.57	97.77	97.77	97.45	97.77	97.87	98.03
T	5	97.14	97.14	97.32	97.62	97.82	97.82	97.32	97.82	98.30	98.72
	15	96.06	96.06	96.22	96.64	97.01	97.02	96.22	97.02	97.73	98.12
	25	94.45	94.45	94.61	95.12	95.58	95.59	94.61	95.59	96.43	96.82
UTIL	60	99.58	99.58	99.70	99.72	99.62	99.62	99.70	99.62	99.80	99.93
	80	97.95	97.95	98.14	98.19	98.08	98.08	98.14	98.08	98.65	98.91
	100	90.11	90.12	90.32	91.47	92.70	92.71	90.32	92.71	94.02	94.83
SC	50	96.21	96.22	96.40	96.77	97.06	97.06	96.40	97.06	97.64	98.04
	100	95.55	95.55	95.70	96.15	96.55	96.55	95.70	96.55	97.34	97.74
IUB	F	98.53	98.53	98.53	99.25	99.94	99.95	98.53	99.95	99.74	99.95
	T	93.24	93.24	93.57	93.67	93.66	93.67	93.57	93.67	95.24	95.83
Average		95.88	95.88	96.05	96.46	96.80	96.81	96.05	96.81	97.49	97.89

Table B.5: Test results on multi-period instances: Ratio of instances with LP strength above certain value

LP Strength	*Extended Formulations*						*Valid Inequalities*			
	(SCF1)	(SCF2)	(MCF1)	(MCF2)	(TF1)	(TF2)	(GSEC)	(S-STAR)	(U-STAR)	(ALL)
$= 100\%$	4.72	4.72	4.72	21.57	34.26	41.30	4.72	41.30	14.44	41.67
$> 99\%$	51.11	51.11	53.70	55.83	66.20	66.20	53.70	66.20	67.22	74.07
$> 95\%$	74.35	74.35	74.81	80.56	80.28	80.28	74.81	80.28	83.43	84.26
$> 90\%$	88.06	88.06	88.43	88.80	88.89	88.89	88.43	88.89	90.28	91.76

Appendix C

Detailed Test Results in Chapter 3

In Appendix C, we report the detailed computational test results in Chapter 3.

Table C.1: Comparison of the LP bound of the models

Factors	*LP Gap* (%)				*Gap Closed* (%)		
	ST	TF	ST-FL	TF-FL	TF	ST-FL	TF-FL
Dim							
5×10	74.6	63.1	70.6	62.2	16.6	5.2	17.7
5×15	73.2	62.4	69.4	61.7	16.2	5.0	17.0
10×10	91.6	84.4	65.9	63.8	8.0	27.9	30.3
10×15	92.1	85.3	70.0	67.9	7.6	23.7	26.1
15×15	97.2	93.2	65.4	63.9	4.1	32.7	34.2
Setup Time							
S	77.4	70.3	51.1	47.3	11.0	29.0	35.7
M	86.1	77.9	69.8	65.2	10.7	17.4	23.8
L	93.7	84.8	83.9	79.1	9.8	10.2	15.6
Demand							
$(0.8, 1/3)$	85.8	77.7	67.7	63.3	10.2	19.7	25.6
$(0.8, 1/2)$	95.2	90.8	67.0	65.7	4.9	29.0	30.4
$(1, 1/3)$	76.8	64.9	68.2	60.7	17.3	10.0	22.0
$(1, 1/2)$	85.2	77.4	70.1	65.9	9.7	16.8	22.1
Total	85.7	77.7	68.3	63.9	10.5	18.9	25.1

Table C.2: Comparison of the computational performance of the models: *Gap* and *Opt*

Factors	*Gap* (%)				*Opt* (%)			
	ST	TF	ST-FL	TF-FL	ST	TF	ST-FL	TF-FL
Dim								
5×10	1.0	0.1	2.4	0.0	96.7	98.3	83.3	100.0
5×15	18.8	4.4	23.7	8.1	35.0	71.7	28.3	56.7
10×10	31.5	25.5	35.7	35.6	11.7	18.3	10.0	5.0
10×15	50.9	40.5	57.4	49.9	3.3	6.7	1.7	1.7
15×15	61.1	53.6	64.4	58.9	0.0	0.0	0.0	0.0
Setup Time								
S	18.4	11.4	21.1	17.2	41.0	54.0	39.0	28.0
M	35.3	25.4	39.8	31.5	24.0	32.0	18.0	21.0
L	44.3	37.7	49.3	42.8	23.0	31.0	17.0	29.0
Demand								
$(0.8, 1/3)$	30.2	21.0	32.4	27.4	36.0	46.7	33.3	26.7
$(0.8, 1/2)$	33.8	24.6	36.4	29.9	28.0	38.7	21.3	52.0
$(1, 1/3)$	30.4	24.8	36.3	29.6	29.3	38.7	26.7	38.7
$(1, 1/2)$	36.3	28.9	41.8	35.1	24.0	32.0	17.3	40.0
Total	32.7	24.8	36.7	30.5	29.3	39.0	24.7	32.7

Table C.3: Comparison of the computational performance of the models: *#Node* and *Time*

Factors	*# Node*				*Time* (s)			
	ST	TF	ST-FL	TF-FL	ST	TF	ST-FL	TF-FL
Dim								
5×10	306301	9609	123924	7899	103.7	37.0	179.2	59.6
5×15	814890	50286	150243	22390	450.4	305.1	504.3	391.4
10×10	209644	5562	67427	2456	555.2	539.0	563.7	579.0
10×15	113881	2186	12330	708	589.4	572.3	593.9	598.2
15×15	19461	37	2480	5	600.0	600.0	600.0	600.0
Setup Time								
S	212255	8141	52716	5549	396.3	340.6	420.1	411.1
M	327046	16186	81573	7321	483.9	440.5	510.6	464.7
L	339204	16281	79553	7204	498.9	450.9	533.9	461.2
Demand								
$(0.8, 1/3)$	257561	9636	72453	6037	423.6	368.5	446.3	418.2
$(0.8, 1/2)$	298199	17258	69246	7078	475.1	429.7	510.3	476.1
$(1, 1/3)$	271736	9605	77482	5814	444.9	402.1	478.3	413.9
$(1, 1/2)$	343845	17645	65942	7836	495.2	442.5	518.0	474.4
Total	292835	13536	71281	6691	459.7	410.7	488.2	445.6

Table C.4: Test results for the LPNF algorithm

γ^{up}	δ	*Relative Solution Value* (%)				*Time* (s)			
		ST	TF	ST-FL	TF-FL	ST	TF	ST-FL	TF-FL
0.7	1	263.0	178.3	120.3	106.5	3.5	6.9	3.9	6.8
	2	276.6	175.5	138.7	116.6	2.5	4.8	2.6	4.7
	3	496.2	179.7	245.2	126.7	2.4	3.9	2.1	4.0
	4	423.7	182.7	201.4	132.8	1.9	3.3	1.8	3.4
	5	330.9	189.8	202.8	133.8	1.7	3.0	1.6	3.1
0.9	1	253.2	170.2	123.1	107.2	3.6	6.9	3.8	6.7
	2	280.3	174.4	140.1	116.1	2.5	4.7	2.6	4.7
	3	488.5	180.6	240.6	129.9	2.5	3.8	2.1	3.9
	4	432.3	185.6	200.9	133.7	2.5	3.3	1.8	3.4
	5	328.4	186.9	201.4	133.0	2.8	2.9	1.6	3.1
Average		357.3	180.4	181.5	123.6	2.6	4.4	2.4	4.4

국문초록

공급망의 생산 계획 단계에서의 주요한 두 가지 단기 의사결정 문제인 Lot-sizing 문제와 Scheduling 문제가 통합된 문제인 Lot-sizing and scheduling problem (LSP)은 계획 대상기간 동안 주어진 복수의 제품에 대한 수요를 최소의 비용으로 만족시키기 위한 단위 기간 별 최적의 생산량 및 생산 순서를 결정한다. 공급망 내의 다양한 요소에 대한 통합적 의사 결정의 중요성이 커짐에 따라 LSP에 대한 관심 역시 산업계와 학계 모두에서 지속적으로 증가하였다. 그러나 최근 수십 년에 걸친 활발한 연구에도 불구하고, 문제 자체가 내포하는 이론적 복잡성 및 실제 산업 환경과 제조 공정의 고도화/복잡화 등으로 인해 LSP를 해결하는 것은 여전히 어려운 문제로 남아있다. 특히 순서의존적 작업준비가 있는 경우 문제가 더욱 어려워진다는 것이 잘 알려져 있다.

본 논문에서는 순서의존적 작업준비가 있는 LSP를 해결하기 위한 정수 최적화 및 근사 동적 계획법 기반의 해법을 제안한다. 먼저, 이론적으로 강성 NP-hard에 속한다는 사실이 잘 알려진 LSP의 근본 구조에 대한 이해를 높이기 위하여 단일 기간만을 고려하는 부분구조에 대해 다룬다. 단일 기간 부분구조에 의해 정의되는 다면체에 대한 이론적 분석을 통해 새로운 유효 부등식 군을 도출하고 해당 유효 부등식들이 극대면(facet)을 정의할 조건에 대해 밝힌다. 또한, 도출된 유효 부등식들이 다항시간 내에 분리 가능함을 보이고, 최대흐름문제를 활용한 다항시간 분리 알고리듬을 고안한다. 실험결과를 통해 제안한 유효 부등식들이 모형의 선형계획 하한강도를 높이는 데 큰 영향을 줌을 확인한다. 또한 해당 부등식들이 모두 추가된 모형과 이론적으로 동일한 하한을 제공하는 확장 수리모형(extended formulation)을 도출한다. 이를 활용하여, 실제 산업에서 발생하는 LSP에서 종종 고려하는 주요한 추가 요소들을 다룰 수 있는 새로운 수리모형을 제안하며 해당 모형이 기존의 모형에 비해 더욱 강한 선형계획 하한을 제공함을

보인다. 이 모형을 바탕으로 빠른 시간 내에 가능해를 찾을 수 있는 선형계획 기반 휴리스틱 알고리듬을 개발한다. 마지막으로 해당 문제에 대한 근사 동적 계획법 알고리듬을 제안한다. 제안하는 알고리듬은 가치함수 근사 기법을 활용하며 특정 상태의 가치를 근사하기 위해 해당 상태에서의 근사함수의 상한 및 하한을 활용한다. 이 때, 좋은 상한 및 하한값을 구하기 위해 제안된 모형의 선형계획 완화문제와 선형계획 기반 휴리스틱 알고리듬을 사용한다. 실험 결과를 통해 제안한 알고리듬이 기존의 방법들과 비교하여 우수한 성능을 보임을 확인한다.

주요어: 생산계획 문제, 순서의존적 작업준비, 정수 최적화, 근사 동적 계획법, 유효 부등식, 확장 수리모형

학번: 2018-32331