THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à l'Université Paris-Dauphine

Complex lot Sizing problem with parallel machines and setup carryover

École Doctorale de Dauphine — ED 543

Spécialité **Informatique**

COMPOSITION DU JURY :

Mme Virginie GABREL
Université Paris-Dauphine
Directrice de thèse

M. Fabio FURINI
Université Paris-Dauphine
Membre du jury

M. Filippo FOCACCI
DecisionBrain
Membre du jury

M. Stéphane DAUZERE-PERES
Ecole des Mines de Saint-Etienne
Membre du jury

Mme Safia KEDAD-SIDHOUM
Université Paris 6
Rapporteure

M. Chengbin CHU
ESIEE Paris
Rapporteur

M. Roberto WOLFLER CALVO
Université Paris 13
Membre du jury

M. Vincent GIARD
Université Paris-Dauphine, EMINES
Président du jury

Soutenue le 28.11.2017 par Xueying SHEN

Dirigée par **Virginie GABREL**

DAUPHINE UNIVERSITÉ PARIS | PSL RESEARCH UNIVERSITY PARIS

Abstract

In this thesis, we study two production planning problems motivated by challenging real-world applications.

In the first part of this manuscript, a production planning problem for an apparel manufacturing company is studied and an optimization tool is developed to tackle it. We propose a decomposition framework composed by an aggregated model and a detailed model, which are solved in sequence. The aggregated problem is shown to be the bottleneck of the approach, which corresponds to a complex capacitated lot sizing problem with setup carryover, parallel machines, production time windows, backlogging and lost sales. This problem is shown to be NP-hard even without the setup costs. Several Mixed Integer Programming (MIP) formulations are proposed and compared from a theoretical and a computational point of view. Moreover, several constructive and local search heuristics are developed to find good quality solutions for large scale instances. We propose two sets of benchmark instances to evaluate the performances of the models and the heuristics. Thanks to extensive computational tests, we showed that the constructive heuristic (called First-Solution Heuristic) together with a Fix&Optimize algorithm is able to compute the best solutions in terms of optimality gap. Finally, the whole production planning approach is presented and its performance is analyzed.

In the second part of this manuscript, a restricted version of the capacitated lot sizing problem with sequence dependent setups is studied, where the setup sequences for each time bucket have to follow the order of a given sequence. Compared to the capacitated lot sizing problem with sequence dependent setup, the new model reduces the number of candidate setup sequences from $O(n!)$ to $O(n2^n)$ where n is the number of products. This problem is shown to be NP-hard. A special case with only two possible setup values is studied and we prove that also in this case the problem remains NP-hard. Moreover, product-oriented and sequence-oriented MIP formulations are developed. A column generation heuristic is also proposed based on the sequence-oriented formulations. Finally, we perform computational tests to evaluate their respective computational performance.

To my dearest grandfather Shen Fengzhi.

"Don't worry, Gromit. Everything's under control!"

— *The Wrong Trousers,* Aardman Animations, *1993*

Acknowledgments

I would like to thank my supervisors Dr. Virginie Gabrel, Dr. Fabio Furini, Dr. Filippo Focacci and Mr. Daniel Godard. You gave this wonderful chance to pursue my study and guide me through my research with tremendous patience and profound knowledge. Without your supervision and constant help, this dissertation would not have been possible.

Deep thanks are owed to Dr. Stéphane Dauzère-Pérès, I would not have accomplished what I have without your insight and expertise that greatly assisted the research. My deep gratitude to the thesis committee member Dr. Safia Kedad-Sidhoum, Dr. Chengbin Chu, Dr. Roberto Wolfler Calvo and Dr. Vincent GIARD, for your precious time and valuable comments on this work.

I would like to express my appreciation to all of my colleagues, it is a great pleasure to work with such cheerful and talented people like you. Special thanks to Kiat Shi Tan, Issam Mazhoud, David Gravot, Giulia Burchi and Désiree Rigonat that you generously shared your experience and knowledge with me and accompanied me through my PhD.

Finally, I would like to express my sincere gratitude to my parents, my husband Alexandre Menif and my dear friends, my deep love to all of you.

Contents

Abstract **ii**

Acknowledgments **v**

Contents **vi**

List of Tables **viii**

List of Figures **x**

1 Industrial and Scientific Context **1**

1.1 Introduction 2

1.2 Production Planning: Lot Sizing Problem 4

1.3 Production Planning in An Apparel Manufacturing Application 12

1.3.1 An Apparel Manufacturing Application 13

1.3.2 Production Planning Problem Modeling 16

1.4 Contributions 17

2 Complex Capacitated Lot Sizing Problem: Formulations and Benchmarks **18**

2.1 Problem Definition 19

2.2 Literature Review 22

2.3 MIP Formulations 28

2.4 Benchmark Instances 35

2.4.1 Benchmark IAP: Real-World Instances and Data Analysis 35

2.4.2 Benchmark IRG: Pseudo-Randomly Generated Instances 43

2.5 Empirical Evaluations 48

2.5.1 MIP Formulation Comparison Considering Setup Cost 48

2.5.2 MIP Formulation Comparison without Considering Setup Cost . . 52

2.5.3 Impact Analysis of Problem Features . . . 54
2.5.4 Computational Results on Benchmark IAP-B and IRG-B . . . 56
2.6 Conclusions . . . 62

3 Complex Capacitated Lot Sizing Problem: Heuristics 63
3.1 Introduction . . . 64
3.2 Constructive Heuristic Algorithms . . . 64
3.2.1 Fix&Relax Algorithm . . . 64
3.2.2 Product Decomposition Based Algorithm . . . 67
3.2.3 First Solution Heuristic Algorithm Based on LP Relaxation . . . 75
3.3 Fix&Optimize algorithm . . . 81
3.4 Computational Results . . . 81
3.4.1 Algorithm Parameter Evaluation . . . 82
3.4.2 Algorithm Comparison Results . . . 86
3.5 Conclusions . . . 99

4 Production Planning Solution to the Apparel Application 100
4.1 Decomposition Approach . . . 101
4.2 Application Performance Analysis . . . 105
4.3 Conclusions . . . 106

5 Capacitated Lot Sizing Problem with A Fixed Product Sequence 107
5.1 Capacitated Lot Sizing Problem with Sequence Dependent Setup . . . 108
5.2 Problem Definition . . . 110
5.3 Problem Formulation . . . 116
5.4 A Special Case Study . . . 122
5.5 Column Generation Approach . . . 127
5.6 Computational Results . . . 130
5.7 Conclusions . . . 133

6 General Conclusion and Future Work 134

Appendix A Data Analysis and Computational Results 137
A.1 CLSC Computational Results . . . 139
A.2 CLSP-FS1 Computational Results . . . 149

List of Tables

1.1 Production planning example: demands 2
1.2 Production planning example: minimum manufacturing cost solution . . . 3
1.3 Production planning example: minimum inventory cost solution 3
1.4 Production planning example: minimum overall cost solution 3
1.5 Overview of review papers of deterministic dynamic LSP 7
1.6 Apparel manufacturing production: learning curve example 15
1.7 Modeling apparel production planning to lot sizing problem 16

2.1 CLSC Example 2.1 data: setups 21
2.2 CLSC Example 2.1 data: capacities 21
2.3 CLSC formulation size comparison 32
2.4 CLSC benchmark instances summary 35
2.5 CLSC real-world benchmark instances 36
2.6 CLSC pseudo-randomly generated benchmark instances 43
2.7 CLSC instance generator parameters 43
2.8 CLSC formulation comparison with setup cost 51
2.9 CLSC formulation comparison without setup cost 53
2.10 CLSC feature - complexity analysis 55
2.11 Computational results: CPLEX on IAP-B 56
2.12 Computational results: CPLEX on IRG-B (1) 57
2.13 Computational results: CPLEX on IRG-B (2) 58
2.14 Computational results: CPLEX on IRG-B (3) 59

3.1 PD algorithm: sorting criteria 68
3.2 PD algorithm: sub-problem 69
3.3 Comparison results: F&R algorithm variations 83
3.4 Comparison results of PD algorithm variations: sub-problems 84
3.5 Comparison results of PD algorithm variations: sorting criteria 85

3.6 Computational results: FSH algorithm on IAP-B 86
3.7 Computational results: FSH algorithm on IRG-B (1) 88
3.8 Computational results: FSH algorithm on IRG-B (2) 89
3.9 Computational results: FSH algorithm on IRG-B (3) 90
3.10 Computational results: heuristic algorithm on IAP-B 92
3.11 Computational results: heuristic algorithm on IRG-B summary 93
3.12 Computational results: heuristic algorithm on IRG-B summary by type . 94
3.13 Computational results: heuristic algorithm on IRG-B (1) 96
3.14 Computational results: heuristic algorithm on IRG-B (2) 97
3.15 Computational results: heuristic algorithm on IRG-B (3) 98

4.1 Detailed and aggregated model in planning phase 103
4.2 Application planning solution evaluation 106

5.1 CLSP-FS1 Example 5.1 data 113
5.2 Theorem 5.2 proof CLSP-FS1-LT instance parameters 123
5.3 Computational results: CLSP-FS1 formulation comparison 130
5.4 Computational results: column generation heuristic based on AG-SO . . . 131
5.5 Computational results: CLSP-FS1-LT formulation comparison 132

List of Figures

1.1 DecisionBrain production planning applications 4
1.2 Technical structure of lot sizing problem in Glock et al. [55] 6
1.3 Apparel manufacturing application: plant structure 13
1.4 Apparel manufacturing application: production procedure 14

2.1 Setup carryover 20
2.2 Time window of demand 20
2.3 CLSC Example 2.1 data: time windows 21
2.4 CLSC Example 2.1 optimal solution with setup cost 22
2.5 CLSC instance R5 analysis: machine capacity distribution 37
2.6 CLSC instance R5 analysis: production time distribution 38
2.7 CLSC instance R5 analysis: setup time distribution 38
2.8 CLSC instance R5 analysis: product-demand distribution 39
2.9 CLSC instance R5 analysis: demand quantity distribution 39
2.10 CLSC instance R5 analysis: demand time window distribution 40
2.11 CLSC instance R5 analysis: demand release/due date distribution 41
2.12 CLSC instance R5 analysis: capacity requirement by time interval 42
2.13 CLSC instance R5 analysis: tardiness cost distribution 42
2.14 LPR time on different types of instances 60
2.15 MIP gap on different types of instances 61
2.16 MIP number of nodes on different types of instances 61

3.1 FR-T algorithm procedure Absi and Kedad-Sidhoum [3] (modified) 66
3.2 PD algorithm flow chart 68
3.3 PD-SI algorithm capacity update example 73
3.4 FSH algorithm flow chart 76
3.5 CLSC Example 2.1 optimal solution (no setup cost) 77
3.6 CLSC Example 2.1 optimal LP relaxation solution (no setup cost) 77

3.7 PD algorithm gap and computational time on pilot benchmark 85
3.8 FSH algorithm computational time on different types of instances 91
3.9 Heuristic algorithm comparison results on pilot instances 93

4.1 Apparel manufacturing decomposition approach 101

5.1 Color change in production 111
5.2 Definition 5.1 illustration 112
5.3 CLSP with sequence dependent setup Example 5.1 optimal solution . . . 113
5.4 CLSP-FS1 Example 5.1 optimal solution 113
5.5 Theorem 5.1 CLSP-FS1 instance optimal solution structure 114
5.6 CLSP-FS1 graph representation of setup sequence 119
5.7 Theorem 5.2 reduction from CLSP with single product to CLSP-FS1-LT . 123
5.8 CLSP-FS1 network representation of the pricing problem 128

Chapter 1

Industrial and Scientific Context

In this manuscript, our research focuses on production planning. The problem is to optimize production plan in manufacturing industry to achieve high customer service level and cost efficiency. Due to our industrial background, we have been exposed to various types of real-world applications. Therefore, all our research is motivated by realistic requirements that we have encountered in building industrial production planning solutions. First, we have studied a production planning problem from apparel manufacturing industry. This problem is brought to our attention by a project that we have worked with a market leader in the apparel industry, which produces 60% of the T-shirts sold in the US. Our research result, including modeling and algorithm design, has been implemented inside the engine of their production planning and scheduling software and improves the daily production efficiency. Second, we have studied a restricted version of a classical production planning problem: capacitated lot sizing problem with sequence dependent setups, which is known to be hard to solve. This newly proposed model considers the planners knowledge in certain industries and therefore simplifies the classical model. By doing so, there is a better chance to deliver reasonable production planning solutions for industries where our model is applicable.

This chapter is organized as follows: In Section 1.1, we present our research background and introduce our study interest at production planning. In Section 1.2, we describe a general picture of production planning, i.e., lot sizing problem. In Section 1.3, the real-world application of production planning problem in apparel manufacturing is introduced. Finally, we summarize major contributions of our research in Section 1.4 as a reading guide for the rest of this manuscript.

1.1 Introduction

Our research is performed under the CIFRE program (Conventions Industrielles de Formation par la REcherche) [30]. Therefore, it is a collaboration between Paris Dauphine University and DecisionBrain (https://www.decisionbrain.com). DecisionBrain is a software company that delivers advanced analytics and optimization solutions to innovative companies who want to apply a scientific approach to decision making. Building production planning and scheduling solution is one of DecisionBrain's expertise.

Generally speaking, production planning is to *decide* the production for a future period of time (planning horizon) *given* limited resources and/or production restrictions to *achieve* optimal customer service level and cost efficiency. Here is a small example modified from [108] to illustrate the concept. An apparel manufacturer produces different types of costumes. One specific type of costumes requires a high setup cost due to the special technique and equipment needs, therefore at most one batch can be produced in one month. Given 200 units of stock at the end of the year, the goal is to plan the production of this costume for the next 8 months (January to August) to minimize the cost while satisfy all forecasted demands. The cost includes: setup cost as $5000 if there is a positive production in a month, unitary processing cost $100, and unitary inventory cost $5. The demand forecast for the next 8 months is given in Table 1.1 and we need to decide the production quantity for each month.

Table 1.1: Production planning example: demands

Jan	Feb	Mar	Apr	May	Jun	Jul	Aug
400	400	800	800	1200	1200	1200	1200

If it is only to minimize the manufacturing cost (setup cost and production cost), we can produce only once in January to satisfy total demands till August. The solution is given in Table 1.2 and the total cost equals to $859,000. This cost includes $700,000 (7,000 x 100) processing cost, $154,000 (30,800 x 5) inventory cost and $5,000 (5,000 x 1) setup cost. If it is only to minimize the inventory cost, we can follow the just-in-time rule and produce in each month the amount it requires. The total cost becomes $740,000 and the solution is given in Table 1.3. However, if it is to minimize the overall cost, the optimal solution has total cost equals to $736,000 and the optimal solution is given in Table 1.4. In the optimal solution, there are two months (February and April) that have no production.

Table 1.2: Production planning example: minimum manufacturing cost solution

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Total
Demand	400	400	800	800	1,200	1,200	1,200	1,200	7,200
Production	7,000	0	0	0	0	0	0	0	7000
Proc. cost	700,000	0	0	0	0	0	0	0	700,000
Setup cost	5,000	0	0	0	0	0	0	0	5,000
Inventory	6,800	6,400	5,600	4,800	3,600	2,400	1,200	0	30,800
Inv. cost	34,000	32,000	28,000	24,000	18,000	12,000	6,000	0	154,000

*Initial inventory = 200

Table 1.3: Production planning example: minimum inventory cost solution

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Total
Demand	400	400	800	800	1,200	1,200	1,200	1,200	7,200
Production	200	400	800	800	1,200	1,200	1,200	1,200	7000
Proc. cost	2,000	4,000	8,000	8,000	12,000	12,000	12,000	12,000	700,000
Setup cost	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	40,000
Inventory	0	0	0	0	0	0	0	0	0
Inv. cost	0	0	0	0	0	0	0	0	0

*Initial inventory = 200

Table 1.4: Production planning example: minimum overall cost solution

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Total
Demand	400	400	800	800	1,200	1,200	1,200	1,200	7,200
Production	600	0	1,600	0	1,200	1,200	1,200	1,200	7,000
Proc. cost	60,000	0	160,000	0	120,000	120,000	120,000	120,000	700,000
Setup cost	5,000	0	5,000	0	5,000	5,000	5,000	5,000	30,000
Inventory	400	0	800	0	0	0	0	0	1,200
Inv. cost	2,000	0	4,000	0	0	0	0	0	6,000

*Initial inventory = 200

Even in this toy example, we can have an insight of the benefit that production planning may bring to the industry. Kellogg Company reports annual cost savings of 4 million dollars by performing optimization to plan the production and distribution decisions for its cereal and convenience foods business [108]. Thanks to the development of IT technology and commercial optimization software, it becomes possible to tackle large scale real-world production planning problems using optimization. Therefore, more and more companies start to realize that introducing scientific method to optimize the

decision process could have a non-negligible impact on their profits and competences. Another reason of the tremendous interest shown in literature in production planning is that different manufacturing industry implies different production planning problems. Therefore production planning problems occur with many variations each with its own complexity and challenges. For instance, to the best of our knowledge, production planning problems studied in this manuscript have never been addressed before.

There are several production planning projects explored in DecisionBrain from different industries, such as apparel manufacturing, semi conductor assembling and testing, and disposable table-ware production (Figure 1.1). Research presented in this manuscript is mainly motivated by the project with an apparel company, which is to build a production planning and scheduling software to arrange mid term and short term productions. The details of this application is presented in Section 1.3. But first of all, we give a general introduction on the production planning problem in Section 1.2, which is also referred as lot sizing problem.



Figure 1.1: DecisionBrain production planning applications

1.2 Production Planning: Lot Sizing Problem

Lot sizing problem

Lot Sizing Problem (LSP) is to plan production resources and activities, especially determine production quantities, to achieve the economical cost and/or more intangible

objectives such as customer service level. The history of LSP can be traced back to the publication of Harris [71], which proposes the Economic Order Quantity (EOQ) model. This problem has continuous time model with infinite time horizon, and all parameters such as demand quantity and inventory holding cost are constant. The solution of this problem can be obtained by a formula directly. Later, different extensions of EOQ have been studied such as Economic Lot Scheduling Problem (ELSP), which extends the problem to multi-item and considers capacity constraints. It is shown to be NP-hard in Gallego and Shaw [52]. However, both EOQ and ELSP consider constant parameters, which is not always the case in real applications. The Wagner-Whitin (WW) model was studied in the seminal papers of Wagner and Whitin [131] and Manne [96] in late 50's. In this model, the planning horizon is decomposed into time buckets and demand quantities vary with time buckets. Therefore, the WW model extends constant parameters to dynamic parameters varying with time and thus is referred to dynamic LSP.

The WW model is defined as follows: Given a planning horizon with T time buckets, let d_t be the product demand quantity for each time bucket $t \in \{1, 2, \ldots, T\}$. The unit inventory holding cost is h_t. Moreover, in each time bucket, to produce a positive quantity of products, there is a setup cost sc_t. The problem is to determine the production quantity in each time bucket so that all demands are satisfied with minimum cost. This problem can be formulated as follows:

$$\begin{aligned} \min \quad & \sum_{t=1}^{T} h_t I_t + sc_t z_t \\ s.t. \quad & I_{t-1} + x_t = d_t + I_t \quad t \in \{1, 2, \ldots, T\} \\ & I_0 = I_T = 0 \\ & x_t \leq b_t z_t \quad t \in \{1, 2, \ldots, T\} \\ & x \in \mathbb{R}_+^T, I \in \mathbb{R}_+^{T+1}, z \in \{0, 1\}^T \end{aligned} \tag{1.1}$$

where b_t is the maximum production quantity in time bucket t.

Different classification schemes are used in literature reviews of LSP such as De Bodt et al. [33], Drexl and Kimms [40], Staggemeier and Clark [123] and Guner Goren et al. [64]. One of the classification scheme divide LSP from two dimensions: models with stationary or dynamic parameters, models with deterministic or stochastic parameters (see Figure 1.2). According to this classification scheme, the EOQ and ELSP will lie in the stationary model whereas the WW model and its extensions lie in the dynamic model.

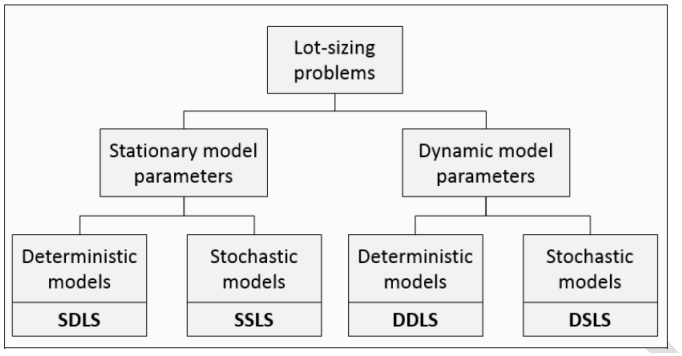


Figure 1.2: Technical structure of lot sizing problem in Glock et al. [55]

In this manuscript, our focus is the deterministic extensions of the WW model, which is Deterministic Dynamic LSP with finite time horizon (DDLS). Giving a comprehensive survey on the literature of DDLS becomes an "impossible mission" due to the flourish research in this domain. For more than half a century development of DDLS, more than 30 literature review papers and books have been published and even two reviews of literature reviews on production planning and inventory management are given by Glock et al. [55] and Guiffrida et al. [62]. To avoid duplication but still provide a set of references for interested readers, we summarize the survey papers related to DDLS in Table 1.5. They are from Glock et al. [55] and Guiffrida et al. [62] together with several papers and books that we believe worth mentioning.

Some papers are not included in the table since their focus is not lot sizing problem. For example, Gelders [53] mainly focuses on the state of the art progress in the production planning, where LSP takes only one section. It covers the WW model, multi-level uncapacitated LSP, capacitated LSP and ELSP, mainly from the perspective of heuristic algorithms. Bitran and Yanasse [17] studies the capacitated LSP and gives complexity analysis over various cost structures. Nahmias [101] gives an overview of the perishable ordering policy, only one paper about deterministic LSP is mentioned, which has random decay.

Table 1.5: Overview of review papers of deterministic dynamic LSP

Reference	Content	Classification
Aggarwal [5]	General view of inventory models.	Dynamic/static model, number of items, number of locations and echelons, characteristics of demand, research objective.
De Bodt et al. [33]	Dynamic LSP with constant costs over time.	Fixed/rolling horizon, deterministic/probabilistic model, single/ multi level.
Bahl et al. [13]	General review from both practitioner and research based literature.	Single/multi level, unconstrained/constrained resources.
Aksoy and Selcuk Erenguc [7]	Multi-item single stage inventory systems with joint setup costs.	Deterministic (static/dynamic models) and stochastic (continuous review/periodic review).
Maes and Wassenhove [95]	Classification and computational review on heuristic algorithms of the multi-item single-level capacitated LSP.	Single-resource heuristics (special-purpose methods), and mathematical-programming-based heuristics (general).
Zoller and Robrade [137]	A review and experimental comparison of algorithms of the WW model with rolling horizon.	
Gupta and Keung [66]	Multi-stage lot-sizing.	Constant/dynamic demands, rolling horizon.
Raafat [111]	Mathematical modelling of deteriorating inventory system especially deterioration as a function of the on-hand level of inventory.	Deteriorating features, LSP features.
Kuik et al. [91]	General view on models.	Strategic/tactical/operational, modeling elements in the LSP (such as planning horizon, static/dynamic demands).
Wolsey [132]	Single item uncapacitated LSP.	
Benton and Park [16]	LSP with several types of discount schemes.	Four discount types are surveyed from both the buyer point of view and the supplyer point of view.
Drexl and Kimms [40]	General review of the LSP and scheduling.	Single/multi level. For the single level, discrete time and continuous time model are considered.
Goyal and Giri [61]	The deteriorating inventory literature review as a continuation of Raafat [111].	Shelf life, demand variations and other conditions or constraints.
Rizk and Martel [113]	Material flow planning in a supply chain, and in particular with deterministic lot-sizing methods.	Single/multiple facility, single/multiple level, single/ multiple items, capacitated/uncapacitated, deterministic/ stochastic, static/dynamic demand.

Table 1.5: Overview of review papers of deterministic dynamic LSP (continue)

Reference	Content	Classification
Staggemeier and Clark [123]	LSP and scheduling models and its algorithms.	Time period, multi machines and other constraints.
Karimi et al. [88]	Models and algorithms of capacitated LSP with single level.	Planning horizon, number of levels, number of products, capacity or resource constraints, deterioration items, static/dynamic/deterministic/stochastic demand, setup structure, inventory shortage.
Brahimi et al. [22]	Single item LSP.	Big/small time buckets, uncapacitated (extensions such as backlogging, multiple facilities) and capacitated (different cost structures).
Pochet and Wolsey [108]	Mixed integer programming formulations for the LSP and its variants	
Zhu and Wilhelm [136]	LSP and scheduling with sequence dependent setup.	
Jans and Degraeve [84]	An overview of the use of meta-heuristics for solving LSP.	Algorithm representation, evaluation, neighborhood definition and genetic operators.
Jans and Degraeve [85]	Modeling deterministic single-level dynamic LSP based on various industrial extensions.	Basic LSP models and their extensions from two directions: modeling the operational aspects in more details, or is more towards tactical and strategic models.
Quadt and Kuhn [109]	Extensions of the capacitated LSP: back-orders, setup carry-over, sequencing, and parallel machines.	Back-orders, setup carry-over, sequencing, and parallel machines.
Allahverdi et al. [8]	LSP with setup cost and setup times.	
Robinson et al. [114]	Updates the review by Aksoy and Selcuk Erenguc [7] of the coordinated LSP.	Single/multiple items, coordinated/uncoordinated setup cost structures, capacitated/uncapacitated.
Buschkühl et al. [25]	Mainly survey the algorithms for the dynamic capacitated LSP for single level and multi level.	Mathematical programming heuristics, Lagrangian heuristics, decomposition and aggregation heuristics, metaheuristics, problem-specific greedy heuristics.
Guner Goren et al. [64]	A review of applications of genetic algorithms in LSP.	Static/dynamic, single/multi level, capacitated/uncapacitated

Capacitated Lot sizing problem

Among all problems in the domain of DDLS, our focus is at one type of LSP which considers the limited resource/machine capacity, called *Capacitated Lot Sizing Problem* (CLSP). Considering different features and cost structures will lead to different types of CLSP. Based on problems studied in this manuscript, we introduce the CLSP model with following parameters:

- $\mathcal{N} = \{1, 2, \ldots, N\}$ a set of N products.
- $\mathcal{T} = \{1, 2, \ldots, T\}$ a set of T time periods.
- cap_t: machine capacity in each time period $t \in \mathcal{T}$.
- d_{it}: demand of each product $i \in \mathcal{N}$ in time period $t \in \mathcal{T}$.
- pt_i: unitary production time of each product $i \in \mathcal{N}$.
- hc_{it}: unitary inventory cost of each product $i \in \mathcal{N}$ in time period $t \in \mathcal{T}$.
- b_{it}: maximum amount of production i that can be produced in $t \in \mathcal{T}$.
- st_i: setup time to product $i \in \mathcal{N}$.
- sc_i: setup cost to product $i \in \mathcal{N}$.

CLSP is to decide the production quantity of each product in each time bucket so that all demands are satisfied with a minimum total cost while respecting machine capacities.

Due to the capacity constraints, the problem is shown to be NP-hard even when there is only a single product by Florian et al. [47] and Bitran and Yanasse [17]. In the case of multiple products, Chen and Thizy [28] proved that it is strongly NP-hard. Karimi et al. [88] have done a nice survey focusing on CLSP with production cost and its solution approaches. Quadt and Kuhn [109] have provided a survey of CLSP with extensions including back-orders, setup carryover, sequencing and parallel machine.

Developing mathematical formulations is the very first step of our research since problems studied in this manuscript have not been studied before to the best of our knowledge. Therefore, in this section, we recall three Mixed Integer Programming (MIP) formulations of CLSP that have been studied and often adapted to other extensions of CLSP in the literature. These formulations are aggregated formulation, facility location formulation and network formulation.

Aggregated (AG) formulation is an intuitive formulation and was proposed by Trigeiro et al. [129]. We introduce following variables for each product $i \in \mathcal{N}$ and time bucket $t \in \mathcal{T}$:

- $x_{it} \in \mathbb{R}_+$: quantity of product i produced in time bucket t;
- $I_{it} \in \mathbb{R}_+$: inventory of product i at the end of time bucket t;
- $z_{it} \in \{0, 1\}$: it equals to 1 if product i is produced in time bucket t, 0 otherwise.

The formulation is given as follows:

$$\min \quad \sum_{i \in \mathcal{N}, t \in \mathcal{T}} hc_{it} I_{it} + \sum_{i \in \mathcal{N}, t \in \mathcal{T}} sc_i z_{it} \tag{1.2}$$

$$s.t. \quad I_{i,t-1} + x_{it} = I_{it} + d_{it} \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{1.3}$$

$$\sum_{i \in \mathcal{N}} pt_i x_{it} + \sum_{i \in \mathcal{N}} sc_i z_{it} \leq cap_t \qquad t \in \mathcal{T} \tag{1.4}$$

$$x_{it} \leq b_{it} z_{it} \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{1.5}$$

$$x_{it}, I_{it} \geq 0, \; I_{i,0} = 0 \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{1.6}$$

$$z_{it} \in \{0, 1\} \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{1.7}$$

The objective function (1.2) is to minimize the total cost including inventory cost $\sum_{i \in \mathcal{N}, t \in \mathcal{T}} hc_{it} I_{it}$ and setup cost $\sum_{i \in \mathcal{N}, t \in \mathcal{T}} sc_i z_{it}$. Constraints (1.3) ensure material balance for each product i in each time bucket t that the total inflow (last bucket ending inventory and production quantity) equals to the outflow (demand d_{it} and ending inventory). Constraints (1.4) guarantee the capacity usage does not exceed the available capacity in each time bucket. Finally, constraints (1.5) link production with setup: there is only a production if there is a corresponding setup for each product in each time bucket.

Facility Location (FL) formulation was first proposed by Krarup and Bilde [90] for cases without capacity restrictions. It is also referred as the transportation problem formulation [34] and simple plant location formulation [90]. Later it has been adapted to other LSP [124]. We introduce decision variables as follows for each product $i \in \mathcal{N}$, time bucket t, k such that $t \leq k \in \mathcal{T}$:

- $x_{itk} \in \mathbb{R}_+$: quantity of product i produced in time bucket t to satisfy demand in later time bucket k;
- $z_{it} \in \{0, 1\}$ is defined as before, it equals to 1 if product i is produced in time bucket t, 0 otherwise.

The formulation is given as follows:

$$\min \quad \sum_{i\in\mathcal{N},t\in\mathcal{T}} hc_{it}\left(\sum_{s\in\mathcal{T}:s\leq t}\sum_{k\in\mathcal{T}:t<k} x_{isk}\right) + \sum_{i\in\mathcal{N},t\in\mathcal{T}} sc_i z_{it} \tag{1.8}$$

$$s.t. \quad \sum_{k\in\mathcal{T}:k\leq t} x_{ikt} = d_{ik} \qquad i\in\mathcal{N}, t\in\mathcal{T} \tag{1.9}$$

$$\sum_{i\in\mathcal{N}} pt_i\left(\sum_{k\in\mathcal{T}:t\leq k} x_{itk}\right) + \sum_{i\in\mathcal{N},t\in\mathcal{T}} st_i z_{it} \leq cap_t \qquad t\in\mathcal{T} \tag{1.10}$$

$$x_{itk} \leq \min\{b_{it}, d_{ik}\} z_{it} \qquad i\in\mathcal{N}, t\in\mathcal{T}, t\leq k\in\mathcal{T} \tag{1.11}$$

$$x_{itk} \geq 0 \qquad i\in\mathcal{N}, t\in\mathcal{T} \tag{1.12}$$

$$z_{it} \in \{0,1\} \qquad i\in\mathcal{N}, t\in\mathcal{T} \tag{1.13}$$

There is a direct link between variables introduced in AG model and those introduced in FL formulation:

$$x_{it} = \sum_{k\in\mathcal{T}:t\leq k} x_{itk} \qquad I_{it} = \sum_{s\in\mathcal{T}:s\leq t}\sum_{k\in\mathcal{T}:t<k} x_{isk}$$

Therefore, the objective function is a simple substitution. Constraints (1.9) state that the total production quantity dedicated for demand d_{ik} equals to the demand quantity. Constraints (1.11) link the production quantity x_{itk} with the corresponding setup z_{it}. Here the upper bound of x_{itk} is no greater than the upper bound of x_{it}, which is the main reason that FL is stronger than AG formulation in the sense of better lower bound from Linear Programming (LP) relaxation. The price for the tighter lower bound is the number of variables, which is increased to $O(NT^2)$.

Network (NW) formulation was first proposed by Eppen and Martin [42]. It is also referred as shortest path/route formulation [34, 122]. We introduce decision variables as follows for each product $i\in\mathcal{N}$, time bucket t,k such that $t\leq k\in\mathcal{T}$:

- $u_{itk}\in[0,1]$: fraction of total demand from time bucket t through k of item i that is produced in t;
- $z_{it}\in\{0,1\}$ is defined as before, it equals to 1 if product i is produced in time bucket t, 0 otherwise.

We also define following constant for simplicity of the formulation:

$$D_{itk} = \sum_{v=t}^{k} d_{iv}, \quad H_{itk} = \sum_{v=t+1}^{k} h_{i,v-1} D_{ivk}, \quad I_{D_{itk}} = \begin{cases} 1 & \text{if } D_{itk} > 0 \\ 0 & \text{otherwise} \end{cases}$$

The formulation is given as follows:

$$\min \quad \sum_{i\in\mathcal{N},t\in\mathcal{T},k\in\mathcal{T}:k\geq t} H_{itk}u_{itk} + \sum_{i\in\mathcal{N},t\in\mathcal{T}} sc_i z_{it} \tag{1.14}$$

$$s.t. \quad \sum_{i\in\mathcal{N},k\in\mathcal{T}:k\geq t} pt_i D_{itk}u_{itk} + \sum_{i\in\mathcal{N},t\in\mathcal{T}} st_i z_{it} \leq cap_t \qquad t\in\mathcal{T} \tag{1.15}$$

$$\sum_{t\in\mathcal{T}} u_{i1t} = 1 \qquad i\in\mathcal{N} \tag{1.16}$$

$$\sum_{k\in\mathcal{T}:k\leq t-1} u_{i,k,t-1} = \sum_{k\in\mathcal{T}:k\geq t} u_{itk} \qquad i\in\mathcal{N}, 1<t\in\mathcal{T} \tag{1.17}$$

$$\sum_{k\in\mathcal{T}:k\leq t} I_{D_{itk}}u_{itk} \leq z_{it} \qquad i\in\mathcal{N}, t\in\mathcal{T} \tag{1.18}$$

$$u_{itk} \geq 0 \qquad i\in\mathcal{N}, t\in\mathcal{T} \tag{1.19}$$

$$z_{it} \in \{0,1\} \qquad i\in\mathcal{N}, t\in\mathcal{T} \tag{1.20}$$

The formulation is easier to understand if we see variables u_{itk} only as binary value. Then based on the proof in [42], the formulation is still valid when $u_{itk} \in [0,1]$. When $u_{itk} = 1$, its corresponding production quantity equals to D_{itk} and inventory cost equals to H_{itk}. The objective function (1.14) is still to minimize the inventory cost and setup cost. Constraints (1.15) ensure capacity restriction in time bucket t. Constraints (1.16) and (1.17) represent flow balance constraints for the source node and other nodes in the underlying network. Constraints (1.18) link production with its corresponding setup for each product i in time bucket t.

FL and NW models are stronger reformulation of AG model for CLSP. Nemhauser and Wosley [104] has shown that in the uncapacitated case, both LP relaxations of FL and NW define the convex hull of the problem. Denizel et al. [34] further proved the equivalence of the LP relaxations of FL and NW formulations for CLSP with constant unitary inventory cost h_i. They also point out that FL formulation has more constraints while its constraint matrix is less dense and has smaller coefficients, therefore different characteristics may be exploited to choose between these two reformulations.

In the next section, we start from another perspective and introduce the real-word application that motivates the first part of our research.

1.3 Production Planning in An Apparel Manufacturing Application

The main problem studied in this manuscript is brought to our attention by a project in manufacturing industry. The company is a market leader in the apparel industry,

which produces 60% of the T-shirts sold in the US. On-time delivery, low production and shipping cost is a critical competitive advantage for the company. To achieve this goal on such large scale, it is essential to optimize production planning and scheduling to perform efficient production. To model and solve the underlying production planning problem, we need to first understand the manufacturing procedure and bottlenecks in this particular case study. Hence, in Section 1.3.1, we first present the apparel company, its manufacturing procedure and bottlenecks. In section 1.3.2, we present the modeling of the problem, which leads to the CLSP studied in Chapters 2 - 4.

1.3.1 An Apparel Manufacturing Application

The company has 10 manufacturing plants over Asia, however, they are independent on the production planning and scheduling level. Therefore, the problem scope is considered as a single plant. A plant layout example is shown in Figure 1.3. In each plant, there are several work centers, each of which corresponds to a production operation. In other words, to produce one piece of clothes, it has to go through several work centers to finish. Moreover, there are normally more than one machine in each work center. Therefore, the production planning has to decide the production quantity for each production line. This implies parallel machines in the underlying lot sizing problem and leads to the first difficulty of the problem.

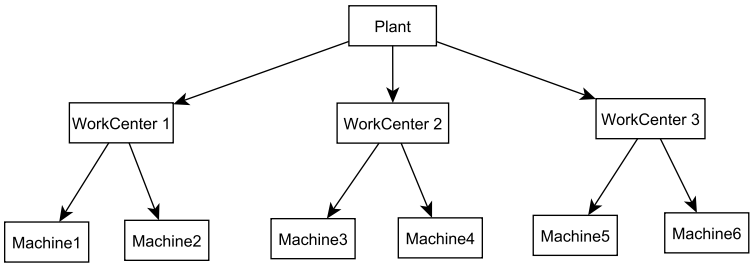


Figure 1.3: Apparel manufacturing application: plant structure

To produce one piece of clothes, all or some of following operations have to be done in sequence which includes *cutting, embroidery, sewing, washing, ironing/dipping, packing and cartoning*. In Figure 1.4, we show the entire manufacturing processes. Each product has a specific production routing. Some orders will route through all the operations, while some orders may skip certain operations (such as washing). On the other hand, some orders will go through the ironing operation, while some orders will go through dipping operation instead. Among all these processes, we could identify a bottleneck

step, which is the sewing process. This is not only because that it mainly depends on workers instead of machines but also because that it consumes most of the time during the production cycle. Due to this reason, we only model the production planning on the sewing process, which leads to the lot sizing problem with single level. Sewing process is the craft of fastening or attaching objects using stitches made with a needle and thread. There are about 20 to 60 sewing lines in each plant. Each sewing line is a group of sewing workers who share the same working shift schedule. Different sewing lines may have different number of workers and different working hours per day, which results in different machine capacities in the lot sizing problem. This together with multiple productions lines leads to the second difficulty because all parallel machines are not identical.

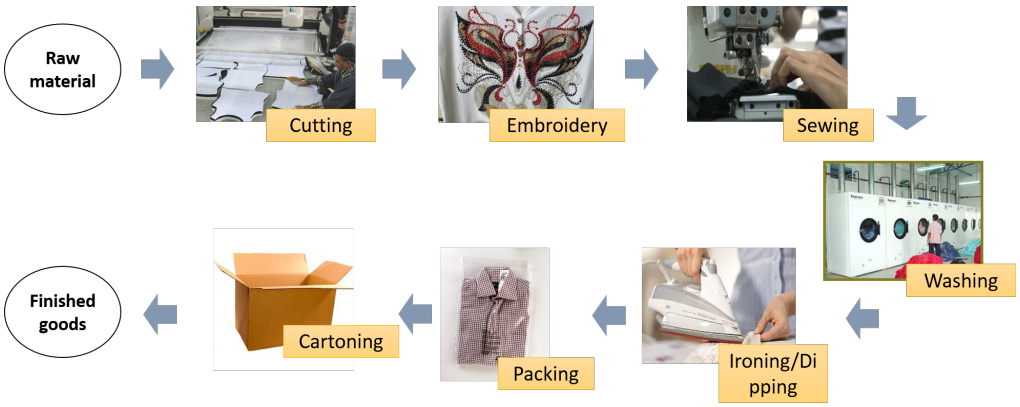


Figure 1.4: Apparel manufacturing application: production procedure

As an apparel company, it produces different types of products such as T-shirts, pants and costumes. In fact, each client demand corresponds to one particular product. This is the third difficulty of our problem that 400 to 1000 products have to be produced in the realistic instances. Different products have different unitary processing time therefore can not be aggregated directly.

Products can be grouped into about 400 different styles in total, which can be further grouped into about 50 style families (currently they are regrouping them to 150+ style families). On a sewing line, changing the style family from one to another requires a setup and an efficiency loss, which is known as learning curve. In this application, the learning curve is given as a list of worker efficiency over 10 working days instead of a function (see example in Table 1.6). Given a product, the processing time of sewing is given by the time needed to process one item divided by the efficiency of the sewing line (which is 1 maximum). To achieve the best efficiency, the same style family is usually

kept for a few days (1 to 2 style families per week) on a sewing line. For small orders, style family can be changed daily. Within the same style family, a style change will also cause a setup cost and setup time, which is minor comparing to style family change. This leads to the fourth difficulty of our problem that setup is sequence dependent, i.e., it depends not only on the current product but also the previous product. LSP with sequence dependent setup is known to be very hard to solve since there is Traveling Salesman Problem embedded inside.

Table 1.6: Apparel manufacturing production: learning curve example

Day	Day1	Day2	Day3	Day4	Day5	Day6	Day7	Day8	Day9	Day10
Efficiency	0.6	0.65	0.7	0.8	0.9	1.0	1.0	1.0	1.0	1.0

Each sales order, i.e., demand, specifies the product required, the quantity, the shipping destination and the due date. It is allowed to delay the delivery or even cancel the delivery of a sale order, however, corresponding penalty will be charged accordingly. There are two levels of tardiness cost. The first level corresponds to a shipment cost by airplane instead of maritime to catch up the due date. The second level corresponds to a compensation for the orders delivered too late. This leads to a different model from classical LSP that we have introduced in Section 1.2 that demands can be delayed or lost with penalty cost, and the cost is defined during a interval. This feature is often referred as *lost sale* and *backlogging* in the literature.

Some aspects are out of the scope of this project. First, the ordering and management of raw material. The existing MRP system will generate the earliest available date for all the raw materials in an order. This date will be used as the earliest possible start time for an order. Second, demand forecasting. All work orders confirmed or forecasted are treated the same as an input to our system. However, as one way to reduce uncertainty, forcasted demands also have a start date to prevent them from producing too early. Therefore, we have a release date for each demand that only after which the production for this demand can start.

To make use of the planning resources efficiently, the company would like to plan its production activities for a future period of time, which is called planning horizon. The duration of the planning horizon considered is between 24 weeks to 56 weeks. The major objective of the production planning includes: maximization of the on-time delivery rate, minimization of the late shipment cost (including expediting transportation cost and sales loss cost), minimization of the learning curve loss and setup costs on the sewing lines. The minimization of the labor cost (due to overtime) is also one goal which is not

considered in the project currently.

1.3.2 Production Planning Problem Modeling

To be able to optimize the production planning in this apparel project, we first have to extract the mathematical model out of the context. Modeling large scale real-world application is always a trade-off between accuracy and solvability, and often has a big impact on the solution quality. In Table 1.7, we present the mapping between the application aspects presented in previous section to a lot sizing problem we will study in following chapters.

Table 1.7: Modeling apparel production planning to lot sizing problem

Application	Lot sizing Problem
Planning horizon 20 - 30 weeks	Weekly time buckets
7 production process step	Bottleneck sewing process, single level LSP
Multiple sewing production lines	Non-identical parallel machines
300 - 1000 products, 1 - 80 style families	Aggregated product for each style family
Sequence dependent learning curve	Setup carryover
Demand release date	Production time window
Demand can be lost or delayed	Lost sales and backlogging

Lot sizing problem with up to seven levels is very complex. Therefore, we only model the bottleneck process and study the single level LSP. Modeling setup and learning curve accurately on the sewing production line will dramatically increase the problem complexity. In order to achieve reasonable performance, we simplify the problem in the production planning level and deal with them in the scheduling phase. First, all setup between styles within the same style family is ignored. Second, each learning curve is transformed to a total capacity loss. In our example in Table 1.6, the efficiency loss equals to $(1 - 0.6) + (1 - 0.65) + (1 - 0.7) + (1 - 0.8) + (1 - 0.9) = 1.35$. Then the capacity loss will be $1.35 \times dailyCapacity$ and the new setup time equals to capacity loss, which depends on both products and production lines. Third, we relax the sequence dependent setup to setup carryover so that the model still prefers to keep the same style family on a production line, but with less complexity.

In summary, the lot sizing problem based on this project consists of single level production, multiple products, parallel machines, backlogging, lost sale, production time window and setup carryover. To the best of our knowledge, this problem is first studied in this manuscript. We will formally define the problem and perform study on it in the following chapters. In the next section, we summarize our major contributions.

1.4 Contributions

In this section, we provide a contribution summary of our research, which can be used as a reading guide for the rest of this manuscript.

First of all, a complex capacitated lot sizing problem with setup carryover is formulated and studied, which is based on the real-world application introduced in Section 1.3. The problem is shown to be NP-hard and different MIP formulations are proposed in Chapter 2. Two sets of benchmark instances are presented to evaluate these formulations. One set consists of real-world application instances, whereas the other set is pseudo-randomly generated and simulates characteristics observed from real-world instances. The computational results show that the problem can not be solved within reasonable time limit by the standard MIP solver CPLEX. Therefore, heuristic algorithms are developed to tackle this problem in Chapter 3. Both constructive and improving heuristic algorithms are developed. We perform experimental tests to evaluate performances of all developed heuristic algorithms and show the efficiency of our algorithms compared to the standard MIP solver CPLEX. In Chapter 4, our study results are implemented in the production planning engine for the apparel company and we show the complete industrial production planning solution.

Second, a special case of capacitated lot sizing problem with sequence dependent setup is studied in Chapter 5, which is called CLSP with a fixed product sequence. In many manufacturing industry, switching production from one product to another will cause setup operations. The setup will consume limited machine capacity and/or cause a setup cost. When the setup depends on the production sequence, i.e., the setup to produce current product depends on both itself and the previous produced product, it is called sequence dependent setup [46, 63]. In this case, both lot sizing and sequencing decisions have to be made. The difficulty of this problem lies in the factorial number of setup sequence candidates to be chosen from for each time bucket. However, in certain manufacturing industries, this number may be reduced if we restrict the model based on the planners' knowledge. We consider a restricted model in which the number of potential setup sequences is reduced to $O(n2^n)$ compared to $O(n!)$ for the CLSP with sequence dependent setups. The problem is shown to be NP-hard, and a column generation heuristic is developed. A set of benchmark instances is tested and computational results are presented to evaluate the algorithm performance.

Chapter 2

Complex Capacitated Lot Sizing Problem: Formulations and Benchmarks

A complex capacitated lot sizing problem is constructed from the apparel manufacturing application presented in the previous chapter. This capacitated lot sizing problem consists of complex features including parallel machines, production time windows, backlogging, lost sales and setup carryover [48]. These features have been studied in different context of lot sizing problems. However, to the best of our knowledge, they are first considered together in this application. In this chapter, we formally define, formulate and analyze the problem.

The chapter is organized as follows: In Section 2.1, we formally define the complex capacitated lot sizing problem. In Section 2.2, we present the literature review of lot sizing problem closely related to our problem classified by features. In Section 2.3, four mixed integer programming formulations are proposed and compared theoretically. In Section 2.4, benchmark instances are presented, including both application instances and pseudo-randomly generated instances with realistic characteristics. In Section 2.5, computational results are presented to evaluate developed formulations. Finally, we conclude in Section 2.6.

2.1 Problem Definition

Input parameters of the problem are:

- $\mathcal{T} = \{1, 2, \ldots, T\}$: set of time buckets.
- $\mathcal{R} = \{1, 2, \ldots, R\}$: set of resources/machines.
- $\mathcal{N} = \{1, 2, \ldots, N\}$: set of products.
- $\mathcal{D} = \{1, 2, \ldots, D\}$: set of demands.
- cap_{rt}: capacity of machine r in time bucket t ($r \in \mathcal{R}$, $t \in \mathcal{T}$).
- pt_i: unitary processing time of product i ($i \in \mathcal{N}$).
- st_{ir}: setup capacity for product i on machine r ($i \in \mathcal{N}$,$r \in \mathcal{R}$).
- sc_{ir}: setup cost for product i on machine r ($i \in \mathcal{N}$,$r \in \mathcal{R}$).
- $p_d \in \mathcal{N}$: required product of demand d ($d \in \mathcal{D}$).
- q_d: quantity of product p_d required by demand d ($d \in \mathcal{D}$).
- $b_d \in \mathcal{T}$: release date of demand d ($d \in \mathcal{D}$).
- $e_d^1 \in \mathcal{T}$: first due date of demand d ($d \in \mathcal{D}$). No extra cost in interval $[b_d, e_d^1)$.
- $e_d^2 \in \mathcal{T}$: second due date of demand d ($d \in \mathcal{D}$).
- tc_d^1: unitary tardiness cost of demand d satisfied at or after e_d^1 ($d \in \mathcal{D}$).
- tc_d^2: unitary tardiness cost of demand d satisfied at or after e_d^2 ($d \in \mathcal{D}$).
- lc_d: unitary lost sale cost of demand d ($d \in \mathcal{D}, lc_d > tc_d^1 + tc_d^2$).
- $\mathcal{D}^i \subseteq \mathcal{D}$: the subset of demands such that $p_d = i$, i.e., $\mathcal{D}^i := \{d \in \mathcal{D} | p_d = i\}$.

The problem is to decide for each machine $r \in \mathcal{R}$ and in each time bucket $t \in \mathcal{T}$, how much to produce of each product $i \in \mathcal{N}$. The objective is to minimize the total cost including lost sale cost, tardiness cost and setup cost. The restriction includes three parts: first, the machine capacities cap_{rt} must not be exceeded by the capacity usage for each machine $r \in \mathcal{R}$ and time bucket $t \in \mathcal{T}$; second, the production to satisfy demand d can only start from its release date; last, setup carryover is considered. This means that to produce product i on machine r during time bucket t, there has to be a setup for i

on r during t. However, if product i is the last product produced in the previous time bucket $t-1$ on machine r, there is no setup needed to produce product i on machine r during time bucket t anymore.

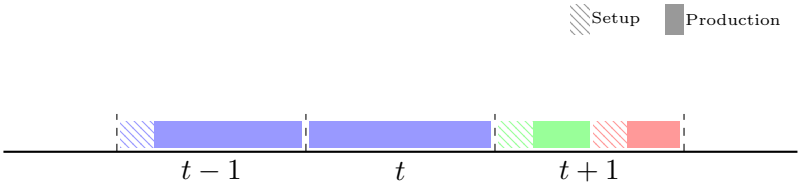


Figure 2.1: Setup carryover

We assumes that there is no more than one setup for each product on each machine during each time bucket. A pseudo formulation may serve to summarize the problem as follows. To the best of our knowledge, it is the first time that this CLSP is studied, we denote it as *CLSC* for simplicity.

$$
\begin{aligned}
(CLSC) \quad \min \quad & LostSaleCost + TardinessCost + SetupCost \\
s.t. \quad & \textit{Material flow conservation constraints} \\
& \textit{Machine capacity constraints} \\
& \textit{Time windows of demands} \\
& \textit{Setup carryover}
\end{aligned}
$$

Based on the definition, we notice that CLSC is different from classical CLSP at the demand definition. In CLSP, demands are normally aggregated by products and time buckets. Hence, a demand is defined for each product in each time bucket. However, in our case it is important to consider individual time window of each demand based on its release date and due dates. Therefore, we separate the concept of product and demand. Each demand d requires one product p_d with quantity q_d, and is given with a release date r_d, two due dates e_d^1, e_d^2, their associated tardiness cost tc_d^1, tc_d^2 and lost sale cost lc_d. In other words, one product is required by a set of demands but each demand is associated to one product.

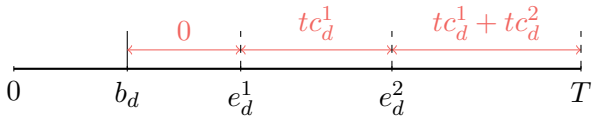


Figure 2.2: Time window of demand

Another difference is that there is no inventory to be taken into consideration. Not only there is no inventory cost, but also the produced product is used to satisfy demands directly, i.e., immediate delivery. These two differences imply that CLSC has certain scheduling features since the cost and material flow are directly connected to demand delivery.

We illustrate the problem in Example 2.1.

Example 2.1. *We consider 2 machines, 3 products and 5 time buckets. Parameters are given in the Table 2.1, Table 2.2 and Figure 2.3. Production time pt_i equals to 1 for all 3 products.*

Table 2.1: CLSC Example 2.1 data: setups

st_{ir}	r_1	r_2
i_1	1	1
i_2	1	1
i_3	1	1

sc_{ir}	r_1	r_2
i_1	5	5
i_2	3	3
i_3	3	3

Table 2.2: CLSC Example 2.1 data: capacities

cap_{rt}	t_1	t_2	t_3	t_4	t_5
r_1	2	1	2	1	2
r_2	1	2	1	2	1

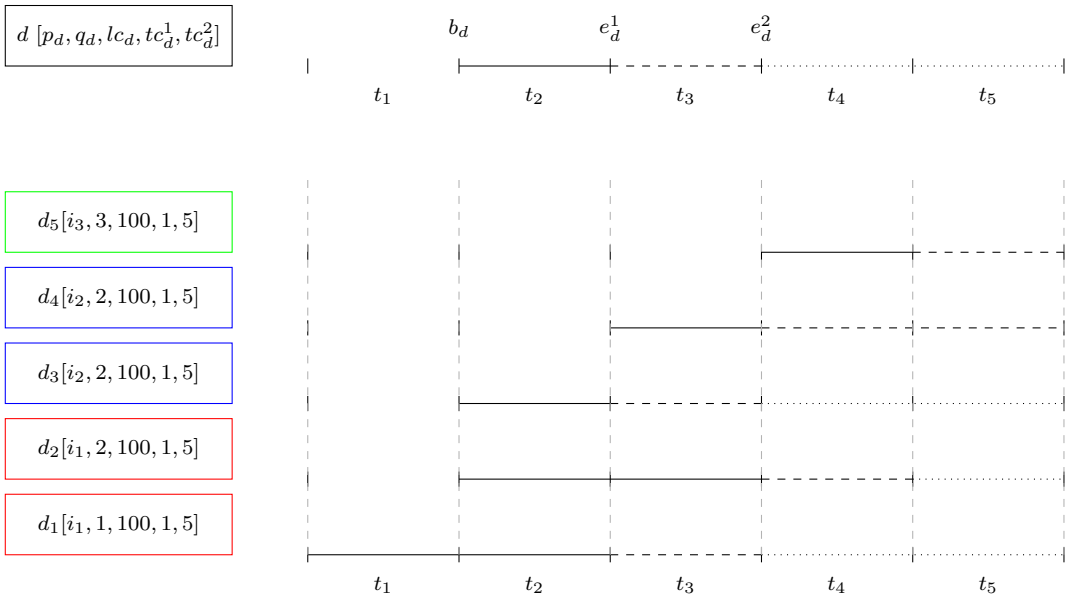


Figure 2.3: CLSC Example 2.1 data: time windows

Demand parameters is given in Figure 2.3. For example, demand d_3 requires 2 units of product i_2. We can start to produce for d_3 from its release date t_2. If we deliver before its first due date t_3, i.e., within $[t_2, t_2]$, it is on time. If we deliver at or after the first due date but before the second due date t_4, i.e., within $[t_3, t_3]$, it is delayed and a unitary tardiness cost of 1 is charged per unit of delivery quantity. If we deliver at or after the second due date t_4, i.e., within $[t_4, t_5]$, it is delayed and a unitary tardiness cost of 6=1+5 is charged per unit of delivery quantity. If we do not fulfill d_3, a lost sale cost 100 per unit is paid.

The optimal solution is described in Figure 2.4 with total cost 6. The lost sale cost is 0 since all demands are satisfied. There is only one setup on machine r_2 in time bucket t_3 for product i_3, hence the setup cost is 3. Demand d_3, d_4 and d_5 are delayed, so the tardiness cost is 3. For instance, demand d_3 is delivered in two lots: time bucket t_2 and time bucket t_3. The first delivery is on time whereas the second delivery is late with a tardiness cost of $1 = 1 \times 1$. Therefore, the tardiness cost of demand d_3 is 1.

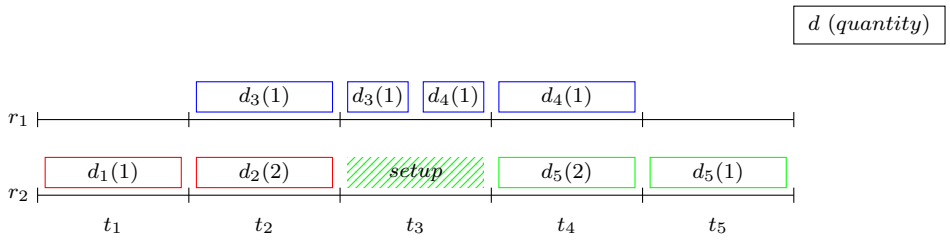


Figure 2.4: CLSC Example 2.1 optimal solution with setup cost

Theorem 2.1. *CLSC is strongly NP-hard.*

Proof. The result follows from the fact that CLSP is strongly NP-hard [28], which can be seen as a special case of CLSC. □

Theorem 2.2. *CLSC without setup cost is NP-hard.*

Proof. Trigeiro et al. [129] proved that CLSP is NP-hard even without setup cost, therefore as an extension of CLSP, CLSC without setup cost is NP-hard. □

2.2 Literature Review

In this section, we review the state-of-the-art literature of LSP that are related to CLSC. Specially, we present the overview based on features, including setup carryover, parallel machines, production time windows, backlogging and lost sale.

Setup carryover. Setup carryover is also called linked lot size or linked production quantities (Haase [69]). In each time bucket, producing a positive amount of products causes a setup time and/or a setup cost. However, if the first product produced in t is the same as the last product produced in the previous time bucket $t-1$, then in time bucket t we can continue to produce the same product without additional setup. This is called setup carryover. The setup carryover is always considered in CLSP with multiple products.

The LSP with setup carryover are first studied in Dillenberger et al. [36, 37]. Since then, most of the research on setup carryover has been focused on the formulation and heuristic algorithm design. In Dillenberger et al. [36, 37], a MIP model has been proposed and a fix-and-relax heuristic algorithm has been developed. In Gopalakrishnan et al. [58], a MIP model has been proposed and a real-world instances with multiple machines and product families is solved by using the solver LINDO. In Haase [69], the setup carryover is restricted to at most one time bucket, a MIP model has been proposed and a priority rule based heuristic algorithm is developed. In Sox and Gao [122], two MIP models are proposed while one is based on the shortest path formulation. Also, a decomposition heuristic algorithm is developed which is based on Lagrangian relaxation. In Gopalakrishnan [56], they extend the formulation in Gopalakrishnan et al. [58] so that it incorporates product dependent setup times and costs. Later in Gopalakrishnan et al. [57] a Tabu Search (TS) algorithm is proposed for this model. In Suerie and Stadtler [124], another formulation is proposed and it is furthermore extended by introducing extra variables and valid inequalities. A MIP solver together with a procedure to add cuts is used to solve the problem. In Briskorn [24], the Lagrangian relaxation based heuristic algorithm proposed in Sox and Gao [122] is modified so that subproblems are guaranteed to be solved optimally. In Karimi et al. [89], a CLSP model is studied which considers multi-item, setup carryover and backlogging. A TS heuristic algorithm is developed for it. In Nascimento and Toledo [102], the problem is extended to multiple plants, therefore the possibility of transporting products between plants is considered. A MIP formulation is proposed and a Greedy Randomized Adaptive Search Procedure (GRASP) heuristic algorithm is designed. In Sahling et al. [116], a multi level CLSP with setup carryover is studied, a MIP formulation is proposed and a MIP based fix-and-optimize heuristic algorithm is developed. In Goren et al. [60], a hybrid approach combining genetic algorithms and a fix-and-optimize heuristic is proposed. It is compared to the TS algorithm developed in Gopalakrishnan et al. [57] and is shown to have a better solution quality with longer computational time. In Gören and Tunal [59], another hybrid approach combining Genetic Algorithms (GAs) and a fix-and-optimize heuristic

is proposed, which is different from that of Goren et al. [60]. In Goren et al. [60], the fix-and-optimize heuristic is embedded in the GA procedure so solve each subproblem, while in Gören and Tunal [59] it runs GAs for a predetermined number of generations and use the overall best solution as the initial solution for the fix-and-optimize heuristic.

Parallel machines. Parallel machines are commonly taken into account in practical production planning such as pharmaceutical industry, disposable products and so on. The introduction of parallel machines may lead to a large amount of symmetric solutions, therefore it increases the difficulties of the problem.

In Özdamar and Birbil [106], a lot sizing and loading problem studied deals with the issue of determining the lot sizes of product families/end items and loading them on parallel facilities to satisfy dynamic demand over a given planning horizon. The facilities here have similar functions as parallel machines. A hybrid algorithm combining TS, GA and Simulated Annealing (SA) is developed. It is further extended to multi-stage model in Ozdamar and Barbarosoglu [105], where a hybrid algorithm based on Lagrangian relaxation, SA and GA is also proposed. In Kang et al. [87], a LSP on parallel machines with sequence dependent setup costs is studied. The problem is solved by a branch and bound algorithm based on column generation. In Meyr [98], a CLSP model with micro time buckets, parallel machines and minimum lot size is studied. A heuristic algorithm combining threshold acceptance and SA with dual re-optimization is also developed. In Quadt and Kuhn [110], a CLSP model with setup times, setup carryover, back-orders, and parallel machines is studied. To find a solution of the original model, the aggregate model is embedded in a lot sizing and scheduling procedure. In Tempelmeier and Copil [126], a CLSP model with parallel machines, sequence dependent setup, shelf life and a single common setup resource is studied. Two MIP based heuristic algorithms including a fix-and-optimize heuristic and a fix-and-relax heuristic are proposed and tested. Some heuristic algorithms are based on Lagrangian relaxation on capacity constraints such as in Toledo and Armentano [128] or demand constraints such as Fiorotto and de Araujo [44] to be able to decompose the problem into sub problems. In Fiorotto et al. [45], a DantzigWolfe decomposition is applied to the demand constraints where the master problem is solved by a combination of Lagrangian relaxation and DantzigWolfe decomposition in a hybrid form. The parallel machines are also considered in Nattaf et al. [103], Almada-Lobo et al. [10] and the bc-prod system see Belvaux and Wolsey [14].

Most cases considering parallel machines are in the context of scheduling, for a though survey we refer to Charrua et al. [26]. There are other papers considering multiple resources without considering setup on machines but only resource capacity or usage

cost and so on. In Diaby et al. [35], the setup is counted for each time bucket which means when a setup is paid once in a time bucket, all machines are able to produce the corresponding product. In Hindi [73], no setup is considered but there is a unit machine usage cost and capacity per machine per time bucket.

Production time windows. In the LSP model with production time windows, each demand has a release date and a due date, during which the production for this demand must be fulfilled. Therefore, the release date and the due date of a demand become its time window. Moreover, there are two cases: customer-specific or non-customer-specific time windows. In the customer-specific case, each demand has a specific release date and the release quantity can not be used to satisfy other demands. In the non-customer-specific case, products produced in s can be used to satisfy any demand that require this product. The release date is used to model raw material availability and customer confirmation date. A customer order can still be canceled before its confirmation date and we would like to avoid producing before it is confirmed.

The LSP model with production time windows is first studied in Brahimi [19], Dauzère-Pérès et al. [32], Brahimi et al. [21]. In Dauzère-Pérès et al. [32], the uncapacitated case is studied and a general dynamic programming pseudo-polynomial algorithm is presented for the customer-specific problem and a polynomial time $O(T^4)$ algorithm is proposed for the non-customer-specific case. In Brahimi et al. [21], the capacitated case is studied which also extends the problem to multi-item. Lagrangian relaxations based heuristics are developed for both cases. In Wolsey [133], for the customer-specific case, tight extended formulations are proposed for both the constant capacity and uncapacitated problems with Wagner-Whitin (non-speculative) costs. For the non-customer-specific case, it is shown to be equivalent to the basic lot-sizing problem with upper bounds on the stocks. Also, polynomial time dynamic programming algorithms and tight extended formulations for the uncapacitated and constant capacity models with general cost are also developed. In Hwang [80], different cost structures are studied and a dynamic programming algorithm with $O(T^5)$ is proposed. In van den Heuvel and Wagelmans [130], four LSP variants are shown to be equivalent which includes the LSP with a remanufacturing option [112], the LSP with production time windows, the LSP with cumulative capacities [120] and the LSP with bounded inventory [94]. In Brahimi et al. [23], the CLSP with multi-item, non-customer-specific production time windows and setup times is studied. A Lagrangian relaxation based heuristic algorithms is developed and a reformulation is proposed. In Absi et al. [4], the production time window is studied together with lost sale as well as early production, backlog on the uncapacitated

LSP. Several properties of the optimal solution are presented for different variants of the problem when production time windows are non-customer specific. Exact dynamic programming algorithms are developed with computational complexity $O(T^2)$.

Backlogging. Backlogging is also called inventory shortage, or backorder in Millar and Yang [99]. If it is possible to satisfy a demand after its due date, it is called backlogging. Together with lost sale, they are common features in practice when there is insufficient capacity or for simulation analysis purpose.

This feature backlogging has been widely studied in the literature. Here we review the literature which is most related to our problem. In Zangwill [134], backlogging is first studied in an LSP model with concave production costs and piecewise concave inventory costs. A dynamic programming algorithm is also proposed. In Pochet and Wolsey [107], mixed integer programming reformulations of the uncapacitated lot-sizing problem with constant cost and backlogging is studied. The linear programming reformulations solves the problem directly, while a cut generation algorithm is also developed with a family of cuts. In Choo and Chan [29], a simple class of heuristic algorithms two-way eyeballing heuristic (TWEH) is presented which first determine the backlogging periods and then the production quantities. This algorithm is further compared in Hsieh et al. [74] with modified algorithms which are originally designed for LSP, the result shows that TWEH is the simplest algorithm with good performance. In Federgruen and Tzur [43], time-variant cost starts to be considered in the model and a $O(T \log T)$ exact algorithm is developed. In Chen et al. [27], a LSP model with piecewise linear costs and capacity restrictions on both production and inventory is studied, also a dynamic programming algorithms is developed. In Millar and Yang [100], the multi-item CLSP with backordering is studied and two heuristic algorithms based on a network-based formulation and Lagrangian decomposition are developed. In Robinson Jr. and Gao [115], backlogging is considered together with coordinated replenishment. A mixed-integer programming formulation is proposed and dual ascent based branch-and-bound algorithm is developed. In Ozdamar and Barbarosoglu [105], the multi-stage CLSP with backlogging on the last stage is studied and a hybrid algorithm is developed which embeds SA and GA into Lagrangean relaxation. In Hung et al. [79], a CLSP model with parallel machines, setup and backlogging is studied and different GA algorithms are used to make setup decisions. In Hung and Chien [78], a multi-level CLSP considers multiple demand classes with backlogging is studied, where the MIP models corresponding to each demand class is solved in sequence. Three heuristic algorithms including TS, GA and SA are developed and compared. In Belvaux and Wolsey [15], different formulation techniques

for a range of LSP is discussed which includes backlogging, start-ups, changeovers and so on. Papers that consider backlogging also include Gupta and Brennan [65], Hung et al. [77], Jans and Degraeve [83], Duda [41], Karimi et al. [89], Megala and Jawahar [97], Gaafar [50], Kämpf and Köchel [86], Huai-En Chiao et al. [76].

There are many papers considering backlogging which considers other topics such as ELSP in Zangwill [135], Blackburn and Kunreuther [18], Hsu and Lowe [75], or based on inventory system in one period in Atkins and Sun [12], Sun and Atkins [125], or integrate pricing and LSP on infinite planning horizon in Abad [1].

Lost sale. Lost sale is also called stockout in Sandbothe and Thompson [118], where it is possible to not meet demands with a penalty cost.

Comparing to backlogging, there are much fewer papers considering LSP with lost sales. The CLSP with lost sale is first studied in Sandbothe and Thompson [118] with constant cost over time period, in which two necessary optimality conditions are stated and two forward algorithms are developed for the constant capacity case and non-constant capacity case. In Sandbothe and Thompson [119], the problem is extended to include also capacity constraints on inventory, optimality conditions are also stated together with a forward algorithm of asymptotically linear time. In Aksen et al. [6], an uncapacitated single-item LSP with lost sales is studied which have a time-variant cost structure. Several structural properties of optimal solutions are proposed and an exact algorithm in linear time $O(T^2)$ is developed. In Absi et al. [4], the lost sale is studied together with production time windows as well as early production, backlog on the uncapacitated LSP. Several properties of the optimal solution are presented for different variants of the problem when production time windows are non customer specific. Exact dynamic programming algorithms are developed with computational complexity $O(T^2)$.

There a few other papers considering lost sale as well such as Abad [1], Teng et al. [127], Huai-En Chiao et al. [76], Abad [2], Ghosh et al. [54]. However, they focus on an integration of pricing and lot sizing with infinite time horizon with perishability or deteriorating inventories.

Among all these features related to CLSC, setup carryover and parallel machines contribute the most to the problem complexity. Without setup, the problem can be solved as a continuous optimization problem. Parallel machines not only increase the problem size but also make the LP relaxation solution provide less guidance to the MIP solution due to the fact that the production in the LP solution will be distributed to all machines. Therefore it is interesting for to study this problem and hopefully develop efficient algorithms for it.

2.3 MIP Formulations

In this section, we present MIP formulations for CLSC. The first three formulations are aggregated formulations with different ways to model setup carryover. The last formulation is adapted from facility location reformulation.

Aggregated Formulation 1 ($Form1$)

For each product $i \in \mathcal{N}$, each machine $r \in \mathcal{R}$, each time bucket $t \in \mathcal{T}$ and each demand $d \in \mathcal{D}$, we first introduce the following decision variables:

- $x_{irt} \in \mathbb{R}^+$: the production quantity of product i on machine r during time t.
- $y_{dt} \in [0, q_d]$: the satisfied quantity of demand d in time bucket $t \geq b_d$.
- $y_d \in [0, q_d]$: the unsatisfied quantity of demand d.

In Haase [67], a MIP formulation for CLSP on a single machine with setup carryover has been introduced. We adapt this formulation to our problem and introduce setup variables for each product $i \in \mathcal{N}$, each machine $r \in \mathcal{R}$ and each time bucket $t \in \mathcal{T}$ as follows:

- $v_{rt} \in [0, 1]$, $v_{rt} > 0$ indicates if more than one product is produced in time bucket t on machine r.
- $z_{irt} \in \{0, 1\}$ equals to 1 if a setup state for product i on machine r exists in time bucket t and 0 otherwise.
- $z^c_{irt} \in \{0, 1\}$ equals to 1 if the setup state for product i is carried over from time bucket $t-1$ to time bucket t on machine r and 0 otherwise.

Then the first formulation ($Form1$) is formally given as follows ($\tilde{\mathcal{T}} = \mathcal{T} \setminus \{1\}$):

$$\min \quad \sum_{d \in \mathcal{D}} lc_d y_d + \sum_{d \in \mathcal{D}, t \in \mathcal{T}: t \geq e^1_d} tc^1_d y_{dt} + \sum_{d \in \mathcal{D}, t \in \mathcal{T}: t \geq e^2_d} tc^2_d y_{dt} + \sum_{i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T}} sc_{ir}(z_{irt} - z^c_{irt}) \tag{2.1}$$

$$s.t. \quad \sum_{r \in \mathcal{R}} x_{irt} = \sum_{d \in \mathcal{D}: p_d = i, t \geq b_d} y_{dt} \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{2.2}$$

$$\sum_{b_d \leq t \in \mathcal{T}} y_{dt} + y_d = q_d \qquad d \in \mathcal{D} \tag{2.3}$$

$$\sum_{i \in \mathcal{N}} pt_i x_{irt} + \sum_{i \in \mathcal{N}} st_{ir}(z_{irt} - z^c_{irt}) \leq cap_{rt} \qquad r \in \mathcal{R}, t \in \mathcal{T} \tag{2.4}$$

$$x_{irt} \leq \Theta_{irt} z_{irt} \qquad i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T} \qquad (2.5)$$

$$\sum_{i \in \mathcal{N}} z^c_{irt} \leq 1 \qquad r \in \mathcal{R}, t \in \mathcal{T} \qquad (2.6)$$

$$z^c_{irt} \leq z_{ir,t-1} \qquad i \in \mathcal{N}, r \in \mathcal{R}, t \in \tilde{\mathcal{T}} \qquad (2.7)$$

$$z^c_{irt} \leq z_{irt} \qquad i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T} \qquad (2.8)$$

$$z^c_{irt} + z^c_{ir,t-1} + v_{r,t-1} \leq 2 \qquad i \in \mathcal{N}, r \in \mathcal{R}, t \in \tilde{\mathcal{T}} \qquad (2.9)$$

$$N v_{rt} \geq \sum_{i \in \mathcal{N}} z_{irt} - 1 \qquad r \in \mathcal{R}, t \in \mathcal{T} \qquad (2.10)$$

$$0 \leq x_{irt} \qquad i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T} \qquad (2.11)$$

$$0 \leq y_{dt}, y_d \leq q_d \qquad d \in \mathcal{D}, t \geq b_d \qquad (2.12)$$

$$z^c_{irt}, z_{irt} \in \{0, 1\} \qquad i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T} \qquad (2.13)$$

$$v_{rt} \in [0, 1] \qquad r \in \mathcal{R}, t \in \mathcal{T} \qquad (2.14)$$

where $\Theta_{irt} = \min\{\frac{cap_{rt}}{pt_i}, \sum_{d \in \mathcal{D}^i, b_d \leq t} q_d\}$. In the objective (2.1), it minimizes the cost including lost sale cost $\sum_{d \in \mathcal{D}} lc_d y_d$, first level tardiness cost $\sum_{d \in \mathcal{D}, t \in \mathcal{T}: t \geq e^1_d} tc^1_d y_{dt}$, second level tardiness cost $\sum_{d \in \mathcal{D}, t \in \mathcal{T}: t \geq e^2_d} tc^2_d y_{dt}$, and setup cost $\sum_{i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T}} sc_{ir}(z_{irt} - z^c_{irt})$. Note that according to setup carryover, a setup cost has to be paid when there is a setup ($z_{irt} = 1$) which is not carried over from the last time bucket ($z^c_{irt} = 0$). Constraints (2.2) state flow balance for each product in each time bucket, the inflow (production) equals to the outflow (demand satisfaction). Constraints (2.3) ensure that for each demand, the unsatisfied demand quantity plus the unsatisfied demand quantity equals to the demand quantity. Constraints (2.4) ensure that the capacity is not exceeded on each machine in each time bucket, where the setup capacity consumption is formulated similarly to the setup cost. Constraints (2.5) link the production and the setup since a positive production of i on r at t requires a setup state for i on r at t. Constraints (2.6) - (2.10) model setup carryover. There is at most one setup state to be carried over to the next time bucket, which is guaranteed by constraints (2.6). A setup state of i on r carried over from $t-1$ to t implies that this state is included in both $t-1$ (2.7) and t (2.8). If there is more than one setup state in one time bucket, i.e., $v_{rt} > 0$, the initial setup state and the last setup state are necessarily different. This is formulated as constraints (2.9). Finally, to fulfill the definition of variable v_{rt}, we have the constraints (2.10).

Aggregated Formulation 2 ($Form2$)

In Sox and Gao [122], two MIP formulations for CLSP on single machine with setup carryover are presented. One is aggregated formulation while the other one is network

formulation. Here we adapt the aggregated formulation to CLSC. Besides variables x_{irt}, y_{dt} and y_d as introduced above, we introduce the following setup variables for each product $i \in \mathcal{N}$, each machine $r \in \mathcal{R}$ and each time bucket $t \in \mathcal{T}$:

- $z^0_{irt} \in \{0,1\}$ equals to 1 if the initial setup state is for product i on machine r in time bucket t, implying that the final setup state for $t-1$ on r is for product i.
- $z^+_{irt} \in \{0,1\}$ equals to 1 if there is a state switch for product i on machine r in time bucket t.

Then the second formulation ($Form2$) is formally given as follows:

$$\min \quad \sum_{d\in\mathcal{D}} lc_d y_d + \sum_{d\in\mathcal{D},t\in\mathcal{T}:t\geq e^1_d} tc^1_d y_{dt} + \sum_{d\in\mathcal{D},t\in\mathcal{T}:t\geq e^2_d} tc^2_d y_{dt} + \sum_{i\in\mathcal{N},r\in\mathcal{R},t\in\mathcal{T}} sc_{ir} z^+_{irt} \tag{2.15}$$

$$s.t. \quad \sum_{r\in\mathcal{R}} x_{irt} = \sum_{d\in\mathcal{D}^i,t\geq b_d} y_{dt} \qquad i\in\mathcal{N}, t\in\mathcal{T} \tag{2.16}$$

$$\sum_{b_d\leq t\in\mathcal{T}} y_{dt} + y_d = q_d \qquad d\in\mathcal{D} \tag{2.17}$$

$$\sum_{i\in\mathcal{N}} pt_i x_{irt} + \sum_{i\in\mathcal{N}} st_{ir} z^+_{irt} \leq cap_{rt} \qquad r\in\mathcal{R}, t\in\mathcal{T} \tag{2.18}$$

$$x_{irt} \leq \Theta_{irt}(z^0_{irt} + z^+_{irt}) \qquad i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T} \tag{2.19}$$

$$\sum_{i\in\mathcal{N}} z^0_{irt} = 1 \qquad r\in\mathcal{R}, t\in\mathcal{T} \tag{2.20}$$

$$z^0_{irt} \leq z^0_{ir,t-1} + z^+_{ir,t-1} \qquad i\in\mathcal{N}, r\in\mathcal{R}, t\in\tilde{\mathcal{T}} \tag{2.21}$$

$$z^+_{jr,t-1} \leq 2 - z^0_{ir,t-1} - z^0_{irt} \qquad i,j\neq i\in\mathcal{N}, r\in\mathcal{R}, t\in\tilde{\mathcal{T}} \tag{2.22}$$

$$0 \leq x_{irt} \qquad i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T} \tag{2.23}$$

$$0 \leq y_{dt}, y_d \leq q_d \qquad d\in\mathcal{D}, b_d\leq t\in\mathcal{T} \tag{2.24}$$

$$z^0_{irt}, z^+_{irt} \in \{0,1\} \qquad i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T} \tag{2.25}$$

The total setup cost is formulated as $\sum_{i\in\mathcal{N},r\in\mathcal{R},t\in\mathcal{T}} sc_{ir} z^+_{irt}$ since the setup cost has to be paid only when there is a setup switch ($z^+_{irt} = 1$). Constraints (2.18) ensure that the total used capacity does not exceed the available capacity, where the setup capacity consumption is formulated similarly to the setup cost. Constraints (2.19) link the setup and production since a positive production of i on r at t requires a setup state for i on r at t, which is either from an initial setup state ($z^0_{irt} = 1$) or a setup switch ($z^+_{irt} = 1$). There is a unique initial setup state for each time bucket on each machine, which is established by constraints (2.20). Also, the initial setup state must be one of the setup

states in the previous time bucket (2.21). However, constraints (2.22) ensure that, on machine r during time bucket t, no setup switch is possible when the initial setup state and the last setup state of t (i.e., the initial setup state of the next time bucket $t+1$) are both for the same product.

Aggregated Formulation 3 ($Form3$)

In Suerie and Stadtler [124], a MIP formulation for CLSP on multiple unrelated machines with setup carryover is presented. This formulation is similar to the $Form2$. In addition to the previously defined binary variables z^0_{irt} and z^+_{irt}, additional variables w_{rt} are introduced for each machine $r \in \mathcal{M}$ and time bucket $t \in \mathcal{T}$:

- $w_{rt} \in [0,1]$ equals to 0 if there is a setup switch on r in t, it is greater than 0 otherwise.

Then the third formulation ($Form3$) is formally given as follows:

$$
\begin{aligned}
\min \quad & (2.15) \\
s.t. \quad & (2.16)-(2.21),(2.23)-(2.25) \\
& z^0_{irt} + z^0_{ir,t-1} \leq 1 + w_{r,t-1} && i \in \mathcal{N}, r \in \mathcal{R}, t \in \tilde{\mathcal{T}} && (2.26) \\
& z^+_{irt} + w_{rt} \leq 1 && i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T} && (2.27) \\
& 0 \leq w_{rt} \leq 1 && \forall r \in \mathcal{R}, t \in \mathcal{T} && (2.28)
\end{aligned}
$$

Basically, we substitute constraints (2.22) with constraints (2.26) - (2.28). Constraints (2.26) implies that on machine r, the initial setup states at $t-1$ and t have to be different when more than one product is produced during $t-1$ ($w_{r,t-1}=0$). Constraints (2.27) ensures that $w_{rt}=0$ when there are more than one setup state during time bucket t.

Facility Location Based Reformulation ($Form3^{FL}$)

In section 1.2 we introduce two reformulations of CLSP. Here we adapt facility location reformation of CLSP to CLSC. For all $d \in \mathcal{D}$, $b_d \leq t \in \mathcal{T}$ and $r \in \mathcal{R}$, we introduce

- $Q_{drt} \in \mathbb{R}^+$: the production quantity of product p_d on machine r during time t to satisfy demand d.

Then it is straightforward to build the relationship between previously defined variables x_{irt}, y_{dt}, y_d and Q_{drt}:

$$x_{irt} = \sum_{d \in \mathcal{D}^i, t \geq b_d} Q_{drt}$$

$$y_{dt} = \sum_{r\in\mathcal{R}} Q_{drt}$$
$$y_d = q_d - \sum_{r\in\mathcal{R}, b_d\le t\in\mathcal{T}} Q_{drt}$$

Here we apply these newly defined production variables on the $Form3$. However, it is not hard to see that we can also reformulate $Form1$ and $Form2$. The formulation is formally given as follows:

$$\min \sum_{d\in\mathcal{D}, t\in\mathcal{T}: t\ge e_d^1} tc_d^1 \sum_{r\in\mathcal{R}} Q_{drt} + \sum_{d\in\mathcal{D}, t\in\mathcal{T}: t\ge e_d^2} tc_d^2 \sum_{r\in\mathcal{R}} Q_{drt} + \sum_{d\in\mathcal{D}} lc_d(q_d - \sum_{r\in\mathcal{R}, b_d\le t\in\mathcal{T}} Q_{drt}) + \sum_{i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T}} sc_{ir} z_{irt}^+ \tag{2.29}$$

$$s.t. \sum_{r\in\mathcal{R}, b_d\le t\in\mathcal{T}} Q_{drt} \le q_d \qquad d\in\mathcal{D} \tag{2.30}$$
$$\sum_{i\in\mathcal{N}, d\in\mathcal{D}^i, b_d\le t} pt_i Q_{drt} + \sum_{i\in\mathcal{N}} st_{ir} z_{irt}^+ \le cap_{rt} \qquad r\in\mathcal{R}, t\in\mathcal{T} \tag{2.31}$$
$$Q_{drt} \le \Theta'_{drt}(z_{irt}^0 + z_{irt}^+) \qquad d\in\mathcal{D}, r\in\mathcal{R}, b_d\le t\in\mathcal{T} \tag{2.32}$$
$$\sum_{d\in\mathcal{D}^i} Q_{drt} \le \Theta_{irt}(z_{irt}^0 + z_{irt}^+) \qquad i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T} \tag{2.33}$$
$$0 \le Q_{drt} \le q_d \qquad d\in\mathcal{D}, r\in\mathcal{R}, b_d\le t\in\mathcal{T} \tag{2.34}$$
$$(2.20) - (2.21), (2.25), (2.26) - (2.28)$$

where $\Theta'_{drt} = \min\{q_d, \Theta_{p_d,r,t}\}$.

A comparison of the formulation size is summarized in Table 2.3. They all have the same number of binary variables, while $Form2$ has less continuous variables than $Form1$ and $Form3$. The number of constraints increases in the order $Form3$, $Form1$ and $Form2$. We compare the formulation $Form3$ and $Form3^{FL}$, the latter has more continuous variables and more constraints.

Table 2.3: CLSC formulation size comparison

Form	# Variables	# Binaries	# Constraints
$Form1$	$3NRT + RT + DT + D$	$2NRT$	$O(NRT)$
$Form2$	$3NRT + DT + D$	$2NRT$	$O(N^2RT)$
$Form3$	$3NRT + RT + DT + D$	$2NRT$	$O(NRT)$
$Form3^{FL}$	$DRT + 2NRT + RT$	$2NRT$	$O(DRT)$

We compare different formulations theoretically and have following conclusions:

Theorem 2.3. *The optimal objective function values of the LP relaxations of formulations $Form1$, $Form2$ and $Form3$, represented by $Obj^*_{LP}(Form1)$, $Obj^*_{LP}(Form2)$ and $Obj^*_{LP}(Form3)$, have following relationship*

$$Obj^*_{LP}(Form2) = Obj^*_{LP}(Form3) \geq Obj^*_{LP}(Form1)$$

Proof. Let $(\tilde{x}, \tilde{y}, \tilde{z}^c, \tilde{z}, \tilde{v})$, $(\bar{x}, \bar{y}, \bar{z}^0, \bar{z}^+)$ and $(\dot{x}, \dot{y}, \dot{z}^0, \dot{z}^+, \dot{w})$ be optimal solutions of the LP relaxation of $Form1$, $Form2$ and $Form3$ respectively, while $f_1(\tilde{x}, \tilde{y}, \tilde{z}^c, \tilde{z}, \tilde{v})$, $f_2(\bar{x}, \bar{y}, \bar{z}^0, \bar{z}^+)$ and $f_3(\dot{x}, \dot{y}, \dot{z}^0, \dot{z}^+, \dot{w})$ are the corresponding optimal objective function values.

First, we prove that $f_2(\bar{x}, \bar{y}, \bar{z}^0, \bar{z}^+) \geq f_3(\dot{x}, \dot{y}, \dot{z}^0, \dot{z}^+, \dot{w})$. Define

$$\bar{w}_{rt} = \min_{i \in \mathcal{N}} \{1 - \bar{z}^+_{irt}\},$$

then we claim that solution $(\bar{x}, \bar{y}, \bar{z}^0, \bar{z}^+, \bar{w})$ is a feasible solution of $Form3$. The only constraints in $Form3$ that are not in $Form2$ is (2.26) and (2.27). Since (2.27) is satisfied by definition of $\bar{w}$, we only need to show that (2.26) holds.

$$\bar{z}^0_{ir,t+1} + \bar{z}^0_{irt} \leq \min_{j \in \mathcal{N}} \{2 - \bar{z}^+_{jrt}\} = 1 + \min_{j \in \mathcal{N}} \{1 - \bar{z}^+_{jrt}\} = 1 + \bar{w}_{rt}.$$

The first inequality is due to constraints (2.22), while the third equality is due to definition of $\bar{w}$. Moreover, $f_3(\bar{x}, \bar{y}, \bar{z}^0, \bar{z}^+, \bar{w})$ equals to $f_2(\bar{x}, \bar{y}, \bar{z}^0, \bar{z}^+)$ by definition of the objective. Therefore $f_2(\bar{x}, \bar{y}, \bar{z}^0, \bar{z}^+) = f_3(\bar{x}, \bar{y}, \bar{z}^0, \bar{z}^+, \bar{w}) \geq f_3(\dot{x}, \dot{y}, \dot{z}^0, \dot{z}^+, \dot{w})$.

Second, we prove that $f_2(\bar{x}, \bar{y}, \bar{z}^0, \bar{z}^+) \leq f_3(\dot{x}, \dot{y}, \dot{z}^0, \dot{z}^+, \dot{w})$. More specifically, we will show that solution $(\dot{x}, \dot{y}, \dot{z}^0, \dot{z}^+)$ is feasible for $Form2$, and has the same objective function value as $f_3(\dot{x}, \dot{y}, \dot{z}^0, \dot{z}^+, \dot{w})$. The only constraints in $Form2$ that do not appear in the $Form3$ is (2.22). For any $j \neq i \in \mathcal{N}$, we have

$$\dot{z}^0_{ir,t+1} + \dot{z}^0_{irt} \leq 1 + \dot{w}_{rt} \leq 1 + 1 - \dot{z}^+_{jrt} = 2 - \dot{z}^+_{jrt}.$$

The first inequality is due to (2.26) while the second inequality is due to (2.27). Therefore, $f_2(\bar{x}, \bar{y}, \bar{z}^0, \bar{z}^+) \leq f_2(\dot{x}, \dot{y}, \dot{z}^0, \dot{z}^+) = f_3(\dot{x}, \dot{y}, \dot{z}^0, \dot{z}^+, \dot{w})$.

Third, we show that $f_2(\bar{x}, \bar{y}, \bar{z}^0, \bar{z}^+) \geq f_1(\tilde{x}, \tilde{y}, \tilde{z}^c, \tilde{z}, \tilde{v})$. First of all, we point out that there exists an optimal solution of $Form1$ satisfying that

$$\bar{z}^0_{irt} + \bar{z}^+_{irt} \leq 1.$$

This is due to the fact that in constraints (2.19), Θ_{irt} is the upper bound of the production quantity x_{irt}. Define

$$\bar{z}_{irt} = \bar{z}^0_{irt} + \bar{z}^+_{irt}$$

$$\hat{z}^c_{irt} = \min\{\bar{z}^0_{irt}, \bar{z}_{ir,t-1}, \bar{z}_{irt}\}$$
$$\bar{v}_{rt} = \frac{\sum_i \bar{z}_{irt} - 1}{N}$$

we will show that the solution $(\bar{x}, \bar{y}, \bar{z}, \hat{z}^c, \bar{v})$ is a feasible solution for $Form1$ which has the same objective function value as $f_2(\bar{x}, \bar{y}, \bar{z}^0, \bar{z}^+)$. Constraints (2.5), (2.6), (2.7), (2.8) and (2.10) hold due to the definition of the variables. For constraints (2.9),

$$\begin{aligned}\bar{v}_{rt} &= \frac{\sum_i \bar{z}_{irt} - 1}{N} = \frac{\sum_i (\bar{z}^0_{irt} + \bar{z}^+_{irt}) - 1}{N} \\ &= \frac{\sum_i \bar{z}^0_{irt} + \sum_i \bar{z}^+_{irt} - 1}{N} = \frac{\sum_i \bar{z}^+_{irt}}{N} \leq \max_i \bar{z}^+_{irt}\end{aligned}$$

Hence,

$$z^0_{ir,t+1} + z^0_{irt} \leq 2 - \max_j z^+_{jrt} \leq 2 - \bar{v}_{rt}.$$

Therefore, $f_2(\bar{x}, \bar{y}, \bar{z}^0, \bar{z}^+) = f_1(\bar{x}, \bar{y}, \bar{z}, \hat{z}^c, \bar{v}) \geq f_1(\tilde{x}, \tilde{y}, \tilde{z}^0, \tilde{z}, \tilde{v})$. □

This theorem shows that the LP relaxation of $Form2$ and $Form3$ provide equivalently lower bounds no worse than $Form1$.

Theorem 2.4. *The formulation $Form3^{FL}$ is stronger than the formulation $Form3$ in the sense that its optimal objective function value of the LP relaxation $Obj^*_{LP}(Form3^{FL})$ is greater than or equal to that of the $Form3$ $Obj^*_{LP}(Form3)$.*

$$Obj^*_{LP}(Form3) \leq Obj^*_{LP}(Form3^{FL})$$

Proof. Let $(\mathring{Q}, \mathring{z}^0, \mathring{z}^+, \mathring{w})$ be the optimal solution of the LP relaxation of $Form3^{FL}$, we will show that there always exists a corresponding solution of $Form3$ which shares the same objective function value. For each $i \in \mathcal{N}$, $r \in \mathcal{R}$ and $t \in \mathcal{T}$, define

$$\ddot{x}_{irt} = \sum_{d \in \mathcal{D}: p_d = i, t \geq b_d} \mathring{Q}_{drt} \tag{2.35}$$
$$\ddot{y}_{dt} = \sum_{r \in \mathcal{R}} \mathring{Q}_{drt} \tag{2.36}$$
$$\ddot{y}_d = q_d - \sum_{b_d \leq t \in \mathcal{T}} \ddot{y}_{dt} \tag{2.37}$$

First, we will show that $(\ddot{x}, \ddot{y}, \mathring{z}^0, \mathring{z}^+, \mathring{w})$ is a feasible solution of $Form3$. Since

$$\sum_{r \in \mathcal{R}} \ddot{x}_{irt} = \sum_{r \in \mathcal{R}} \sum_{d \in \mathcal{D}: p_d = i} \mathring{x}_{irtd} = \sum_{d \in \mathcal{D}: p_d = i} \sum_{i = p_d, r \in \mathcal{R}} \mathring{x}_{irtd} = \sum_{d \in \mathcal{D}: p_d = i} \ddot{y}_{dt}$$

constraints (2.16) hold. Constraints (2.17) hold due to the definition of $\ddot{y}_d$, whereas constraints (2.18) hold due to the definition of $\ddot{x}_{irt}$. As for the production and setup linking constraints (2.5), it holds due to constraints (2.33).

Second, the solution $(\ddot{x}, \ddot{y}, \ddot{z}^0, \ddot{z}^+, \ddot{w})$ and $(\mathring{Q}, \mathring{z}^0, \mathring{z}^+, \mathring{w})$ share the same objective function value due to the fact that the objective function of $Form3^{FL}$ is based on the substitution of (2.35) - (2.37). Therefore, the theorem holds. □

2.4 Benchmark Instances

To perform experimental study on the problem CLSC, we introduce two sets of benchmark instances in this section. One set consists of real-world instances of the apparel application, whereas the other one comes from a pseudo-random instance generator designed to simulate real-world problems. The benchmark instances are summarized in Table 2.4, in which we present the type of the instances (Type), its notation (Notation), the number of instances it contains (Size) and some comments.

Table 2.4: CLSC benchmark instances summary

Type	Notation	Size	Comment
Application instances	IAP-A	3	
(IAP)	IAP-B	4	
Randomly generated	IRG-A	810	small size
(IRG)	IRG-B	108	medium to big size

We present the real-world instances, their characteristics and the data analysis in Section 2.4.1. Based on the real data analysis, the instance generator and characteristics of pseudo-randomly generated instances are given in Section 2.4.2.

2.4.1 Benchmark IAP: Real-World Instances and Data Analysis

We have collected 7 real-world instances so far. The characteristics of our application instances are summarized in Table 2.5, which include instance type, instance name, number of time buckets T, number of machines R, number of products N, number of demands D and the *capacity requirement ratio* Γ defined as

$$\Gamma := \frac{\sum_{d\in\mathcal{D}} pt_{p_d} q_d}{\sum_{r\in\mathcal{R},t\in\mathcal{T}} cap_{rt}}. \tag{2.38}$$

Although Γ does not consider the setup consumed capacity, it is an indicator of the machine loads. The larger it is, the more the machine is used. Moreover, instance R6

(R8) is based on instance R5 (R7) that it eliminates frozen horizon activities based on the current production plan.

Table 2.5: CLSC real-world benchmark instances

Type	Instance	T	R	N	D	Γ(%)	Comment
IAP-A	R1	27	3	3	313	99	
	R2	36	28	18	1188	30	
	R3	30	29	1	595	33	
IAP-B	R5	25	30	46	668	91	
	R6	25	30	36	431	74	R5 with frozen horizon
	R7	20	31	80	1428	40	
	R8	20	31	73	1404	41	R7 with frozen horizon

Calculation of the capacity requirement ratio is actually data analysis. Performing data analysis in a real-world application is often necessary and important. First, it helps to detect possible data error which is a common issue in practice. Second, it helps to discover the data structure and may lead to efficient tailored heuristics. Third, due to the fact that data collection and verification might be a long and struggling process, randomly generated instances are needed to develop optimization engine for production planning. Therefore, in the rest of this section, we take instance R5 as a representative to perform analysis. There are many parameters in our problem, therefore we group these parameters into following features first and perform analysis from these perspectives:

1. Problem size: N, R, D, T.
2. Machine capacity: cap_{rt}.
3. Production time: pt_i.
4. Setup time and cost: st_{ir} and sc_{ir}.
5. Product and demand distribution: p_d.
6. Demand quantity: q_d.
7. Time window of demand: b_d, e_d^1, e_d^2.
8. Demand cost structure: tc_d^1, tc_d^2, lc_d.

First of all, the size of instance R5 is given in Table 2.5, which has 25 time buckets corresponding to around 6 months planning horizon, 46 products, 30 parallel machines

with different capacities and 668 demands. The detailed data analysis is given in Table A.1 - A.5 in Appendix.

In Figure 2.5, for each machine r, we give its minimum capacity, maximum capacity and average capacity over its nonzero-capacity time buckets, and capacities for each time bucket. We have following observations: 1) Machine capacities vary between 1,000,000 and 4,000,000. Parallel machines have very different capacity layout. 2) Even for one machine, its capacity changes from time bucket to time bucket. However, it seems that over most time buckets, the capacity is relatively stable around one level for most machines. In other time buckets, the capacity has a decrease. 3) Although there are irregular machines such as $r3$ which only has nonzero-capacity in $t2$, the overall capacity trends for all machines have similar pattern. For example, in time bucket $t10$ and $t12$, there is a decrease for all machines' capacities. This is mainly due to national holidays.

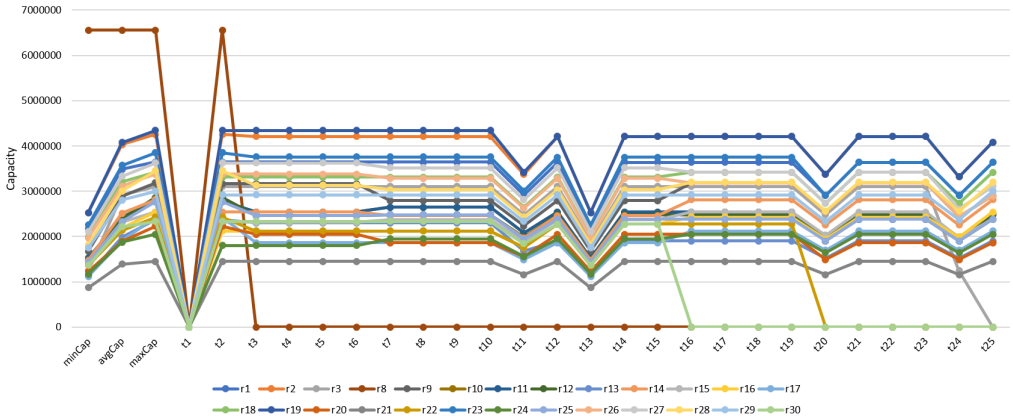


Figure 2.5: CLSC instance R5 analysis: machine capacity distribution

In Figure 2.6, we present the unit production time for each product. For example, to produce one unit of product 1, the production time is 1529. For each product, the number of demands $|\mathcal{D}^i|$ is also shown in the right axis. We cannot conclude a distribution pattern for the production time, and there is no relationship between the production time and its demand size. In most cases, the production time is between 1000 and 2000 seconds.

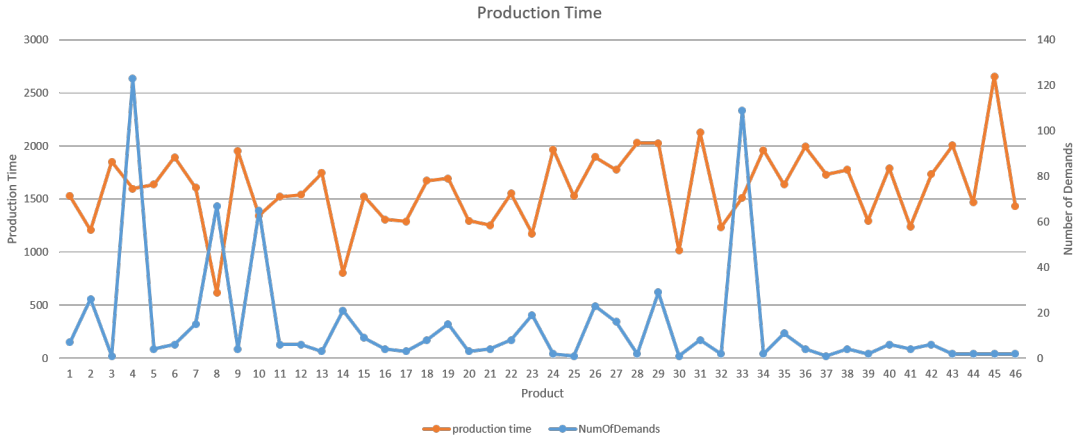


Figure 2.6: CLSC instance R5 analysis: production time distribution

In Figure 2.7, for each machine, we give the average setup time over its capacity ratio, which is calculated as $\frac{1/N \cdot \sum_{i \in \mathcal{N}} st_{ir}}{\frac{\sum_{t \in \mathcal{T}, cap_{rt} > 0} cap_{rt}}{\sum_{t \in \mathcal{T}, cap_{rt} > 0} 1}}$. We observe that the setup time is on average very large that it takes 40% to 50% of the capacity in each time bucket. The setup cost equals to the setup time times unitary labor cost.

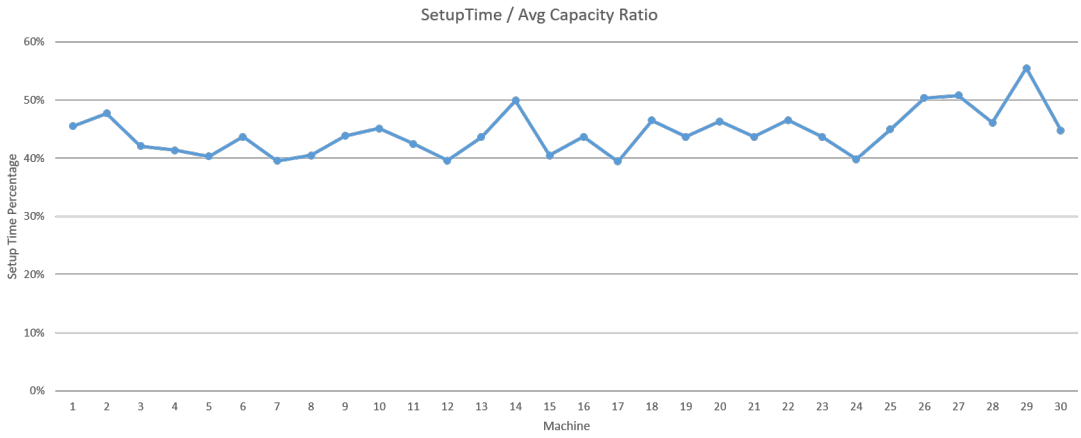


Figure 2.7: CLSC instance R5 analysis: setup time distribution

In Figure 2.8, we show the product distribution. For each product i, the left vertical axis represents the number of demands that require this product, i.e., $|\mathcal{D}^i|$. For each product i, the right vertical axis shows the percentage of its required capacity over all products, which is calculated by $\frac{\sum_{d \in \mathcal{D}^i} pt_i q_d}{\sum_{j \in \mathcal{N}} \sum_{d \in \mathcal{D}^j} pt_j q_d}$. For example, product 1 has 7 demands and is responsible for 1.19% of total capacity requirement. We have the following observations: 1) Number of demands and capacity requirement of each product are not

always positively correlated. For example, Product 4 has most number of demands as 123, but its capacity requirement takes 19.77%, which is less comparing to Product 33. This is due to difference in production time of different demands. 2) There are several "big" products, such as Product 33 and Product 4. Overall, 20% products covers more than 80% of demand capacity requirement.

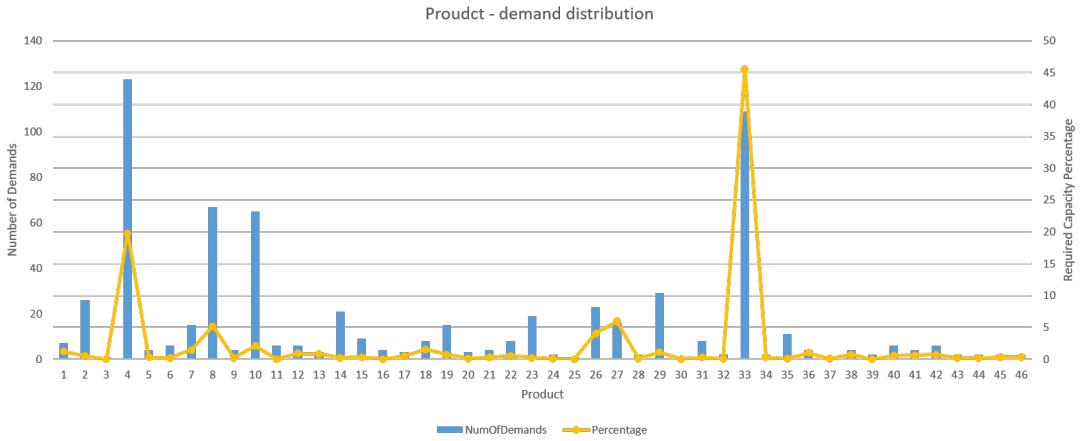


Figure 2.8: CLSC instance R5 analysis: product-demand distribution

In Figure 2.9, the quantity of each demand is sorted in nondecreasing order. The quantity distribution seems to share the same shape as an exponential function.

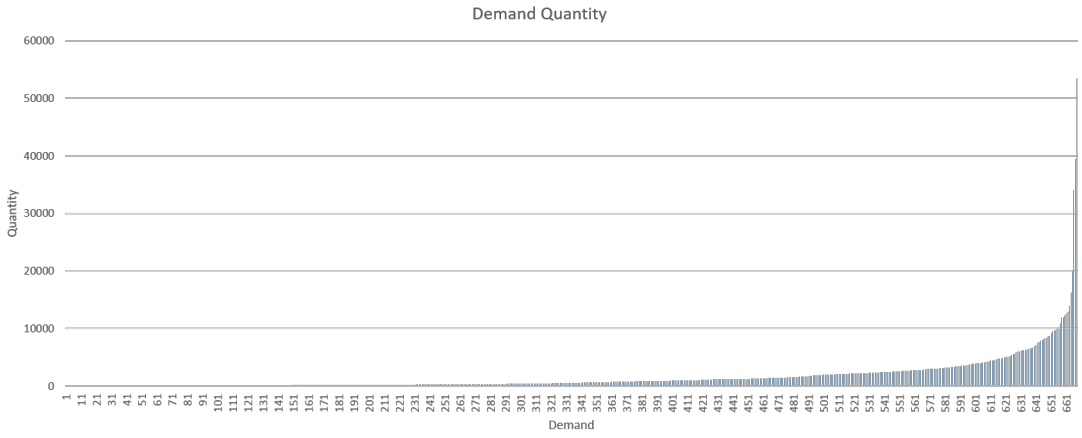


Figure 2.9: CLSC instance R5 analysis: demand quantity distribution

In Figure 2.10, . The release time and the first due date construct a on-time time window for each demand, the length of which equals to $e_d^1 - r_d$. The same also applies to the second due date and first due date. In Figure 2.10, the horizontal axis represents all

possible time window length while the vertical axis represents the number of demands that falls into this category. The blue lines corresponds to the length of interval between release date and first due date $e_d^1 - r_d$, the orange lines corresponds to the length of interval between first due date and second due date $e_d^2 - e_d^1$. For example, there are 91 demands the length $e_d^1 - r_d$ equals to 1, which means that they have 1 time bucket to be produced and satisfied on time. We observe that more than 95% of demands has the time interval value $e_d^1 - r_d$ and $e_d^2 - e_d^1$ less than or equals to 4. Moreover, more than 45% of demands has the value $e_d^1 - r_d = 4$ and more than 68% of demands has the value $e_d^2 - e_d^1 = 4$. This is in accordance with the fact that there is normally 1 month between the release date and the first due date.

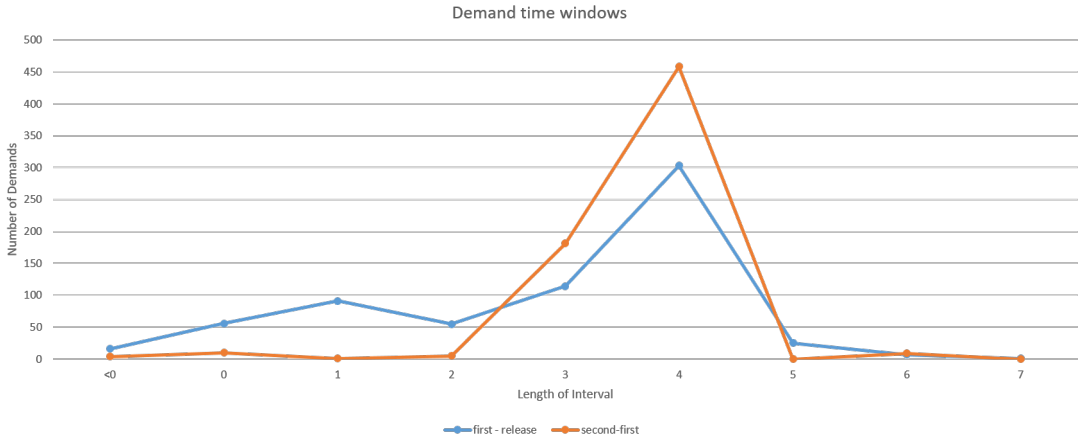


Figure 2.10: CLSC instance R5 analysis: demand time window distribution

Next we analyze the demand release date, first due date and second due date distribution. In Figure 2.11, the horizontal axis represents the time bucket $t \in \mathcal{T}$, while the vertical axis represents the number of demands. The blue curve corresponds to the number of demands that are released at t, i.e., $r_d = t$. The red curve corresponds to the number of demands that are first due at t, i.e., $e_d^1 = t$, whereas the green curve corresponds to the number of demands that are second due at t, i.e., $e_d^2 = t$.

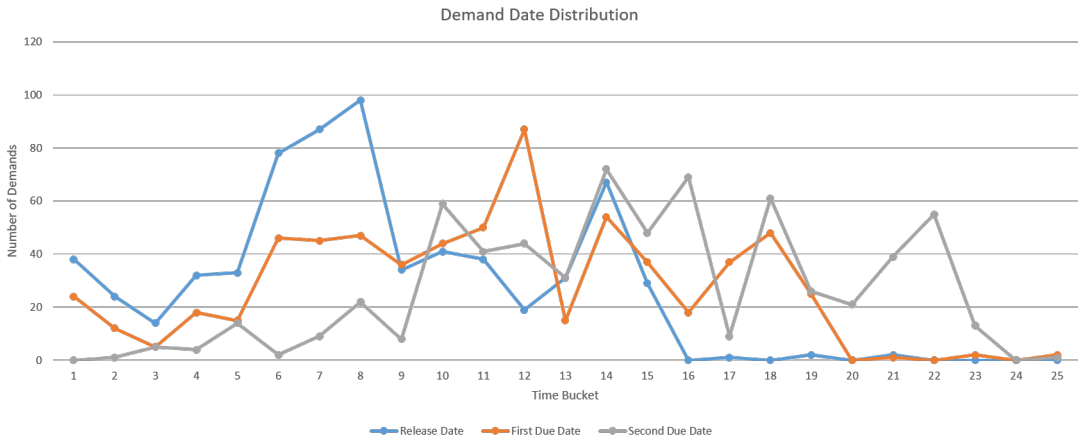


Figure 2.11: CLSC instance R5 analysis: demand release/due date distribution

We have following observations:

- 99% of demands are released in the first 60% time buckets ($t = 1$ to $t = 15$). Moreover, the number of released demands has a peak in the middle of these time buckets.
- 99% of demands are due in the first 76% time buckets ($t = 1$ to $t = 19$). This matches the on-time window length we have observed before.
- 88% of demands have second due date in the last 60% time buckets ($t = 1$ to $t = 19$). This matches the window length $e_d^2 - e_d^1$ we have observed before.
- All dates distribution seem to have a similar shape as the possibility density function of nominal distribution.

Based on this analysis, we can analyze the capacity requirement based on time window instead of on the whole time horizon. For each consecutive time window with length 4 $[t, t+3]$, we calculate the set of demands $\mathcal{D}_t$ that have $[r_d, e_d^1 - 1] \subseteq [t, t+3]$. Then for each time interval $[t, t+3]$, the percentage of demands $\frac{|\mathcal{D}_t|}{D}$ is given as the blue columns on the left vertical axis, while their total required capacity ratio as $\frac{\sum_{d \in \mathcal{D}_t} pt_{p_d} q_d}{\sum_{r \in \mathcal{R}, t \leq s \leq t+3} cap_{rt}}$ is given as the orange dot in the right vertical axis. This value reflect that to produce and satisfy all demands on time, what is the required capacity. For example, in the time interval $[t1, t4]$, we have 7.63% demands released and should be satisfied in this interval to be on time. Their required production capacity takes 115% of the total available capacity in this interval. By looking at the graph and values, we observe that most of demands requirement are concentrated on the first 68% of the time horizon. Therefore,

the capacity constraint is tight on this part of the time horizon and if there is not enough capacity many demands will be pushed late.

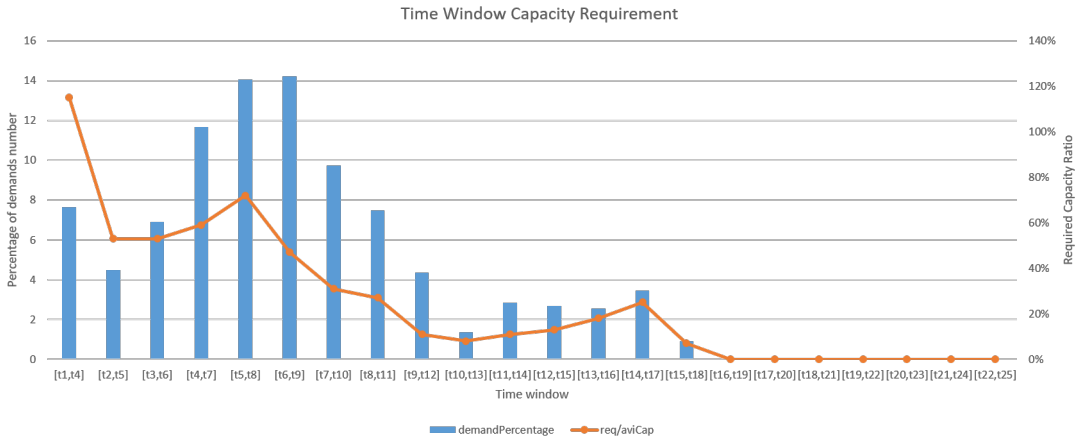


Figure 2.12: CLSC instance R5 analysis: capacity requirement by time interval

Finally, we analyze the cost structure. In this instance, the unit lost sale cost lc_d is set artificially as 1000, while the first level tardiness cost is 2.5 and the second level tardiness cost is 14 for all demands. To have a better view, we show the tardiness cost structure of instance R7 in the following Figure 2.13. We have following observations: 1) For the first level tardiness cost, 80% of demands have tc_d^1 as 2.5 and 8% of demands have tc_d^1 as 1.8. 2) For the second level tardiness cost, all of them have the cost as 14. Based on these two instances, we conclude that the first level tardiness cost has a large possibility to be 2.5 while the second level tardiness cost equals to 14.

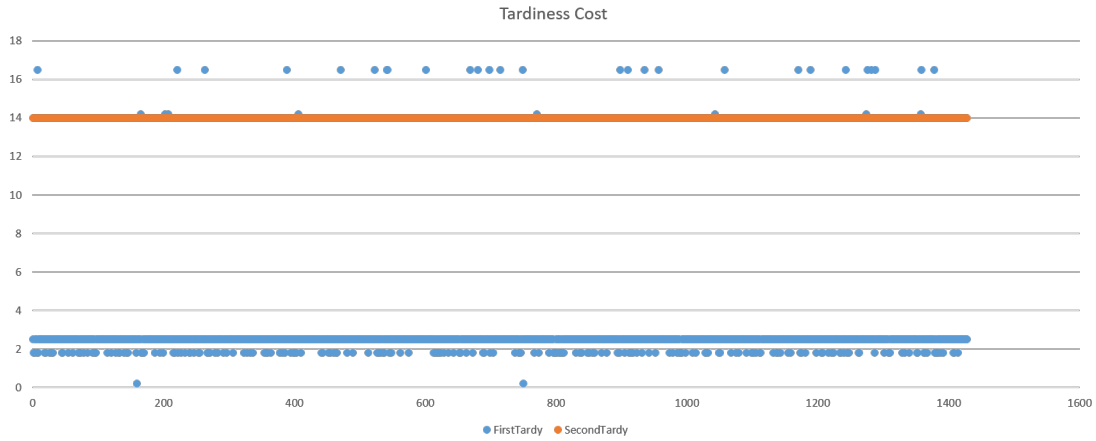


Figure 2.13: CLSC instance R5 analysis: tardiness cost distribution

2.4.2 Benchmark IRG: Pseudo-Randomly Generated Instances

Due to the limited number of real-world instances, it requires us to generate benchmark instances to perform computational test. In this section, we propose an instance generator and 2 sets of instances, of which the number of contained instances are summarized in Table 2.6:

Table 2.6: CLSC pseudo-randomly generated benchmark instances

Notation	Size	T	N	R	D	Γ(%)
IRG-A	810	{4,9,13}	{4,8,12}	{1,5,10}	{50,100,200}	{75,90}
IRG-B	108	{25}	{50,75,100}	{15,20,30}	{500,750,1000}	{75,90}

Instance Generator

We introduce an instance generator which is able to produce instances of CLSC with different characteristics.

The input parameters for the instance generator is given in Table 2.7:

Table 2.7: CLSC instance generator parameters

Data	IG Parameter	Description
T, R, N, D	T, R, N, D	Instance size
p_d	$size(p)$, $weight(p)$	Approximately $weight(p)$ % of demands belong to the first $size(p)$ % of products.
pt_i	$lbd(pt)$, $ubd(pt)$	$pt_d \in [lbd(pt), ubd(pt)]$, $pt_i = \left\lceil \frac{\sum_{d\in\mathcal{D}^i} pt_d q_d}{\sum_{d\in\mathcal{D}^i} q_d} \right\rceil$
q_d	$lbd(q)$, $mbd(q)$, $ubd(q)$, $mpt(q)$	Approximately with possibility $mpt(q)$, $q_d \in [lbd(q), mbd(q)]$; otherwise, $q_d \in [mbd(q), ubd(q)]$.
e_d^1	$size1(d)$,$size2(d)$, $prob1(d)$, $prob2(d)$	With possibility $prob1(d)$, $e_d^1 \in$ first $size1(d)$% part of time horizon; with possibility $prob2(d)$, $e_d^1 \in$ second $size1(d)$% part of time horizon; otherwise, $e_d^1 \in$ remaining part of time horizon.
r_d	$pf(dr)$	With possibility $pf(dr)$, $r_d = \max\{1, e_d^1 - 4\}$; otherwise, $r_d \in \max\{1, [e_d^1 - 3, e_d^1 - 1]\}$.

Table 2.7: CLSC instance generator parameters (continued)

Data	IG Parameter	Description
e_d^2	$pf(dd)$	With possibility $pf(dd)$, $e_d^2 = \min\{T+1, e_d^1+4\}$; otherwise, $e_d^2 \in \min\{T+1, [e_d^1+1, e_d^1+3]\}$. If $e_d^2 == T+1$, it implies that there is no second due date.
tc_d^1	$lbd(tc^1)$, $ubd(tc^1)$	$tc_d^1 \in [lbd(tc^1), ubd(tc^1)]$
tc_d^2	$lbd(tc^2)$, $ubd(tc^2)$	$tc_d^2 \in [lbd(tc^2), ubd(tc^2)]$
lc_d	$lbd(lc)$, $ubd(lc)$	$lc_d \in [lbd(lc), ubd(lc)]$
cap_{rt}	R, $Type(cap)$	Types of capacity allocation and target demand capacity ratio.
st_{ir}	$lbd(st)$, $ubd(st)$	$st_{ir} \in [lbd(st)cap_{rt}, ubd(st)cap_{rt}]$
sc_{ir}	$lbd(sc)$, $ubd(sc)$	$sc_{ir} \in [lbd(sc), ubd(sc)]$ and is positively proportional to st_{ir}.

The procedure of the instance generator is described in the following, which uses above parameters to realize all instance data. Let $randI(l,u)$ represent a random integer number between l and u inclusively following uniform distribution, while $randD(l,u)$ represent a random real number between l (inclusive) and u (exclusively) following uniform distribution.

1. Instance size T, M, N, D is given by parameters directly. Without loss of generality, $N \leq D$.

2. Demand product p_d: $\alpha \leftarrow randD(0,1)$ and $\beta \leftarrow \lfloor N \cdot size(p) \rfloor$.
   If $\alpha < weight(p)$ and $\beta > 0$, $p_d \leftarrow randI(1,\beta)$.
   Otherwise, $p_d \leftarrow randI(\beta+1, N)$.

3. pt_i: for each demand $d \in \mathcal{D}^i$, $pt_d = randI(lbd(pt), ubd(pt))$, then $pt_i = \left\lceil \frac{\sum_{d\in\mathcal{D}^i} pt_d q_d}{\sum_{d\in\mathcal{D}^i} q_d} \right\rceil$.

4. Demand quantity q_d: $\alpha \leftarrow randD(0,1)$.
   If $\alpha < mpt(q)$, $q_d \leftarrow randI(lbd(q), mbd(q))$.
   Otherwise, $q_d \leftarrow randI(mbd(q), ubd(q))$.

5. Demand first due date e_d^1: $\alpha \leftarrow randD(0,1)$, $\beta \leftarrow \lfloor T \cdot size1(d) \rfloor$, $\gamma \leftarrow \lfloor T \cdot size2(d) \rfloor$.
   If $\alpha \leq prob1(d)$ and $\beta > 0$, $e_d^1 \leftarrow randI(1,\beta)$.

Else if $\alpha \leq prob2(d)$ and $\gamma > 0$, $e_d^1 \leftarrow randI(\beta + 1, \gamma)$.
Otherwise, $e_d^1 \leftarrow randI(\gamma + 1, T)$.

6. Demand release time b_d: $\alpha \leftarrow randD(0, 1)$.
If $\alpha \leq pf(dr)$, $r_d = \max\{1, e_d^1 - 4\}$.
Otherwise, $r_d \in \max\{1, [e_d^1 - 3, e_d^1 - 1]\}$.

7. Demand second due date e_d^2: $\alpha \leftarrow randD(0, 1)$.
If $\alpha \leq pf(dd)$, $e_d^2 = \min\{T + 1, e_d^1 + 4\}$.
Otherwise, $e_d^2 \in \min\{T + 1, [e_d^1 + 1, e_d^1 + 3]\}$.
If $e_d^2 == T + 1$, it implies that there is no second due date.

8. Demand tardiness cost $tc_d^1 \leftarrow randI(lbd(tc^1), ubd(tc^1))$, $tc_d^2 \leftarrow randI(lbd(tc^2), ubd(tc^2))$.

9. Demand lost sale cost $lc_d \leftarrow randI(lbd(lc), ubd(lc))$.

10. Machine capacity: let $lbd = \lfloor 0.75b \rfloor$, $ubd = \lfloor 1.25b \rfloor$ where $b = \frac{\sum_{d \in \mathcal{D}} pt_i q_d}{M \cdot T \cdot R}$.

    - If $Type(cap) = Constant$, $cap_r \leftarrow randI(lbd, ubd)$, $cap_{rt} = cap_r$ for $r \in \mathcal{R}$.
    - If $Type(cap) = Various$, $cap_{rt} \leftarrow randI(lbd, ubd)$.
    - If $Type(cap) = TwoLevel$, let $lowCapRatio = 0.5$ and $lowTBSize = 0.2$,
      - Randomly select a subset of time buckets $\hat{T} \subseteq \mathcal{T}$ and $|\hat{T}| = \lfloor lowTBSize \cdot T \rfloor$.
      - For each machine $r \in \mathcal{R}$, $avgCap_r \leftarrow randI(lbd, ubd)$,
        $uCap_r \leftarrow \lfloor avgCap_r / ((1 - lowTBSize) + lowTBSize * lowCapRatio) \rfloor$,
        $lCap_r \leftarrow ucap_r \cdot lowCapRatio$.
      - For $t \in \mathcal{T}$, if $t \in \hat{T}$ $cap_{rt} \leftarrow lCap_r$; otherwise, $cap_{rt} \leftarrow uCap_r$.

11. Setup time st_{ir}: For $r \in \mathcal{R}$, let $avgCap_r = \left\lfloor \frac{\sum_{t \in \mathcal{T}} cap_{rt}}{\sum_{t \in \mathcal{T}, cap_{rt} > 0} 1} \right\rfloor$. For $i \in \mathcal{N}$, $st_{ir} \leftarrow randI(\lfloor lbd(st) \cdot avgCap_r \rfloor, \lfloor ubd(st) \cdot avgCap_r \rfloor)$.

12. Setup cost. Let $minSt = \min_{i \in \mathcal{N}, r \in \mathcal{R}} st_{ir}$, $maxSt = \max_{i \in \mathcal{N}, r \in \mathcal{R}} st_{ir}$, which are the minimum and maximum of setup time.
$a := (ubd(sc) - lbd(sc))/(maxSt - minSt)$; $b := (lbd(sc) * maxSt - ubd(sc) * minSt)/(maxSt - minSt)$;
If $minST == maxST$, set all $sc_{ir} \leftarrow lbd(sc)$.
Otherwise $sc_{ir} \leftarrow a \cdot st_{ir} + b$.

Pseudo-randomly generated benchmark Instances

To study the problem property from computational point of view, as well as developing efficient heuristic algorithms to tackle it, we propose two sets of pseudo randomly generated instances in this section with different characteristics.

Simple structure instances First of all, we generate a set of small size instances of which the optimal objective value might be reached. In this way, we may easily compare the performances of different MIP formulations. For each possible combination of the parameters, we generate 5 instances to limit bias. Other parameters for the randomly generated instances are described as follows:

- All combinations of T, M, N, D with following values $T \in \{4, 9, 13\}$, $M \in \{1, 5, 10\}$, $N \in \{4, 8, 12\}$, $D \in \{50, 100, 200\}$.
- $size(p) = 100.0$, $weight(p) = 100.0$
- $lbd(pt) = 20$, $ubd(pt) = 40$. Also, $pt_i := \left\lfloor \frac{\sum_{d \in \mathcal{D}^i} pt_d}{\sum_{d \in \mathcal{D}^i} 1} \right\rfloor$
- $lbd(q) = 1$, $mbd(q) = 10$, $ubd(q) = 10$, $mpt(q) = 1.0$
- $size1(d) = 30$,$size2(d) = 70$, $prob1(d) = 0.1$, $prob2(d) = 0.9$
- b_d has 80% possibility to be set as $\max\{0, e_d^1 - a\}$ and 20% possibility to be set as $\max\{0, e_d^1 - a/2\}$ where $a = \lceil \sqrt{T} \rceil$.
- e_d^2 has 80% possibility to be set as $\max\{0, e_d^1 - a\}$ and 20% possibility to be set as e_d^1.
- $lbd(tc^1) = 1$, $ubd(tc^1) = 1$
- $lbd(tc^2) = 5$, $ubd(tc^2) = 5$
- $lbd(lc) = 20$, $ubd(lc) = 20$
- $cap_{rt} = rand(0.75b, 1.25b)$ where $b = \frac{\sum_{d \in \mathcal{D}} cap_d q_d}{M \cdot T}$.
- $st_{ir} = st_i = rand(0.75c, 1.25c)$ where $c = 0.1 \cdot \frac{\sum_{r \in \mathcal{R}, t \in \mathcal{T}} cap_{rt}}{M \cdot T}$.

Application based instances. In Section 2.4.1, we have presented the data analysis based on a real application instance. Based on this analysis, we propose our first set of testing instances with following instance generator parameter values:

- Three size of instance including $(T, M, N, D) = (25, 15, 50, 500)$, $(25, 20, 75, 750)$ and $(25, 30, 100, 1000)$.
- $size(p) = 0.2$, $weight(p) = 0.8$
- $lbd(pt) = 100$, $ubd(pt) = 200$
- $lbd(q) = 1$, $mbd(q) = 1000$, $ubd(q) = 5000$, $mpt(q) = 0.95$
- $size1(d) = 20$, $size2(d) = 60$, $prob1(d) = 0.1$, $prob2(d) = 0.85$
- $pf(dr) = 0.5$
- $pf(dd) = 0.7$
- $lbd(tc^1) = 1$, $ubd(tc^1) = 2$
- $lbd(tc^2) = 10$, $ubd(tc^2) = 10$
- $lbd(lc) = 200$, $ubd(lc) = 200$
- $R \in \{0.75, 0.90\}$, $Type(cap) = TwoLevel$
- $lbd(st) = 0.4$, $ubd(st) = 0.5$
- $lbd(sc) = 0$, $ubd(sc) = 0$

To introduce certain level of varieties, propose following instances with different features. All instances have the same instance generator parameters value unless specified:

- Equally distributed products (ProdEven):
  - $size(p) = 100.0$, $weight(p) = 100.0$
- Constant capacity (Capconst):
  - $Type(cap) = Constant$
- Different capacity (Capdiff):
  - $Type(cap) = Various$
- Small setup time ratio (STLow):
  - $lbd(st) = 0.1$, $ubd(st) = 0.2$
- Evenly distributed demand due date (DemEven):

– $size1(d) = 100$, $prob1(d) = 1.0$, $size2(d) = 0$, $prob2(d) = 0.0$

For each possible combination of the parameters, we generate 3 different instances to limit bias.

2.5 Empirical Evaluations

In this section, we present experimental results to study CLSC from computational perspective. All experiments run on computer with Intel Core i7-4790 2.50 GHz 3.60, 16 GB shared memory, under the Linux Ubuntu 12.4 operating system.

First of all, different MIP formulations are compared. All developed formulations are solved for benchmark IRG-A and IAP-A with standard MIP solver CPLEX 12.6.1. In the practical application, lost sales cost and tardiness cost have higher priority than setup cost. Therefore, tests are done considering setup cost and without considering setup cost in Section 2.5.1 and Section 2.5.2 respectively.

Second, many features are involved in CLSC. In Section 2.5.3, we test different variants of CLSC on one application instance by relaxing one feature at a time. In this way, we analyze the impact of features to the problem difficulty.

Furthermore, based on the insight given by the formulation comparison test, one formulation with overall best performance is chosen. Benchmark IRG-B and IAP-B are solved with the chosen formulation by standard MIP solver CPLEX 12.6.1 on full cores given 1 hour time limit. The results are given in Section 2.5.4.

2.5.1 MIP Formulation Comparison Considering Setup Cost

In the Table 2.8, we present the summary computational results using CPLEX to solve the MIP models and their LP relaxations on the benchmark instances IRG-A and IAP-A with standard MIP solver CPLEX 12.6.1 on one thread given 10 minutes time limit.

In the table, the computing time is expressed in seconds. For each instance parameter (T, R, D, N, Γ) and values, we give the average results over all instances that have the corresponding value. In the Row T/A, averages values over all tested instances are reported while its Column Opt reports the total number of optimally solved instances for each formulation. In the first two columns, we present the parameters and their values. For example, for number of time buckets T, there are three values $\{4, 9, 13\}$ for IRG-A instances. In Column Opt and Time, we report the number and the average computing time over all instances with given parameter value solved to prove optimality within the time limit. In Column Nodes and Gap, we report the number of explored nodes and the

exit gap when CPLEX terminates. This gap represents the relative difference between the primal and the dual bounds computed by the CPLEX at the time limit. In Column LPT, it reports the average computational time to solve the LP relaxation over all the instances sharing this parameter value. In Column LPG, we measure the quality of the LP relaxation which is calculated as

$$LPG = \frac{BestMip - LPVal}{BestMip}, \tag{2.39}$$

where $bestMip$ is the best known MIP solution of all formulations and $LPVal$ is the optimal LP relaxation objective value of the given formulation. On all instances except two, the three formulations $Form1$, $Form2$ and $Form3$ are characterized by the same $LPVal$. In the remaining two cases the difference is less than 0.001, thus we only report the LPG once under column $Form3$. For IAP-A instances, we report the objective function values returned by the solver in Column Obj.

As far as the computing time necessary to calculate the LP relaxation is concerned, the average values over all tested instances are 0.37 seconds, 0.74 seconds and 0.30 seconds for $Form1$, $Form2$ and $Form3$ respectively. For IRG-A instances, the average computational time of the LP relaxation are 0.24, 0.26 and 0.20 seconds, which are quite close. Therefore, the difference mainly comes from IAP-A instances. Thanks to the shortest average computing time for the LP relaxation, $Form3$ is able to explore more nodes within the given time limit.

According to the Column Opt, the table shows that as the problem size increases, instances become more difficult to solve. The parameter N, which has an impact on the number of binary variables, affects most the solvability of the instances. Take $Form2$ for example, when N increases from 4 to 12, the number of instances decreases from 258 to 59, whereas when D increases from 50 to 200, the number is only reduced by 36. The number of time buckets T has a smaller impact on the computing time. We can observe that $Form1$ explores a higher number of nodes, this is probably due to the fact that it struggles to find good quality integer solutions. On the other hand, $Form2$ and $Form3$ explore almost the same number of nodes for the randomly generated instances, while the $Form3$ explores more nodes for the 3 real-world instances. Regarding the number of randomly generated instances solved to be proven optimality, $Form3$ solves 416 instances and $Form2$ solves 411 instances, while $Form1$ solves only 379. A similar behavior can be also observed for the exit gap. For the real-world instances, we observe similar results, i.e., the $Form3$ shows the best performance. Hence, according to the computational experiments, $Form3$ shows the best overall computational performance. This is due to the fact that it has the least number of constraints with the same number

of binary variables compared to the other two formulations, and its LP relaxation can be solved faster.

Next we compare two models of different production quantity decisions. Since $Form3$ performs best among three different setup models, we fix the setup carryover model based on $Form3$ therefore compare the formulation $Form3$ and $Form3^{FL}$. The average computational time of the LP relaxation of $Form3^{FL}$ on IRG-A is 1.60 seconds while for $Form3$ it is only 0.2. For the real instance $R2$, CPLEX even fails to solve the LP relaxation of $Form3^{FL}$ optimally given 10 minutes time limit. With the price of longer computational time, $Form3^{FL}$ shows a lower LP Gap, as proven in the Theorem 2.4. However, even though $Form3^{FL}$ has a stronger lower bound, due to its longer computational time on each node, it only solves 365 instances out of 810 instances to optimality. In summary, $Form3$ still gives better performance comparing to $Form3^{FL}$, therefore gives overall best performance among all presented formulations.

Table 2.8: CLSC formulation comparison with setup cost

		$Form1$					$Form2$					$Form3$						$Form3^{FL}$					
Para	Val	Opt	Time	Nodes	Gap	LPT	Opt	Time	Nodes	Gap	LPT	Opt	Time	Nodes	Gap	LPT	LPG	Opt	Time	Nodes	Gap	LPT	LPG
T	4	165	32	85970	4.64	0.0	175	35	49088	4.16	0.0	178	36	55347	3.89	0.0	31.01	158	39	18641	5.66	0.5	16.20
	9	123	69	41168	15.81	0.2	128	61	26469	15.28	0.2	131	65	31207	14.95	0.2	51.19	115	89	17907	18.98	1.6	35.92
	13	91	112	35839	24.17	0.5	108	102	20961	23.34	0.5	107	97	25667	23.13	0.4	59.98	92	105	16089	27.11	2.7	48.74
R	1	212	48	18748	2.70	0.0	217	55	12988	2.50	0.0	218	53	13563	2.75	0.0	65.30	226	48	21390	2.34	0.0	32.36
	5	97	68	50892	23.18	0.2	101	46	30784	22.46	0.2	105	58	37983	21.93	0.1	48.48	83	93	16101	27.17	1.1	41.29
	10	70	104	93338	18.74	0.5	93	91	52746	17.82	0.6	93	81	60674	17.29	0.5	28.40	56	136	15145	22.25	3.7	27.21
N	4	231	47	58790	1.40	0.1	258	47	25497	0.51	0.0	258	44	22985	0.59	0.0	35.07	212	59	9025	2.98	0.6	23.91
	8	87	95	64440	17.23	0.2	94	87	46402	16.48	0.2	94	86	55293	16.25	0.2	49.80	89	104	21291	19.95	1.6	35.04
	12	61	83	39746	25.99	0.4	59	81	24619	25.78	0.5	64	93	33943	25.12	0.4	57.31	64	68	22321	28.83	2.6	41.92
D	50	151	66	63995	11.75	0.2	156	53	35211	11.13	0.2	159	58	43104	10.64	0.1	46.81	149	56	23920	13.58	0.4	36.91
	100	126	56	47988	15.78	0.2	135	60	32839	15.06	0.3	137	57	37585	14.76	0.2	48.18	129	89	18361	17.67	1.0	33.79
	200	102	68	50994	17.09	0.3	120	72	28469	16.60	0.4	120	70	31532	16.56	0.3	47.19	87	73	10355	20.51	3.4	30.17
Γ	75	211	60	41299	14.53	0.2	226	50	23728	14.01	0.2	230	52	29214	13.60	0.1	49.07	206	67	13839	16.77	1.2	33.88
	90	168	68	67353	15.22	0.3	185	74	40617	14.51	0.3	186	72	45600	14.37	0.3	45.72	159	77	21252	17.74	2.0	33.36
T/A		379	63	54326	14.87	0.2	411	61	32173	14.24	0.3	416	61	37407	13.99	0.2	47.39	365	72	17546	17.25	1.6	33.62
Inst		Obj	Time	Nodes	Gap	LPT	Obj	Time	Nodes	Gap	LPT	Obj	Time	Nodes	Gap	LPT	LPG	Obj	Time	Nodes	Gap	LPT	LPG
R1		654,807	12	1827	0.00	0.1	654,807	2.6	434	0.00	0.0	654,807	3.1	404	0.00	0.0	14.25	654,807	34.7	469	0.00	0.5	14.03
R2		2,239,793	600	2	97.36	107.9	29,619,416	600	0	99.80	395.8	1,086,637	600	452	94.56	78.2	100	1E9	600	0	100	600	-
R3		10,672	0	0	0.00	0.0	10,672	0.0	0	0.00	0.0	10,672	0.0	0	0.00	0.0	0.00	10,672	2.1	0	0.00	0.9	0.00

2.5.2 MIP Formulation Comparison without Considering Setup Cost

In the apparel application, setup cost has secondary priority comparing to lost sale cost and tardiness cost. In other words, minimizing lost sale cost and tardiness cost is the most important objective. Therefore, in this section, we perform experiments on benchmark instances without setup cost but only setup time. In the Table 2.9, we present the summary computational results using CPLEX to solve the MIP models and its LP relaxations on the benchmark instances IRG-A and IAP-A with standard MIP solver CPLEX 12.6.1 on one thread given 10 minutes time limit. The table has the same layout as Table 2.8.

First of all, the optimal objective function values of LP relaxation of formulations $Form1$, $Form2$ and $Form3$ are the same except for 15 instances. For these 15 instances, $Form2$ and $Form3$ have larger objective function values than $Form1$, but the relative difference is less than 0.0001. Therefore, we only show the LP Gap under $Form3$ column defined as (2.39).

For the performance of formulations to solve the problem, we observe similar results as CLSC with setup cost. First, if we look at the overall performance over 810 instances of IRG-A, $Form3$ solved 476 instances to prove optimality with the average absolute gap equals to 4.06%. The number of instances that are solved to optimality for $Form1$, $Form2$ and $Form3^{FL}$ are 460, 468 and 424 respectively. Hence, the experimental results shows the superior performance among all developed formulations.

For the IAP-A instances, without setup cost, all of them are solved to prove optimality within a few seconds. However, due to the large formulation size, $Form3^{FL}$ took around 4 minutes to prove optimality of the solution.

Table 2.9: CLSC formulation comparison without setup cost

		$Form1$					$Form2$					$Form3$						$Form3^{FL}$					
Para	Val	Opt	Time	Nodes	Gap	LPT	Opt	Time	Nodes	Gap	LPT	Opt	Time	Nodes	Gap	LPT	LPG	Opt	Time	Nodes	Gap	LPT	LPG
T	4	205	33.0	57230	0.34	0.0	211	45.0	33768	0.32	0.0	214	39.4	36530	0.29	0.0	16.84	186	60.9	16830	0.58	0.2	6.9
	9	127	75.6	40526	5.10	0.1	123	40.1	26566	5.20	0.3	127	58.4	29364	5.11	0.2	25.58	104	47.2	16316	7.06	1.0	15.3
	13	128	35.8	31595	7.08	0.3	134	49.5	22915	6.92	0.6	135	42.4	25327	6.79	0.4	24.35	134	45.2	14521	9.73	1.5	17.2
R	1	214	37.2	18963	3.57	0.0	213	41.4	16839	3.81	0.0	215	37.8	17276	3.82	0.0	53.99	222	31.8	21773	3.43	0.0	25.3
	5	137	56.0	54654	5.44	0.1	145	59.5	29227	5.17	0.2	149	61.8	31683	5.05	0.2	8.78	119	84.4	13909	7.15	0.7	8.4
	10	109	48.7	55735	3.50	0.4	110	32.8	37183	3.46	0.7	112	37.7	42261	3.32	0.5	3.99	83	62.6	11985	6.79	2.0	5.7
N	4	219	24.8	33367	0.46	0.1	224	20.7	23845	0.35	0.0	227	29.3	23535	0.37	0.1	14.38	194	32.2	8974	1.33	0.4	8.1
	8	143	54.8	48945	4.49	0.2	147	54.0	30907	4.27	0.2	148	48.0	31840	4.24	0.2	23.91	130	68.4	17905	5.77	0.8	13.9
	12	98	78.3	47039	7.56	0.3	97	87.3	28498	7.83	0.7	101	77.5	35845	7.58	0.4	28.47	100	71.6	20788	10.28	1.6	17.4
D	50	153	44.1	59071	3.28	0.1	159	48.8	35406	3.11	0.3	161	44.4	40181	2.96	0.2	20.44	145	47.4	22467	3.76	0.3	14.8
	100	156	49.0	37713	3.97	0.2	155	45.8	26664	4.00	0.3	160	54.8	30210	3.92	0.2	22.93	143	47.4	15504	5.28	0.8	12.6
	200	151	43.4	32567	5.26	0.2	154	40.3	21179	5.34	0.4	155	36.5	20829	5.31	0.3	23.39	136	63.5	9696	8.32	1.7	12.0
Γ	0.75	299	36.3	34915	2.41	0.1	298	34.6	22200	2.43	0.2	299	29.0	25081	2.24	0.1	19.41	286	45.6	11490	2.95	0.5	9.3
	0.90	161	62.7	51319	5.93	0.3	170	63.2	33299	5.87	0.5	177	72.9	35733	5.89	0.4	25.10	138	66.9	20288	8.63	1.4	16.9
T/A		460	45.2	43117	4.17	0.2	468	44.7	27750	4.15	0.3	476	45.0	30407	4.06	0.2	22.25	424	52.8	15889	5.79	0.9	13.1
Inst		Obj	Time	Nodes	Gap	LPT		Time	Nodes	Gap	LPT		Time	Nodes	Gap	LPT	LPG		Time	Nodes	Gap	LPT	LPG
R1		442,906	0.5	51	0.00	0.1		0.3	14	0.00	0.0		0.3	8	0.00	0.1	0.83		19.5	400	0.00	0.4	0.83
R2		59,153	5.8	0	0.00	1.1		11.7	0	0.00	12.7		4.8	0	0.00	1.2	0.00		241.1	0	0.00	91.4	0.00
R3		10,672	0.1	0	0.00	0.1		0.1	0	0.00	0.0		0.1	0	0.00	0.0	0.00		2.2	0	0.00	1.3	0.00

2.5.3 Impact Analysis of Problem Features

CLSC consists of many features studied in CLSP. Here we perform experiments to understand the impact of each feature to the problem difficulty. Due to the priority difference of demand cost (lost sale cost and tardiness cost) and setup cost in the application, we solve CLSC without setup cost. As a reference, the original problem and its LP relaxation are solved as well and their results are shown in the row MIP and LP respectively. The formulation is $Form3$ since it gives the overall best performance. Other variants include:

- ~~Parallel machines~~: all parallel machines are aggregated into one machine 0 with capacity $cap_{0t} = \sum_{r \in \mathcal{R}} cap_{rt}$ for $t \in \mathcal{T}$.
- ~~Lost sale~~: no lost sales so that all demands must be satisfied.
- ~~Tardiness~~: each demand has only the first due date and it is either satisfied on time or lost.
- ~~Due dates~~: each demand has no due dates and can be satisfied any time by the end of the planning horizon without penalty cost.
- ~~First due date~~: each demand has only the second due date e_d^2. It is considered to be on time if satisfied before e_d^2 and it is considered to be delayed if satisfied at or after e_d^2 with a unitary tardiness cost of $tc_d^1 + tc_d^2$.
- ~~Release date~~: each demand has release date b_d equals to the beginning of the time horizon so that it can be satisfied from the very first time bucket.

The computational result is presented in Table 2.10. Each variant is solved by CPLEX with time limit 2 hours. For each problem, the returned objective function value, the computational time, the exit gap, the best lower bound returned by CPLEX and the number of explored nodes are given in Column Obj, Time, Gap, LB and # Nodes respectively.

We have following observations: first of all, the original problem is hard to solve that after 2 hours solving, the solution still has gap of 86%. Moreover, parallel machines contribute a lot to the problem difficulty that if we aggregate all machines, the problem can reach 0.1% gap after 2 hours of solving. Also, lost sales help CPLEX to find feasible solution. Actually, with developed heuristic algorithm, we find the near optimal solution of instance R5 and the lost sale cost equals to 0. However, when we set demands' satisfaction as a constraints, CPLEX has difficult time to find feasible solution. Eventually,

after 2 hours, no integer solution has been found and the feasibility of the problem is reported unknown. Without considering backlogging, each demand can either be on time or lost. We observe a small gap as 6.3% for this variant. However, the objective function value is on the level of 3×10^8, which might be the reason of the small gap. Therefore, we can not conclude that this variant is easier. In the case of ~~Due dates~~, different from ~~Tardiness~~, all demands can be satisfied any time after its release date and be considered on time. Therefore, it is equivalent to optimize only lost sale cost and provide a feasible solution to the original problem. Moreover, its objective represents the lost sale cost of the solution. Comparing to solving the original problem, the real objective function value of this solution ($>$ 37,900,675) is worse. The exit gap is 100% due to the lower bound of 0. Next variant is ~~First due date~~, which ignores the first due date and has a simpler cost structure. So the solution of this problem is also feasible for the original problem and its cost represents the lost sale cost and partial tardiness cost (tardiness cost due to second level due dates). Similar to no due dates, the solution quality is worse than that of the original problem. Therefore, considering backlogging seems to make the problem easier for CPLEX to solve. Finally, we cancel all the release dates, and the problem solution is quite similar to the original problem. On one hand, we were expecting a lower objective function value than the original problem if both this one and the original problem are solved to optimality because that there are less constraints on the demand satisfaction time. On the other hand, it is also possible that the release date function as cuts and speed up the solving processing. Therefore, the result is a bit unexpected and we will continue to test once we collect more realistic instances. In summary, parallel machines make the problem more difficult to solve while backlogging and lost sales seem to make the problem easier to solve, at least for the tested standard MIP solver.

Table 2.10: CLSC feature - complexity analysis

Variants	Time	Obj	LB	Gap(%)	# Nodes
LP	104	2,935,797	-	-	-
MIP	T.L.	22,564,237	2,975,645	86.8	0
~~Parallel machines~~	T.L.	3,082,657	3,078,520	0.1	151056
~~Lost sale~~	T.L.	-	-	-	-
~~Tardiness~~	T.L.	359,044,010	336,491,186	6.3	30
~~Due dates~~	T.L.	37,900,675	0	100.0	0
~~First due date~~	T.L.	113,773,295	634,788	99.4	0
~~Release date~~	T.L.	20,720,657	2,774,592	86.6	0

*Time unit in seconds, T.L. = 7200s

2.5.4 Computational Results on Benchmark IAP-B and IRG-B

We have performed formulation comparison on benchmark instances IAP-A and IRG-A. It is shown that formulation $Form3$ gives the best overall performance. Therefore, in this section, we present test on benchmark IAP-B and IRG-B without considering setup cost with $Form3$ to evaluate the problem difficulty and analyze the computational behavior on standard MIP solver. Computational results of CPLEX for benchmark IAP-B with 1 hour time limit is given in Table 2.11, while the result for benchmark IRG-B is given in Table 2.12 - 2.14.

For each instance, its characteristic are given in first six columns including the number of time buckets T, the number of machines R, the number of products N, the number of demands D and the capacity requirement ratio Γ defined as (2.38). We show the optimal objective value and the computational time of the LP relaxation in Column LP/Obj and LP/Time respectively. The computational results of CPLEX of the original MIP model are given in the MIP section. The returned objective function values, computational times, best known lower bounds and number of explored nodes are given in Column MIP/Obj, MIP/Time, MIP/LB and MIP/#Node. The Column BestLB gives best known lower bounds, which is the maximum between the optimal objective function value of the LP relaxation and the best known lower bound returned from the MIP solving $\max\{Lbd, LP/Obj\}$. The relative gap based on this best known lower bound, which equals to $\frac{MIP/Obj - BestLB}{\max\{1, MIP/Obj\}}$ is given in the Column MIP/Gap.

Table 2.11: Computational results: CPLEX on IAP-B

	Characteristics					LP		MIP					
Inst	T	R	N	D	Γ	Obj	Time	Obj	Time	Gap	LB	#Node	BestLB
R5	25	30	46	668	91	2,935,797	79	35,511,200	3600	91.6	2,973,702	0	2,973,702
R6	25	30	36	425	74	1,277,107	18	1,456,011	3600	7.7	1,344,501	97	1,344,501
R7	20	31	80	1428	40	2,217,260	118	2,692,957	3601	16.7	2,244,422	29	2,244,422
R8	20	31	73	1404	41	2,081,921	83	2,597,838	3600	18.7	2,111,181	0	2,111,181
AVG						2,128,021	74	10,564,501	3600	33.7	2,168,451	32	2,168,451

Table 2.12: Computational results: CPLEX on IRG-B (1)

	Characteristics						LP		MIP					
Inst	T	R	N	D	Γ	Type	Obj	Time	Obj	Time	Gap	Lbd	#Node	BestLB
B1	25	15	50	500	741	DF	0	22	329607	3600	100.0	0	0	0
B2	25	15	50	500	771	DF	0	20	769798	3600	100.0	0	0	0
B3	25	15	50	500	758	DF	0	18	69398	3600	100.0	0	398	0
B4	25	15	50	500	924	DF	136154	32	9485931	3600	98.5	144520	0	144520
B5	25	15	50	500	841	DF	0	23	1084421	3600	100.0	0	0	0
B6	25	15	50	500	896	DF	0	31	6291857	3600	100.0	0	0	0
B7	25	20	75	750	766	DF	0	81	88795819	3600	100.0	0	0	0
B8	25	20	75	750	732	DF	0	106	83386167	3600	100.0	0	0	0
B9	25	20	75	750	784	DF	0	85	85466741	3600	100.0	0	0	0
B10	25	20	75	750	851	DF	0	109	83980195	3600	100.0	0	0	0
B11	25	20	75	750	907	DF	0	134	90015737	3600	100.0	0	0	0
B12	25	20	75	750	871	DF	0	94	89549343	3600	100.0	0	0	0
B13	25	30	100	1000	758	DF	0	359	115106676	3600	100.0	0	0	0
B14	25	30	100	1000	739	DF	0	240	121228481	3600	100.0	0	0	0
B15	25	30	100	1000	737	DF	0	355	12857474	3600	100.0	0	0	0
B16	25	30	100	1000	898	DF	0	602	123816283	3600	100.0	0	0	0
B17	25	30	100	1000	938	DF	12126	309	127298893	3600	100.0	0	0	12126
B18	25	30	100	1000	909	DF	11563	329	116072579	3600	100.0	0	0	11563
AVG								164						
B19	25	15	50	500	692	ProdEven	0	39	132638	3600	100.0	0	0	0
B20	25	15	50	500	758	ProdEven	0	40	58077709	3600	100.0	0	0	0
B21	25	15	50	500	764	ProdEven	0	47	8404981	3600	100.0	0	0	0
B22	25	15	50	500	890	ProdEven	35419	45	13053180	3600	99.5	66276	0	66276
B23	25	15	50	500	914	ProdEven	0	60	62081602	3600	100.0	0	0	0
B24	25	15	50	500	852	ProdEven	0	53	10247860	3600	100.0	0	0	0
B25	25	20	75	750	769	ProdEven	0	163	92162919	3600	100.0	0	0	0
B26	25	20	75	750	769	ProdEven	0	171	91068198	3600	100.0	0	0	0
B27	25	20	75	750	758	ProdEven	0	152	67557529	3602	100.0	0	0	0
B28	25	20	75	750	941	ProdEven	5897	146	43534412	3600	99.9	23468	0	23468
B29	25	20	75	750	920	ProdEven	87616	151	32120464	3600	99.6	112633	0	112633
B30	25	20	75	750	931	ProdEven	15132	182	35640301	3600	99.9	37008	0	37008
B31	25	30	100	1000	789	ProdEven	0	599	127510049	3600	100.0	0	0	0
B32	25	30	100	1000	728	ProdEven	0	528	120922514	3600	100.0	0	0	0
B33	25	30	100	1000	784	ProdEven	0	637	119911245	3600	100.0	0	0	0
B34	25	30	100	1000	922	ProdEven	26135	391	118721109	3600	100.0	0	0	26135
B35	25	30	100	1000	924	ProdEven	203073	375	120294447	3600	99.8	0	0	203073
B36	25	30	100	1000	912	ProdEven	0	747	110465941	3600	100.0	0	0	0
AVG								251						

Table 2.13: Computational results: CPLEX on IRG-B (2)

	Characteristics						LP		MIP					
Inst	T	R	N	D	Γ	Type	Obj	Time	Obj	Time	Gap	Lbd	#Node	BestLB
B37	25	15	50	500	731	Capconst	0	18	23204	3600	100.0	0	1612	0
B38	25	15	50	500	785	Capconst	0	33	1163883	3600	100.0	0	0	0
B39	25	15	50	500	776	Capconst	0	21	625842	3600	100.0	0	0	0
B40	25	15	50	500	910	Capconst	0	41	18011097	3601	100.0	45	0	45
B41	25	15	50	500	843	Capconst	0	24	5487443	3600	100.0	0	0	0
B42	25	15	50	500	942	Capconst	18404	36	8834308	3600	99.7	30688	0	30688
B43	25	20	75	750	760	Capconst	0	70	83214545	3600	100.0	0	0	0
B44	25	20	75	750	694	Capconst	0	98	4254029	3600	100.0	0	0	0
B45	25	20	75	750	750	Capconst	0	59	5242020	3600	100.0	0	0	0
B46	25	20	75	750	919	Capconst	3292	84	92254335	3600	100.0	6124	0	6124
B47	25	20	75	750	906	Capconst	0	127	88539706	3600	100.0	0	0	0
B48	25	20	75	750	943	Capconst	48861	138	92440700	3600	99.9	65835	0	65835
B49	25	30	100	1000	753	Capconst	0	244	121162208	3600	100.0	0	0	0
B50	25	30	100	1000	750	Capconst	0	231	14690362	3600	100.0	0	0	0
B51	25	30	100	1000	792	Capconst	0	269	111052268	3600	100.0	0	0	0
B52	25	30	100	1000	900	Capconst	0	447	114397623	3600	100.0	0	0	0
B53	25	30	100	1000	928	Capconst	65636	291	114067019	3600	99.9	0	0	65636
B54	25	30	100	1000	949	Capconst	84790	287	113712126	3600	99.9	-30208	0	84790
AVG								140						
B55	25	15	50	500	752	Capdiff	0	23	257750	3600	100.0	0	0	0
B56	25	15	50	500	742	Capdiff	0	20	85133	3600	100.0	0	880	0
B57	25	15	50	500	738	Capdiff	0	22	18704	3600	100.0	0	1661	0
B58	25	15	50	500	894	Capdiff	0	32	7976495	3600	100.0	0	0	0
B59	25	15	50	500	897	Capdiff	0	31	7333092	3601	100.0	0	0	0
B60	25	15	50	500	898	Capdiff	2153	33	11514296	3600	99.8	21153	0	21153
B61	25	20	75	750	754	Capdiff	0	79	86537869	3600	100.0	0	0	0
B62	25	20	75	750	757	Capdiff	0	78	91957690	3600	100.0	0	0	0
B63	25	20	75	750	748	Capdiff	0	126	92125122	3600	100.0	0	0	0
B64	25	20	75	750	906	Capdiff	0	166	87013629	3600	100.0	0	0	0
B65	25	20	75	750	901	Capdiff	0	150	70307604	3600	100.0	0	0	0
B66	25	20	75	750	908	Capdiff	1098	78	20872475	3600	100.0	3773	0	3773
B67	25	30	100	1000	752	Capdiff	0	256	123548573	3600	100.0	0	0	0
B68	25	30	100	1000	751	Capdiff	0	260	117798084	3600	100.0	0	0	0
B69	25	30	100	1000	749	Capdiff	0	249	13873785	3600	100.0	0	0	0
B70	25	30	100	1000	897	Capdiff	0	473	113178684	3600	100.0	0	0	0
B71	25	30	100	1000	904	Capdiff	0	689	117260538	3600	100.0	0	0	0
B72	25	30	100	1000	907	Capdiff	427	171	117514463	3600	100.0	1653	0	1653
AVG								163						

Table 2.14: Computational results: CPLEX on IRG-B (3)

	Characteristics						LP		MIP					
Inst	T	R	N	D	Γ	Type	Obj	Time	Obj	Time	Gap	Lbd	#Node	BestLB
B73	25	15	50	500	753	STLow	0	10	0	37	0.0	0	0	0
B74	25	15	50	500	753	STLow	0	11	0	84	0.0	0	0	0
B75	25	15	50	500	752	STLow	0	14	0	277	0.0	0	0	0
B76	25	15	50	500	905	STLow	0	29	5148006	3600	100.0	0	0	0
B77	25	15	50	500	963	STLow	78579	25	7241378	3600	98.8	87085	0	87085
B78	25	15	50	500	944	STLow	128953	26	5381298	3601	97.5	132912	0	132912
B79	25	20	75	750	754	STLow	0	35	0	465	0.0	0	0	0
B80	25	20	75	750	767	STLow	0	59	0	1200	0.0	0	0	0
B81	25	20	75	750	755	STLow	0	40	0	1008	0.0	0	0	0
B82	25	20	75	750	907	STLow	0	78	87599277	3600	100.0	0	0	0
B83	25	20	75	750	961	STLow	142893	78	15890973	3600	99.1	146115	0	146115
B84	25	20	75	750	895	STLow	0	74	88968761	3600	100.0	0	0	0
B85	25	30	100	1000	747	STLow	0	184	0	1745	0.0	0	0	0
B86	25	30	100	1000	778	STLow	0	180	0	1809	0.0	0	0	0
B87	25	30	100	1000	767	STLow	0	175	0	2127	0.0	0	0	0
B88	25	30	100	1000	893	STLow	0	251	116164709	3600	100.0	0	0	0
B89	25	30	100	1000	900	STLow	0	332	120352296	3600	100.0	0	0	0
B90	25	30	100	1000	888	STLow	0	535	121208248	3600	100.0	0	0	0
AVG								119						
B91	25	15	50	500	757	DemEven	0	16	7154	3600	100.0	0	1698	0
B92	25	15	50	500	717	DemEven	0	20	1	3600	100.0	0	1845	0
B93	25	15	50	500	758	DemEven	0	19	21171	3600	100.0	0	1747	0
B94	25	15	50	500	874	DemEven	0	24	54451635	3600	100.0	0	0	0
B95	25	15	50	500	874	DemEven	0	29	7334313	3600	100.0	0	0	0
B96	25	15	50	500	902	DemEven	0	35	13755823	3600	100.0	0	0	0
B97	25	20	75	750	722	DemEven	0	56	83679	3600	100.0	0	0	0
B98	25	20	75	750	737	DemEven	0	61	93744125	3600	100.0	0	0	0
B99	25	20	75	750	757	DemEven	0	78	91176566	3600	100.0	0	0	0
B100	25	20	75	750	944	DemEven	0	141	86302661	3600	100.0	0	0	0
B101	25	20	75	750	987	DemEven	150352	185	92421684	3600	99.8	213992	0	213992
B102	25	20	75	750	895	DemEven	0	95	89646764	3600	100.0	0	0	0
B103	25	30	100	1000	724	DemEven	0	312	121114882	3600	100.0	0	0	0
B104	25	30	100	1000	779	DemEven	0	414	115605436	3600	100.0	0	0	0
B105	25	30	100	1000	761	DemEven	0	366	118987924	3600	100.0	0	0	0
B106	25	30	100	1000	918	DemEven	0	710	100250337	3600	100.0	0	0	0
B107	25	30	100	1000	902	DemEven	0	427	114328741	3600	100.0	0	0	0
B108	25	30	100	1000	885	DemEven	0	441	112508993	3600	100.0	0	0	0
AVG								190						

First, we show the computational time of LP relaxation in Figure 2.14. For each instance size, we show the LPR time for different types of generated instances. All

instance types have similar trend that the LPR time goes up when the instance size increases. Also, when the required capacity ratio R increases from 75% to 90%, the LPR time increases as well for most cases. For type ProdEven, of which each product has the same number of demands statistically, the LPR time is larger than others except the last instance size. Moreover, the objective function value based on the LP relaxation are zero in most cases. This is due to the instance generation have required capacity ratio as 90% maximum, therefore with relaxed setup in LP relaxation, most of the work order can be fulfilled fully on time.

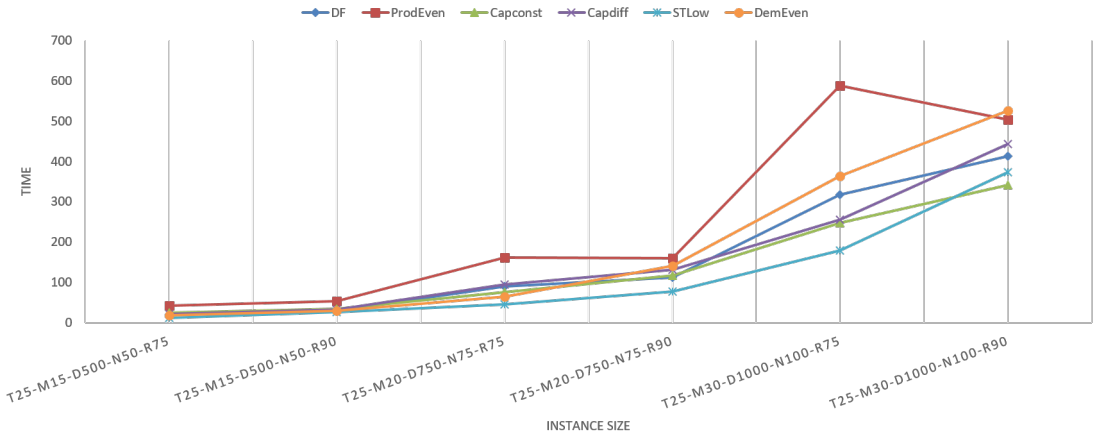


Figure 2.14: LPR time on different types of instances

Second, we show the MIP gap in Figure 2.15. It shows a clear trend that type STLow instances with capacity requirement ratio $R = 75$ are easy to solve. In fact, the type STLow instances, there are 9 instances out of 18 are solved to optimality. This is mainly due to the low setup time and capacity requirement, which makes instances easier to solve since capacity is the bottleneck of our problem. All other type of instances have gap near 100%, this may all because they are hard to solve, but also due to the reason of the poor lower bounds given by LP relaxation. As we mentioned, in most of case, the LP relaxation objective value is 0.

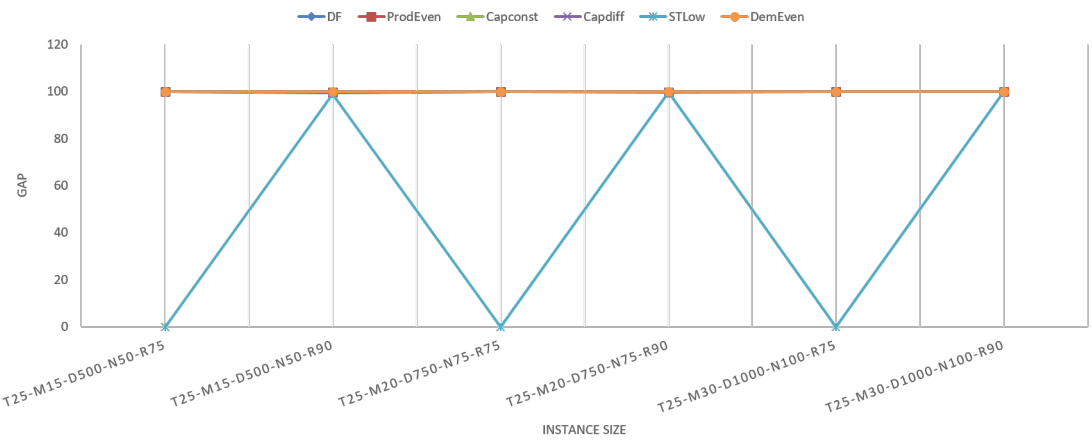


Figure 2.15: MIP gap on different types of instances

To understand better the instance difficulty and the solution quality given by CPLEX, we show the number of nodes explored in Figure 2.16. We can see that only for the instances with size $T25 - M15 - D500 - N50 - R75$, there are nodes explored. For all other instances, CPLEX got stuck at the first heuristic solution and returns a big gap around 99%. It fails to explore more nodes other than the root node.

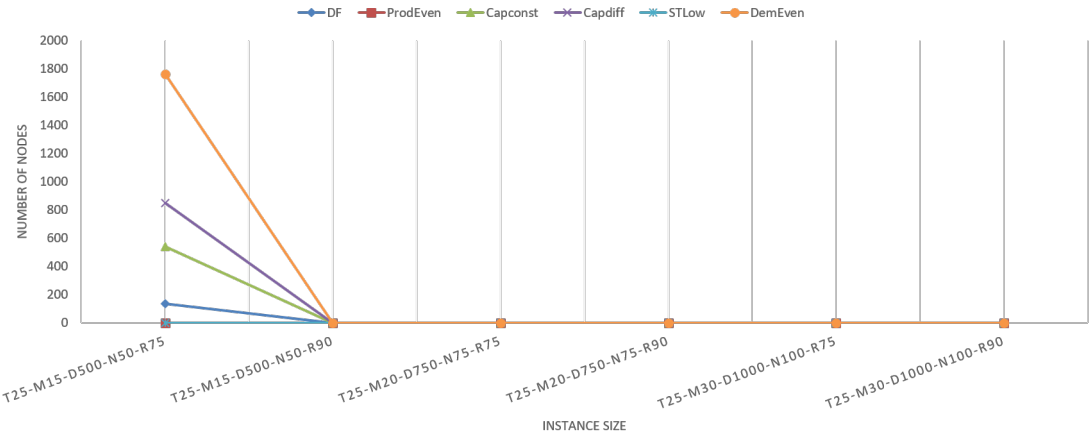


Figure 2.16: MIP number of nodes on different types of instances

Based on this computational result, we conclude that CPLEX has bad performance due to the bad lower bound based on the LP relaxation and the large problem size which prevent its embedded heuristics to find good solution.

2.6 Conclusions

We study a new variant of the CLSP, which is based on a real-world application. The problem combines, for the first time, several classical features of the LSP such as setup carryover and production time windows. We present and compare three different MIP formulations of the problem. We prove that one of the formulations is weaker since it may provide worse LP relaxation bounds. A set of instances are randomly generated and extensive computational experiments are conducted to compare these formulations. The results show that one of the formulation $Form3$ gives the overall best performance on both real-world instances and randomly generated instances. A library of instances is available online, and we hope that this can stimulate further research on this very challenging rich real-world LSP (http://decisionbrain.com/ISCO2016).

Chapter 3

Complex Capacitated Lot Sizing Problem: Heuristics

In the previous chapter, we have introduced the complex capacitated lot sizing problem CLSC. It is shown to be NP-hard and cannot be solved efficiently by a standard MIP solver based on our computational experiments. Therefore, we develop heuristic algorithms to tackle CLSC in this chapter.

The chapter is organized as follows: In Section 3.1, we present the introduction. In Section 3.2, different constructive algorithms are developed, while in Section 3.3 a Fix & Optimize algorithm is presented. Computational results are given in Section 3.4 to evaluate algorithm performances. Finally, we conclude the chapter in Section 3.5.

3.1 Introduction

In the apparel application, the service level key performance indicator has superior priority than operational cost. In other words, tardiness cost and lost sale cost are much more important than setup cost. Our goal is to address the production planning problem from the application. Therefore, we only consider CLSC without setup cost in this chapter.

All heuristic algorithms we have developed are based on the MIP formulation of the problem. As shown before, formulation $Form3$ gives the best overall performance. Therefore, we use $Form3$ to develop and test the algorithm whenever the MIP formulation of CLSC is required.

In the following sections, we present both constructive heuristics and improvement heuristics.

3.2 Constructive Heuristic Algorithms

In this section, we propose three heuristic algorithms to construct feasible solutions to CLSC. They include Fix&Relax algorithm, which is a classical algorithm widely used for CLSP, *product decomposition based algorithm* which explores real instances structure, and *first solution heuristic algorithm* based on the LP relaxation, which has a general framework. We adapt the Fix&Relax algorithm based on the time bucket and the machine decomposition, and it solves a series of MIP models. The PD algorithm is based on the product decomposition, which is also based on solving a series of MIP model. However, its MIP models are on a smaller scale compared to those in the Fix&Relax algorithm. Finally, the first solution heuristic algorithm is based on variables fixation and it solves a series of LP models. Therefore, all constructive heuristics utilize mathematical models but have difference models and scales. For some algorithms, different variations and configurations are developed, and their analysis and comparison results are presented in Section 3.4.

3.2.1 Fix&Relax Algorithm

Fix&Relax (F&R) algorithm [108] is one of the most commonly used heuristic algorithm in practice for CLSP. The algorithm has a general framework and is easy to implement. Furthermore, it gives decent performance in many cases as shown in [3, 72]. Therefore, we adapt F&R algorithm to CLSC in this section.

The essence of F&R algorithm is to decompose a problem, so that in each iteration

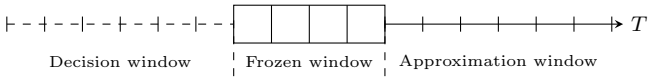
a simpler and smaller sub-problem is solved to construct part of the feasible solution. Normally the problem is divided into three parts:

- Frozen window: integer variables in this set are fixed at a given value.
- Decision window: all variables in this set are defined as in the original problem to be decided.
- Approximation window: integer variables in this set are relaxed to be continuous.

Intuitively, there are three decomposition strategy: period-oriented, product-oriented and machine-oriented. However, due to the setup carryover, the F&R algorithm can not be adapted directly to our problem. Decomposing the problem by products and fix the related setup variables may easily lead to infeasibility. Therefore, we only develop the F&R algorithm based on the time period and the machine decomposition. The product-oriented decomposition method is presented in the next section.

Period-oriented F&R algorithm

The F&R algorithm with period-oriented decomposition (FR-T) decomposes the problem by time buckets. Absi and Kedad-Sidhoum [3] provided a nice presentation on the procedure of FR-T algorithm, we cite it here in Figure 3.1. In kth iteration, we have a time window $[a_k, b_k]$, of which the length is σ. Time buckets $t \in [1, a_k)$ are frozen window, which means all binary variables are fixed at the value of the solution from the previous iteration. Time buckets $t \in [a_k, b_k]$ are decision window, in which all decision variables remain the same as in the original problem. The remaining time buckets $t \in (b_k, T]$ are approximation window that all binary variables are relaxed to continuous variables. We solve this problem and obtain solution sol^k. Note that in sol^k, all binary variables belong to time buckets $[1, b_k]$ have binary values, while the rest can be fractional. Then in the next iteration $k+1$, the decision window is moved forward by a step size δ where $0 < \delta < \sigma$. Therefore we have new decision window $[a_{k+1}, b_{k+1}]$ where $a_{k+1} \leftarrow a_k + \delta$, $b_{k+1} \leftarrow \min\{b_k + \delta, T\}$. Comparing to the sub-problem considered in iteration k, binary variables belonging to $[a_k, a_{k+1} - 1]$ are fixed and variables belonging to $[b_k + 1, b_{k+1}]$ are set back to binary variables. We solve this updated sub-problem in iteration $k+1$ and the process is repeated until the end of the time horizon is reached.



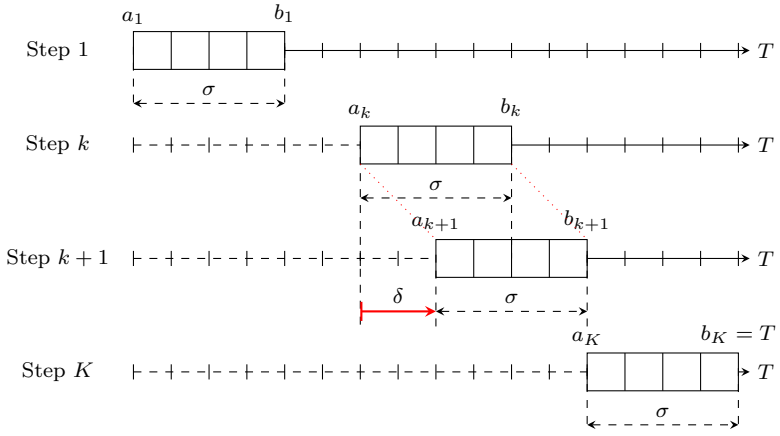


Figure 3.1: FR-T algorithm procedure Absi and Kedad-Sidhoum [3] (modified)

Given time window $[a, b]$ and initial solution sol^0, we define the following sub-problem $P^{FRT}([a, b], sol^0)$:

$$
\begin{aligned}
(P^{FRT}([a,b], sol^0)) \quad \min \quad & (2.15) \\
s.t. \quad & (2.16) - (2.21), (2.23) - (2.25), (2.26) - (2.28) \\
& z^0_{irt} = sol^0(z^0_{irt}),\ z^+_{irt} = sol^0(z^+_{irt}) \quad i \in \mathcal{N}, r \in \mathcal{R}, t \in [1, a-1] \\
& z^0_{irt}, z^+_{irt} \in \{0,1\} \quad i \in \mathcal{N}, r \in \mathcal{R}, t \in [a, b] \\
& z^0_{irt}, z^+_{irt} \in [0,1] \quad i \in \mathcal{N}, r \in \mathcal{R}, t \in [b+1, T]
\end{aligned}
$$

We give the pseudo code of FR-T in Algorithm 1.

Algorithm 1: F&R algorithm with period-oriented decomposition (FR-T)

Input: σ length of decision window, δ step size

Result: x_{irt}, z^0_{irt}, z^+_{irt}, y_{dt} and y_d

$k \leftarrow 1$, $a_k \leftarrow 1$, $b_k \leftarrow \min\{\sigma, T\}$, $sol^0 = \emptyset$;

while $a_k \leq T$ **do**

  $sol^k \leftarrow$ Solve sub-problem $P^{FRT}([a_k, b_k], sol^{k-1})$;

  $a_{k+1} \leftarrow a_k + \delta$, $b_{k+1} \leftarrow \min\{b_k + \delta, T\}$, $k \leftarrow k + 1$;

end

Machine-oriented F&R algorithm

The F&R algorithm with machine-oriented decomposition (FR-M) decomposes the decision by machine. Given two non-intersect subsets of machines $\mathcal{R}^F, \mathcal{R}^O \subseteq \mathcal{R}$ and initial solution sol^0, we define the sub-problem $P^{FRM}(\mathcal{R}^F, \mathcal{R}^O, sol^0)$ as follows:

$(P^{FRM}(\mathcal{R}^F, \mathcal{R}^O, sol^0) \quad \min \quad (2.15)$

$$
\begin{aligned}
s.t. \quad & (2.16)-(2.21),(2.23)-(2.25),(2.26)-(2.28) \\
& z^0_{irt} = sol^0(z^0_{irt}),\ z^+_{irt} = sol^0(z^+_{irt}) \quad i \in \mathcal{N}, t \in \mathcal{T}, r \in \mathcal{R}^F \\
& z^0_{irt}, z^+_{irt} \in \{0,1\} \quad i \in \mathcal{N}, t \in \mathcal{T}, r \in \mathcal{R}^O \\
& z^0_{irt}, z^+_{irt} \in [0,1] \quad i \in \mathcal{N}, t \in \mathcal{T}, r \in \mathcal{R} \backslash \{\mathcal{R}^F \cup \mathcal{R}^O\}
\end{aligned}
$$

We give the pseudo code of FR-M in Algorithm 2.

Algorithm 2: F&R algorithm with machine-oriented decomposition (FR-M)

Input: σ length of the decision window, δ step size, π a permutation of all machines

Result: x_{irt}, z^0_{irt}, z^+_{irt}, y_{dt} and y_d

```
k ← 1, a_k ← 1, b_k ← min{σ, R}, sol^0 = ∅;
while a_k ≤ R do
    R^F ← {π_l ∈ R : 1 ≤ l ≤ a_k − 1};
    R^O ← {π_l ∈ R : a_k ≤ l ≤ b_k};
    sol^k ← Solve sub-problem P^FRM(R^F, R^O, sol^{k−1});
    a_{k+1} ← a_k + σ, b_{k+1} ← min{b_k + σ, R}, k ← k + 1;
end
```

Even though F&R algorithm with product decomposition can not be directly applied to our problem, in the next section, we develop a heuristic algorithm based on the product decomposition.

3.2.2 Product Decomposition Based Algorithm

In this section, we develop a constructive algorithm based on some observation from the application data. Data analysis in Section 2.4.1 shows that even though there are many products, normally 20% of products cover 80% of demands. This feature can actually be observed in many industrial cases. Therefore, we would like to develop an algorithm to make use of this knowledge. One idea is to decompose the problem based on products, and solve a sequence of sub-problems based on each product. We call it the Product Decomposition (PD) based algorithm.

The PD algorithm flow chart is shown in Figure 3.2. First, we sort all products to obtain an order π based on certain criteria. Then for each product $i = \pi(k)$, a sub-problem $P^{PD}(i)$ is built and solved by a standard MIP solver. After each iteration, the current production plan is updated and it moves to the next product $\pi(k+1)$.

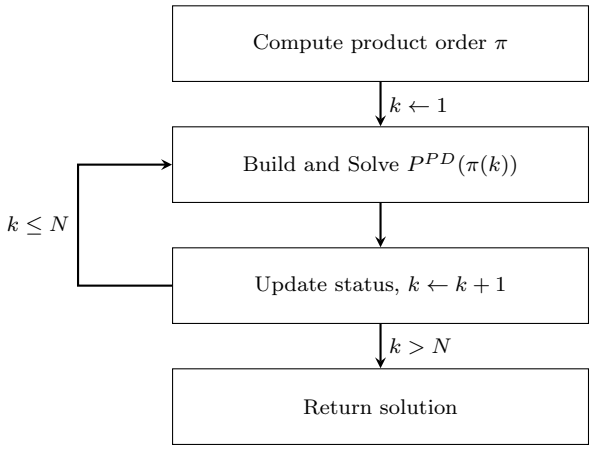


Figure 3.2: PD algorithm flow chart

There are two critical components in PD algorithm: sorting criteria and definition of the sub-problem. In the rest of the section, we introduce several variants of PD based on different strategies of these two components and complete the algorithm.

Sorting criteria

All products can be sorted with increasing or decreasing order based on an assigned value. We introduce two ways to define this value: demand capacity and release date. Therefore based on increasing or decreasing direction, we have four sorting criteria, which are summarized in Table 3.1.

Table 3.1: PD algorithm: sorting criteria

Criteria	Value	Direction
CI	demand capacity $dc(i)$ (defined in (3.1))	Increase ↗
CD	demand capacity $dc(i)$ (defined in (3.1))	Decrease ↘
RI	Release date $rd(i)$ (defined in (3.2))	Increase ↗
RD	Release date $rd(i)$ (defined in (3.2))	Decrease ↘

For a given product i, define the demand capacity as the total required capacity of the demands for this product as follows:

$$dc(i) = \sum_{d \in \mathcal{D}^i} pt_i q_d \tag{3.1}$$

For a given product i, define the release date as the earliest release date of all its demands as follows:

$$rd(i) = \min_{d \in \mathcal{D}^i} r_d \tag{3.2}$$

As shown in Table 3.1, we have four different criteria. For example, the criteria CI represents ordering products according to the increasing value of $dc(i)$. In the Example 2.1, the order of product is $\pi = (i1, i3, i2)$ based on CI with the demand capacity value $dc(i1) = 3$, $dc(i2) = 4$, $dc(i3) = 3$.

Sub-problem definition

For each product, we define a sub-problem, which only plans the production of the current product. On one hand, each sub-problem optimizes the original objective including lost sale cost and tardiness cost for demands belonging to current product. On the other hand, it also has to consider to leave capacity for remaining products to be planned. In the rest of this section, we explain two types of sub-problems, which are summarized in Table 3.2.

Table 3.2: PD algorithm: sub-problem

Sub-problem	StartupCost	IdleCost	FullyUsage
SI	✓	✓	×
F	✓	×	✓

Sub-problem with start-up and idle cost For one product sub-problem, instead of setup carryover, we consider start-up as well as the idle cost. Due to the start-up cost, the solution prefers production in continuous time buckets on a machine. However, if there is only start-up cost, some time buckets will be used partially just to make production continuous and avoid a new start-up. Therefore, we introduce idle cost, which tries to reduce capacity waste for products not planned yet.

Given a product j and available capacity $avlcap(r, t)$ defined for each machine r in time bucket t, we introduce following variables for each machine $r \in \mathcal{R}$ and time bucket $t \in \mathcal{T}$:

- x_{jrt} is defined as in formulation $Form3$ for CLSC, it represents the production quantity of product j on machine r during time bucket t.
- $\gamma_{rt} \in \{0, 1\}$ equals to 1 if machine r in time bucket t is used, i.e., there is a positive production for j.
- $\theta_{rt} \in \{0, 1\}$ equals to 1 if there is a start up in t for machine r, which means that machine r is not used in $t - 1$ ($\gamma_{r,t-1} = 0$) whereas it is used in t ($\gamma_{r,t-1} = 0$).

Let $w_{sp} \geq 0$ and $w_{ic} \geq 0$ be the weight of start-up cost and idle cost respectively. The sub-problem $P_{SI}^{PD}(j)$ with start-up and idle cost is defined as follows:

$$\begin{aligned}(P_{SI}^{PD}(j)) \quad \min \quad & \sum_{d\in\mathcal{D}^j} lc_d y_d + \sum_{d\in\mathcal{D}^j, t\in\mathcal{T}: t\geq e_d^1} tc_d^1 y_{dt} + \sum_{d\in\mathcal{D}^j, t\in\mathcal{T}: t\geq e_d^2} tc_d^2 y_{dt} \\ & + w_{sp} \sum_{r\in\mathcal{R}, t\in\mathcal{T}} \theta_{rt} \\ & + w_{ic} \sum_{r\in\mathcal{R}, t\in\mathcal{T}} (avlcap(r,t)\gamma_{rt} - pt_j x_{jrt} - st_{jr}\theta_{rt}) \qquad (3.3)\end{aligned}$$

$$s.t. \quad \sum_{r\in\mathcal{R}} x_{jrt} = \sum_{d\in\mathcal{D}^j} y_{dt} \qquad t \in \mathcal{T} \qquad (3.4)$$

$$\sum_{t\in\mathcal{T}: t\geq b_d} y_{dt} + y_d = q_d \qquad d \in \mathcal{D}^j \qquad (3.5)$$

$$pt_j x_{jrt} + st_{jr}\theta_{rt} \leq avlcap(r,t)\gamma_{rt} \qquad r \in \mathcal{R}, t \in \mathcal{T} \qquad (3.6)$$

$$x_{jrt} \leq \Theta_{jrt}\gamma_{rt} \qquad r \in \mathcal{R}, t \in \mathcal{T} \qquad (3.7)$$

$$x_{jrt} \geq \gamma_{rt} \qquad r \in \mathcal{R}, t \in \mathcal{T} \qquad (3.8)$$

$$\theta_{rt} \geq \gamma_{rt} - \gamma_{r,t-1} \qquad r \in \mathcal{R}, t \in \mathcal{T} \qquad (3.9)$$

$$\gamma_{rt}, \theta_{rt} \in \{0,1\} \qquad r \in \mathcal{R}, t \in \mathcal{T} \qquad (3.10)$$

$$x_{jrt} \geq 0 \qquad r \in \mathcal{R}, t \in \mathcal{T} \qquad (3.11)$$

$$y_{dt} \geq 0, y_d \geq 0 \qquad d \in \mathcal{D}^j, b_d \leq t \in \mathcal{T} \qquad (3.12)$$

In the objective (3.3), we minimize not only the lost sale cost and tardiness cost, but also the start-up cost and idle cost. As in $Form3$ for CLSC problem, constraints (3.4) ensure the material flow balance in each time bucket while constraints (3.5) state the flow balance for each demand, i.e., the satisfied quantity plus the unsatisfied quantity of each demand should equal to its total required quantity. Constraints (3.6) require total used capacity is no more than the total available capacity on each machine for each time bucket. The total used capacity includes production used capacity $pt_j x_{jrt}$ and start-up used capacity $st_{jr}\theta_{rt}$. On the other hand, the given available capacity is $avlcap(r,t)\gamma_{rt}$, which equals to 0 when the capacity is not used ($\gamma_{rt} = 0$) and equals to $avlcap(r,t)$ when the capacity is used ($\gamma_{rt} = 1$). Constraints (3.7) link production with setup usage that there is a positive production only if the capacity is used. On the other hand, constraints (3.8) ensure at least 1 unit is produced when the capacity is used. Finally, we define start-up variables with constraints (3.9) that given machine r if capacity is not used in $t-1$ but used in t, there is a start-up in time bucket t.

Note that in the objective function, we mix two types of objectives together: original objective (including lost sale cost and tardiness cost) and capacity usage cost (including start-up cost and idle cost). Obviously, they do not have the same priority and serve different purpose. Hence, we need to treat them differently. One way is to adjust the coefficient w_{sp} and w_{ic} accordingly to distinguish different levels of importance (which is actually hard to define in practice). Another way is to use goal programming. Since original objective dominates capacity usage objective, we choose to use goal programming. In the first rank, we try to minimize lost sale cost and tardiness cost, while in the second rank we try to minimize start-up cost and idle cost.

Given a MIP defined as $\min_{x\in\mathcal{X}} f_1(x) + f_2(x)$, the goal programming with objective $f_1(\cdot)$ as rank 1 and $f_2(\cdot)$ as rank 2 is to solve following two optimization problems in sequence:

$$\begin{aligned} &1) \quad z_1^* = \min_{x\in\mathcal{X}} f_1(x) \\ &2) \quad \min_{x\in\mathcal{X}, f^1(x)\leq z_1^*} f_2(x) \end{aligned}$$

Therefore, for each product, we solve two sub-problems $P_{SI1}^{PD}(j)$ and $P_{SI2}^{PD}(j)$ in sequence, which are defined as follows:

$$\begin{aligned} (P_{SI1}^{PD}(j)) \quad \min \quad & \sum_{d\in\mathcal{D}^j} lc_d y_d + \sum_{d\in\mathcal{D}^j, t\in\mathcal{T}: t\geq e_d^1} tc_d^1 y_{dt} + \sum_{d\in\mathcal{D}^j, t\in\mathcal{T}: t\geq e_d^2} tc_d^2 y_{dt} \\ s.t. \quad & (3.4) - (3.12) \end{aligned}$$

Let $obj(P_{SI1}^{PD}(j))$ be the obtained objective function value of problem $P_{SI1}^{PD}(j)$,

$$\begin{aligned} (P_{SI2}^{PD}(j)) \quad \min \quad & w_{sp} \sum_{r\in\mathcal{R}, t\in\mathcal{T}} \theta_{rt} + w_{ic} \sum_{r\in\mathcal{R}, t\in\mathcal{T}} (avlcap(r,t)\gamma_{rt} - pt_j x_{jrt} - st_{jr}\theta_{rt}) \\ s.t. \quad & (3.4) - (3.12) \\ & \sum_{d\in\mathcal{D}^j} lc_d y_d + \sum_{d\in\mathcal{D}^j, t\in\mathcal{T}: t\geq e_d^1} tc_d^1 y_{dt} + \sum_{d\in\mathcal{D}^j, t\in\mathcal{T}: t\geq e_d^2} tc_d^2 y_{dt} \leq obj(P_{SI1}^{PD}(j)) \end{aligned} \tag{3.13}$$

In the first rank, we try to minimize the lost sale cost and tardiness cost in problem $P_{SI1}^{PD}(j)$. Given the obtained feasible objective function value $obj(P_{SI1}^{PD}(j))$, we minimize the start-up cost and idle cost in rank 2. However, constraint (3.13) ensures that the lost sale cost and tardiness cost is not worse than what we have achieved in rank 1.

Sub-problem with fully capacity usage Another way of compact capacity usage is to force almost full capacity usage unless it is a switch-off time bucket. It means that

if several continuous time buckets are used, then the capacity in all time buckets but the last one should be fully used. So on machine r, if time bucket t and $t+1$ are used, then capacity in t must be fully used. In this way, each sub-problem also considers leaving capacity for remaining products to be planned. This sub-problem is defined as follows:

$$
\begin{aligned}
(P_F^{PD}(j)) \quad \min \quad & \sum_{d\in\mathcal{D}^j} lc_d y_d + \sum_{d\in\mathcal{D}^j, t\in\mathcal{T}:t\geq e_d^1} tc_d^1 y_{dt} + \sum_{d\in\mathcal{D}^j, t\in\mathcal{T}:t\geq e_d^2} tc_d^2 y_{dt} \\
& + w_{sp} \sum_{r\in\mathcal{R}, t\in\mathcal{T}} \theta_{rt} \\
s.t. \quad & (3.4)-(3.12) \\
& pt_j x_{jrt} + st_{jr}\theta_{rt} \geq avlcap(r,t)(\gamma_{rt} + \gamma_{r,t+1} - 1) \quad r\in\mathcal{R}, t\in\mathcal{T}\backslash\{T\}
\end{aligned} \tag{3.14}
$$

We also consider to use goal programming, where the first rank is also to minimize the lost sale cost and tardiness cost while the second rank is to minimize the start-up cost.

PD algorithm variants

In this section, we complete two variants of PD algorithm.

First of all, once we solve sub-problems for all products, we use following model to recover the setup values for the original problem. Given solution value γ_{irt} and θ_{irt} for $i\in\mathcal{N}$, $r\in\mathcal{R}$ and $t\in\mathcal{T}$, we define the problem as follows:

$$
\begin{aligned}
(P(\gamma_{irt},\theta_{irt})) \min \quad & 0 \\
s.t. \quad & \sum_{i\in\mathcal{N}} z_{irt}^0 = 1 && r\in\mathcal{R}, t\in\mathcal{T} \\
& z_{irt}^0 \leq z_{ir,t-1}^0 + z_{ir,t-1}^+ && i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T}\backslash\{1\} \\
& z_{irt}^0 + z_{ir,t+1}^0 \leq 1 + w_{rt} && i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T}\backslash\{T\} \\
& z_{irt}^+ + w_{rt} \leq 1 && i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T} \\
& z_{irt}^0 = 1 \quad \textit{If } \gamma_{irt}=1, \theta_{irt}=0 && i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T} \\
& z_{irt}^0 + z_{irt}^+ = 1 \quad \textit{If } \theta_{irt}=1 && i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T} \\
& z_{irt}^+ = 0 \quad \textit{If } \gamma_{irt}=0 && i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T}
\end{aligned}
$$

The PD algorithm with start-up and idle cost (PD-SI) is given in the Algorithm 3. The crucial part is to update the available capacities for the next sub-problem. After planning one product, the available capacity includes non-used time bucket and the remaining capacity in the switch-off time bucket. As shown in the example in Figure

3.3, time buckets tb_1 and tb_4 are never used therefore their capacity is available for the next product. Moreover, the time bucket tb_3 and tb_5 are switch-off time buckets therefore their remaining capacity is also available for the next product. However, time bucket tb_2 is partially used and is not a switch-off time bucket, therefore even though there is capacity left but is considered nonavailable for the next product.

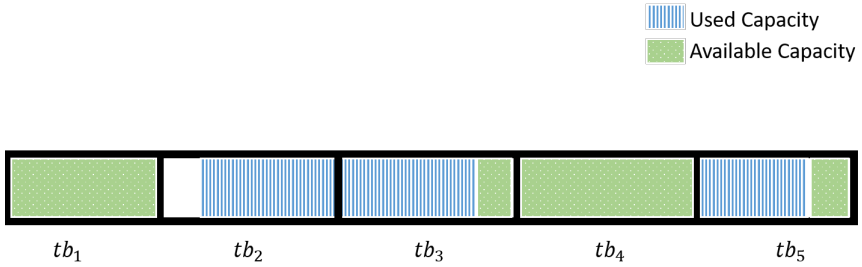


Figure 3.3: PD-SI algorithm capacity update example

The PD algorithm with start-up and fully capacity usage (PD-F) is given in the Algorithm 4.

Algorithm 3: The PD-SI algorithm

Input: Chosen sorting criteria $SortCriteria$

Result: x_{irt}, z^0_{irt}, z^+_{irt}, y_{dt} and y_d

for *each product* i, *each machine* r, *time bucket* t **do**
  $\gamma_{irt} \leftarrow 0$, $\theta_{irt} \leftarrow 0$,;
  $recap(r,t) \leftarrow cap_{rt}$;
  $S(r,t) = \emptyset$;
end
Sort products, $\pi \leftarrow ComputeProductOrder(SortCriteria)$;
for $k = 1$ *to* N **do**
  $j \leftarrow \pi(k)$;
  for *each machine* r, *time bucket* t **do**
    if $S(r,t) \cap S(r,t+1) = \emptyset$ **then**
      $avlcap(r,t) \leftarrow recap(r,t)$;
    end
    else
      $avlcap(r,t) \leftarrow 0$;
    end
  end
  $(x'_{jrt}, y'_{dt}, y'_d, \gamma'_{rt}, \theta'_{rt}) \leftarrow$ Solve sub-problem $P^{PD}_{SI}(j)$ with capacity $avlcap(r,t)$;
  for *each machine* r, *time bucket* t, *demand* $d \in \mathcal{D}^j$ **do**
    $x_{jrt} \leftarrow x'_{jrt}$, $y_{dt} \leftarrow y'_{dt}$, $y_d \leftarrow y'_d$;
    $recap(r,t) \leftarrow recap(r,t) - \left(pt_j x'_{jrt} + st_{jr}\theta'_{rt}\right)$;
    $\gamma_{irt} \leftarrow \gamma'_{rt}$, $\theta_{irt} \leftarrow \theta'_{rt}$,;
    $S(r,t) \leftarrow S(r,t) \cup \{j\}$;
  end
end
$(z^0_{irt}, z^+_{irt}) \leftarrow$ Solve $P(\gamma_{irt}, \theta_{irt})$;

Algorithm 4: The PD-F algorithm

Input: Chosen sorting criteria $SortCriteria$

Result: x_{irt}, z^0_{irt}, z^+_{irt}, y_{dt} and y_d

for *each product* i, *each machine* r, *time bucket* t **do**
  $\gamma_{irt} \leftarrow 0$, $\theta_{irt} \leftarrow 0$,;
  $recap(r,t) \leftarrow cap_{rt}$;
  $S(r,t) = \emptyset$;
end
Sort products, $\pi \leftarrow ComputeProductOrder(SortCriteria)$;
for $k = 1$ *to* n **do**
  $j \leftarrow \pi(k)$;
  for *each machine* r, *time bucket* t **do**
    $avlcap(r,t) \leftarrow recap(r,t)$;
  end
  $(x'_{jrt}, y'_{dt}, y'_d, \gamma'_{rt}, \theta'_{rt}) \leftarrow$ Solve sub-problem $P_F^{PD}(j)$ with capacity $avlcap(r,t)$;
  for *each machine* r, *time bucket* t, *demand* $d \in \mathcal{D}^j$ **do**
    $x_{jrt} \leftarrow x'_{jrt}$, $y_{dt} \leftarrow y'_{dt}$, $y_d \leftarrow y'_d$;
    $recap(r,t) \leftarrow recap(r,t) - \left(pt_j x'_{jrt} + st_{jr}\theta'_{rt}\right)$;
    $\gamma_{irt} \leftarrow \gamma'_{rt}$, $\theta_{irt} \leftarrow \theta'_{rt}$,;
    $S(r,t) \leftarrow S(r,t) \cup \{j\}$;
  end
end
$(z^0_{irt}, z^+_{irt}) \leftarrow$ Solve $P(\gamma_{irt}, \theta_{irt})$;

3.2.3 First Solution Heuristic Algorithm Based on LP Relaxation

In the previous section, we have introduced the PD algorithm, which is based on the special feature that we have observed in the application instances. In this section, we develop a rather general constructive heuristic algorithm which is based on the MIP formulation and its LP relaxation.

The algorithm flow chart is shown in Figure 3.4. First, we solve the LP relaxation of the problem. Based on this continuous solution, a set of binary variables is selected to be fixed. The LP relaxation of the updated formulation is solved again with fixed variables and new binary variables are selected to be fixed. We repeat this process until either there is no more variable that fulfills our condition to be selected, or the LP relaxation is infeasible. In the first case, we solve the resulting MIP formulation

with fixed variables. In the second case, a repair process is performed and we solve the resulted MIP formulation with fixed variables. Eventually, if a solution is returned by solving this final MIP formulation, we obtain a feasible solution to the original problem.

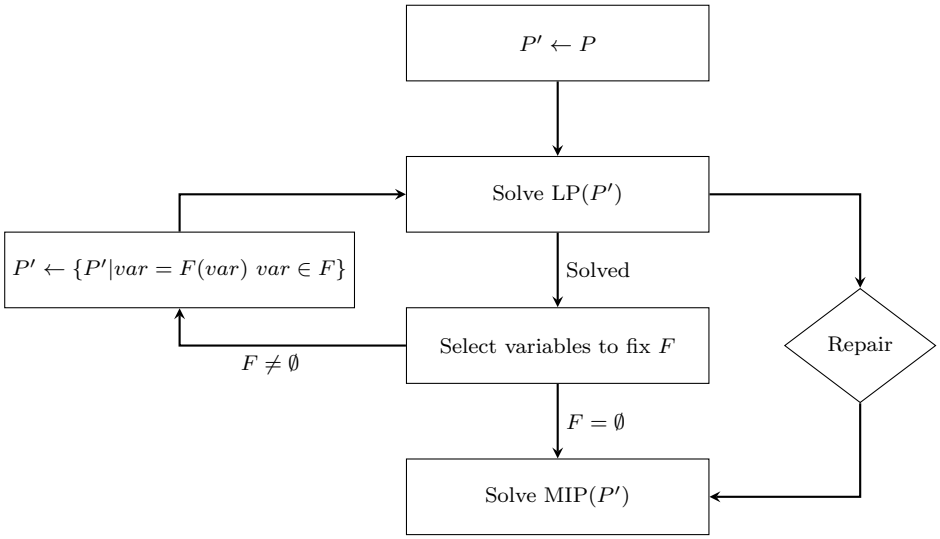


Figure 3.4: FSH algorithm flow chart

There are two issues to consider when we fix variables: first, what variables should be fixed; second, how to avoid infeasibility. We address these two issues in the following.

Variable selection

First we analyze the LP solution of the problem to understand which variables should be fixed. In the Example 2.1, we give the optimal MIP solution without setup cost in Figure 3.5 and the optimal LP relaxation solution in Figure 3.6. In time bucket t_2 on machine r_1, there is not enough capacity to produce and setup for both product $i1$ and $i2$. However, the LP relaxation solution have them both produced by sharing the initial state ($z^0_{i2,r1,t2} = \frac{1}{3}$, $z^0_{i3,r1,t2} = \frac{2}{3}$). Producing both products in $t2$ is a potential conflict and if we help the LP solver to resolve this conflict, i.e., make a decision and set $z^0_{i1,r1,t2} = 1$, then only product $i1$ will be produced in $t2$. Therefore, the idea is that for each machine r, we select a time bucket with highest potential conflict, then a variable is selected to be fixed to resolve this conflict.

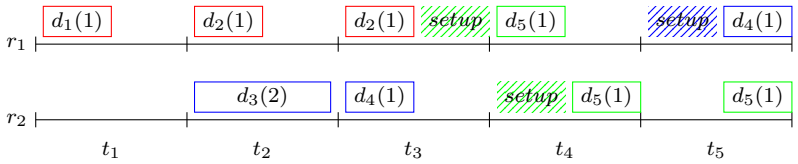


Figure 3.5: CLSC Example 2.1 optimal solution (no setup cost)

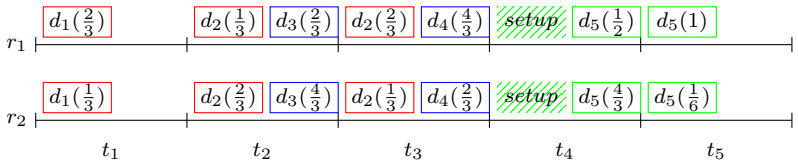


Figure 3.6: CLSC Example 2.1 optimal LP relaxation solution (no setup cost)

The selection strategy is as follows: for each machine r, define

$$vio(r,t) = \begin{cases} \sum_{i\in\mathcal{N}} st_{ir}\lceil z^+_{irt}\rceil - cap_{rt} & \text{if } \exists i \text{ s.t. } z^0_{irt} = 1 \\ \sum_{i\in\mathcal{N}} st_{ir}\lceil z^+_{irt} + z^0_{irt}\rceil - cap_{rt} & \text{otherwise} \end{cases} \tag{3.15}$$

It represents the potential capacity violation when we round up fractional setup variable values to integers. When $vio(r,t) \leq 0$, it implies that even if we setup all the products indicated in the LP relaxation solution, there is enough capacity in (r,t). When $vio(r,t) > 0$, it means that the LP relaxation solution might be making a wrong decision that there is not enough capacity to produce all the products indicated in the continuous solution.

To simplify the algorithm description, we introduce following notations which has the same structure as Map in Java.

Definition 3.1. *Let $\mathcal{B}$ represent the set of all binary variables in a MIP problem. Given a set $F \subseteq \mathcal{B} \times \{0,1\}$, which is a set of variable and binary value pair, we introduce the notation*

- *Denote $var \in F_v$ if there exists $val \in \{0,1\}$ such that $(var, val) \in F$.*
- *Given $var \in \mathcal{B}$, define*

$$F(var) := \begin{cases} val \text{ such that } (var,val) \in F & \text{if } var \in F_v \\ -1 & \text{otherwise} \end{cases}$$

Given a continuous solution of CLSC, the variable selection in FSH algorithm is given as follows in Algorithm 5:

Algorithm 5: The variable selection algorithm of FSH algorithm

Input: sol^{LPR} as a continuous solution, a set of variables already fixed $F^0 \subseteq \mathcal{B} \times \{0,1\}$;

Result: A subset of binary variables $F \subseteq \mathcal{B} \times \{0,1\}$

$F \leftarrow \emptyset$;
for *each* $r \in \mathcal{R}$ **do** /* Select one variable for each r */
  for *each* $t \in \mathcal{T}$ **do**
    Compute $vio(r,t)$ as (3.15) based on sol^{LPR};
  end
  if $\max_t vio(r,t) > 0$ **then** /* If exists conflict */
    Let t^* such that $vio(r,t^*) = \max_{t\in\mathcal{T}} vio(r,t)$; /* t with max conflict */
    Let $\mathcal{N}^* \subseteq \mathcal{N}$ be the subset of products such that $z^0_{irt} \notin F^0$, $z^+_{irt} \notin F^0$;
    Let $i^* \in \mathcal{N}^*$ such that $pt_{i^*} sol^{LPR}(x_{i^*,rt}) = \max_{i\in\mathcal{N}^*} pt_i sol^{LPR}(x_{irt})$;
    ; /* i with max production capacity */
    if $z^0_{i^*,rt} \notin F^0$ **then**
      $F \leftarrow F \cup \{(z^0_{i^*,rt}, 1)\}$;
      $F \leftarrow F \cup$ UpdateBounds$(z^0_{i^*,rt}, 1)$;
    end
    else
      $F \leftarrow F \cup \{(z^+_{i^*,rt}, 1)\}$;
      $F \leftarrow F \cup$ UpdateBounds$(z^+_{i^*,rt}, 1)$;
    end
  end
end

"UpdateBounds(var,val)" is a function to propagate bounds of other variables after fixing var at val according to the rules introduced in the following section. It returns a set of variables and their fixed values.

Infeasibility issue

If the fixing strategy is proper, we may even prevent the situation of infeasible LP relaxation or MIP, therefore guarantees to return a feasible solution. Due to the nature of our problem, the major infeasiblity comes from the relations between setup variables z^0_{irt} and z^+_{irt}. Therefore, one way is to introduce analysis rules and perform constraint propagation. All the propagation rules are summarized as follows:

- If set $z^0_{irt} = 0$
  - If $z^+_{irt} = 0$, set $z^0_{ir,t+1} = 0$
  - If $z^0_{ir,t+1} = 1$, set $z^+_{irt} = 1$
- If set $z^0_{irt} = 1$
  - For $i \neq j \in \mathcal{N}$, set $z^0_{jrt} = 0$
  - Set $z^+_{irt} = 0$
  - If $z^0_{ir,t+1} = 1$, for $i \neq j \in \mathcal{N}$, set $z^+_{jrt} = 0$
  - If there exits j, $i \neq j \in \mathcal{N}$, set $z^0_{ir,t+1} = 0$
  - If $z^0_{ir,t-1} = 1$, for $j \in \mathcal{N}$, set $z^+_{jr,t-1} = 0$
  - If $z^0_{ir,t-1} = 0$, set $z^+_{ir,t-1} = 1$
  - If $z^+_{ir,t-1} = 0$, set $z^0_{ir,t-1} = 1$
  - If there exits $j \in \mathcal{N}$ such that $z^+_{jr,t-1} = 1$, set $z^0_{ir,t-1} = 0$
  - If $z^0_{ir,t-1}$ and $z^+_{ir,t-1}$ are both free, For $i \neq j \in \mathcal{N}$, if $recap(r, t-1) < st_{jr}$, set $z^+_{jr,t-1} = 0$
- If set $z^+_{irt} = 0$
  - If $z^0_{irt} = 0$, set $z^0_{ir,t+1} = 0$
  - If $z^0_{ir,t+1} = 1$, set $z^0_{irt} = 1$
- If set $z^+_{irt} = 1$
  - For $i \neq j \in \mathcal{N}$, if z^+_{jrt} is free and $recap(r, t) < st_{jr}$, set $z^+_{jrt} = 0$
  - Set $z^0_{irt} = 0$
  - For $j \in \mathcal{N}$, if $z^0_{jrt} = 1$, set $z^0_{jr,t+1} = 0$
  - For $j \in \mathcal{N}$, if $z^0_{jr,t+1} = 1$, set $z^0_{jrt} = 0$
  - For $j \in \mathcal{N}$, if $z^+_{jrt} = 0$, set $z^0_{jr,t+1} = 0$

where $recap(r, t) = cap_{rt} - \sum_{i \in \mathcal{N}, z^+_{irt} \text{ is fixed to } 1} st_{ir}$.

Unfortunately, we can not guarantee that the LP relaxation and the final MIP are feasible based on the above rules. One future research direction is to develop a general constraint propagator which can detect infeasibility and we can have roll back function to recover from the infeasible fixation.

Repair Procedure

If infeasibility is detected at the kth iteration during running FSH, we try to repair by rolling back to the previous iteration. Let $F(k)$ be the set of all variables fixed after the kth iteration and F_k be the set of all variables fixed during the kth iteration. We recover all variables $var \in F_k$ back to free binary variables and solve the original MIP model with variables $var \in F(k) \backslash F^k$ fixed to the given value during FSH.

First solution heuristic algorithm (FSH)

The pseudo code of the FSH algorithm is given in Algorithm 6.

Algorithm 6: FSH algorithm

Input: P as the original problem. ;

Result: *sol*

Initialize: $P' \leftarrow P$, $F^{all} \leftarrow \emptyset$, $F^{pre} \leftarrow \emptyset$;
while do
  $sol^{LPR} \leftarrow$Solve the LP Relaxation of P';
  if $sol^{LPR} == NULL$ **then** `/* LPR infeasible */`
    $F^{all} \leftarrow F^{all} \backslash F^{pre}$;
    $P' \leftarrow \{P' : var = val \ \ \forall (var, val) \in F^{all}\}$;
    Break;
  end
  $F^{cur} \leftarrow$ SelectVariablesToFix(sol^{LPR}, F^{all});
  if $F^{cur} == \emptyset$ **then** `/* No variables to fix */`
    Break;
  end
  else
    $F^{Fea} \leftarrow$ Recursively set var at val for $(var, val) \in F^{cur}$ based on feasibility rules;
    $F^{cur} \leftarrow F^{cur} \cup F^{Fea}$;
  end
  $F^{pre} \leftarrow F^{cur}$, $F^{all} \leftarrow F^{all} \cup F^{cur}$;
  $P' \leftarrow \{P' : var = val \ \ \forall (var, val) \in F^{all}\}$;
end
$sol \leftarrow$ Solve P' as MIP;

3.3 Fix&Optimize algorithm

In previous sections, we introduce several constructive heuristic algorithms to build up a solution for CLSC from scratch. In this section, we develop a local search algorithm to further improve the solution quality. Fix&Optimize (F&O) algorithm [108] is another commonly used method for LSP. The algorithm starts from an initial solution. Then in each iteration, partial variables are fixed, while the remaining variables are optimized to try to improve the solution quality. After each iteration, variables in the decision window are updated and the process is repeated until certain criteria is reached. The final solution is no worse than the initial solution. The idea is to solve a smaller MIP problem in each iteration to search for a better solution.

In this section, we develop a F&O algorithm based on period-oriented decomposition. Given a feasible solution of CLSC as sol^0, we define the problem $P^{FO}([a_k, b_k], sol^0)$ as follows:

$$
\begin{aligned}
(P^{FO}([a,b], sol^0)) \quad \min \quad & (2.15) \\
s.t. \quad & (2.16)-(2.21), (2.23)-(2.25), (2.26)-(2.28) \\
& z^0_{irt} = sol^0(z^0_{irt}),\ z^+_{irt} = sol^0(z^+_{irt}) \\
& \quad i \in \mathcal{N}, r \in \mathcal{R}, t \in [1, a-1] \cup [b+1, T] \\
& z^0_{irt}, z^+_{irt} \in \{0,1\} \quad i \in \mathcal{N}, r \in \mathcal{R}, t \in [a,b]
\end{aligned}
$$

The algorithm is formally presented in Algorithm 7:

Algorithm 7: The Fix&Optimize (F&O) algorithm for CLSC

Input: σ length of the decision time window, δ step size

Result: x_{irt}, z^0_{irt}, z^+_{irt}, y_{dt} and y_d

1 $k \leftarrow 1$, $a_k \leftarrow 1$, $b_k \leftarrow \min\{\sigma, T\}$, $sol^0 = \emptyset$;
2 **while** $a_k \leq T$ **do**
3 $\quad sol^k \leftarrow$ Solve sub-problem $P^{FO}([a_k, b_k], sol^{k-1})$;
4 $\quad a_{k+1} \leftarrow a_k + \delta$, $b_{k+1} \leftarrow \min\{b_k + \delta, T\}$, $k \leftarrow k+1$;
5 **end**

3.4 Computational Results

In this section, we present computational results of above developed heuristic algorithms to evaluate and compare their performance. First, parameter evaluation is performed on the pilot benchmark instances to evaluate different configurations of the F&R algorithm

and the PD algorithm. Then all developed heuristic algorithms are compared based on their chosen configurations which give the overall best performance.

3.4.1 Algorithm Parameter Evaluation

We use benchmark IAP-B, which includes four real-world instances, as pilot benchmark instances. Computational results of pilot instances with CPLEX given 1 hour time limit is used as a reference, we recall the computational results here, which have been presented in Table 2.11 in Section 2.5.4:

Table 2.11: Computational results: CPLEX on IAP-B

	Characteristics					LP		MIP					
Inst	T	R	N	D	Γ	Obj	Time	Obj	Time	Gap	LB	#Node	BestLB
R5	25	30	46	668	91	2,935,797	79	35,511,200	3600	91.6	2,973,702	0	2,973,702
R6	25	30	36	425	74	1,277,107	18	1,456,011	3600	7.7	1,344,501	97	1,344,501
R7	20	31	80	1428	40	2,217,260	118	2,692,957	3601	16.7	2,244,422	29	2,244,422
R8	20	31	73	1404	41	2,081,921	83	2,597,838	3600	18.7	2,111,181	0	2,111,181
AVG						2,128,021	74	10,564,501	3600	33.7	2,168,451	32	2,168,451

Parameter evaluation of Fix&Relax (F&R) algorithm

Parameters for the F&R algorithm include decomposition strategy, decision window size σ and step size δ. We test following combinations of different parameters:

- Decomposition strategy: TO, MO
- (σ, δ): {(2, 1), (3, 1), (3, 2)}

The computational results are shown in Table 3.3. The time limit of each iteration is set to 600 seconds.

Table 3.3: Comparison results: F&R algorithm variations

	FR-TO-(2,1)			FR-TO-(3,1)			FR-TO-(3,2)		
Instance	Obj	Time	Gap	Obj	Time	Gap	Obj	Time	Gap
R5	-	-	-	-	-	-	-	-	-
R6	1,413,088	5387	4.9	1,391,377	5854	3.4	1,422,186	3181	5.5
R7	4,322,763	5553	48.1	3,458,993	5195	35.1	4,274,910	3080	47.5
R8	-	-	-	3,292,260	5465	35.9	3,935,323	3036	46.4
AVG	2,867,925	5470	26.5	2,714,210	5505	24.8	3,210,806	3099	33.1
	FR-MO-(2,1)			FR-MO-(3,1)			FR-MO-(3,2)		
Instance	Obj	Time	Gap	Obj	Time	Gap	Obj	Time	Gap
R5	-	-	-	-	-	-	-	-	-
R6	1,442,343	4111	6.8	1,430,022	6292	6.0	1,423,001	2260	5.5
R7	2,448,047	14661	8.3	2,432,500	16572	7.7	2,468,958	8371	9.1
R8	2,309,079	10990	8.6	2,266,813	12583	6.9	2,363,615	6718	10.7
AVG	2,066,490	9921	7.9	2,043,112	11816	6.9	2,085,191	5783	8.4

According to the computational results, we have following observations:

1) For all tested configurations, the F&R algorithm fails to solve the instance R5. The reason is that it fails to find a feasible solution in the first iteration within the time limit set on each iteration (600 seconds). We have also tried to increase this time limit from 600 seconds to 1200 seconds, we observe the same result for instance R5 that the F&R algorithm still fails to find a feasible solution to the sub-problem in the first iteration. This implies that the sub-problems constructed in F&R is still too hard to solve for CPLEX based on tested instances and formulation.

2) We observe the overall best solution quality on the F&R algorithm with machine-oriented decomposition, decision window length as 3 and step size as 1 (FR-MO-(3,1)). It returns an average gap of 6.9%, whereas it is 33.7% for CPLEX. However, the average computational time of FR-MO-(3,1) is more than 3 hours. Therefore, we observe a natural trade-off between solution qualities and computational times.

3) Comparing the two decomposition strategies of the F&R algorithm, MO based F&R algorithm offers better solution quality than TO based F&R algorithm, but with a much longer computational time. Also, one does not dominate the other since we observe that the algorithm FR-TO-(2,1) obtains better solution than its counter part FR-MO-(2,1).

4) Comparing the different decision window sizes and step sizes, the parameter $(3, 1)$ gives best solution quality on average for both decomposition strategies. In fact, it is true for all cases except one that the FR-MO-(3,2) algorithm gives better solution than

FR-MO-(3,1) for instance R6.

In summary, F&R algorithm based on machine decomposition with decision window size and step size as (3,1) gives the overall best performance. However, the F&R algorithm can not address the most difficult instance R5. Also, the computational time of F&R is on average very long, which exceeds 3 hours for some configurations. Hence, we conclude that F&R algorithm is not efficient enough on our problem, therefore we stop testing it on other benchmark instances.

Parameter evaluation of PD algorithm

The parameter for the PD algorithm includes sorting criteria and sub-problem definition.

- Sorting criteria: the demand capacity $dc(\cdot)$ and the release date $rd(\cdot)$ in increasing and decreasing order.
- Sub-problem definition: the sub-problem with the start-up and the idle cost (SI), the sub-problem with full usage (F).

We first evaluate two sub-problems based on the same sorting criteria: the demand capacity $dc(\cdot)$ in decreasing order. The computational results are presented in Table 3.4.

Table 3.4: Comparison results of PD algorithm variations: sub-problems

	PD-SI-DC			PD-F-DC		
Instance	Obj	Time	Gap	Obj	Time	Gap
R5	5,740,173	70	48.2	5,738,345	74	48.2
R6	2,574,221	8	47.8	2,450,901	7	45.1
R7	3,713,166	154	39.6	3,800,412	315	40.9
R8	4,211,277	180	49.9	4,159,586	285	49.2
AVG	4,059,709	103	46.3	4,037,311	170	45.9

Based on the same sorting criteria of decreasing required capacity, the PD algorithm based on the SI sub-problem (PD-SI-DC) and the PD algorithm based on the F sub-problem (PD-F-DC) obtain solutions with similar quality. Over tested instances, the average gap of PD-SI-DC and PD-F-DC are 46.3% and 45.9%, which implies that the PD-F-DC algorithm have slightly better performance on average. However, there is no dominance between these two variants. Moreover, we observe a longer average computational time on the PD-F-DC (170 seconds) algorithm than PD-SI-DC (103 seconds). This is due to the fact that the almost-full-capacity-usage constraints introduces more difficulty to the sub-problems.

We then evaluate the sorting criteria based on the same sub-problem definition SI. The computational result is shown in Table 3.5.

Table 3.5: Comparison results of PD algorithm variations: sorting criteria

	PD-SI-DC			PD-SI-IC			PD-SI-DR			PD-SI-IR		
Instance	Obj	Time	Gap	Obj	Time	Gap	Obj	Time	Gap	Obj	Time	Gap
R5	5,740,173	70	48.2	56,347,969	117	94.7	66,499,812	87	95.5	31,158,680	146	90.5
R6	2,574,221	8	47.8	3,576,452	19	62.4	3,573,873	16	62.4	2,539,376	72	47.1
R7	3,713,166	154	39.6	4,083,438	344	45.0	3,986,194	365	43.7	3,693,703	325	39.2
R8	4,211,277	180	49.9	4,159,176	271	49.2	3,982,998	271	47.0	3,886,077	368	45.7
AVG	4,059,709	103	46.3	17,041,759	188	62.9	19,510,719	185	62.1	10,319,459	228	55.6

We observe that the variant with decreasing demand capacity (PD-SI-DC) gives the overall best performance, since it has the lowest average gap and the shortest average computational time. Its average gap is 46.3%, whereas the average gap is more than 55% for other criteria. The major improvement comes from the instance R5. Compared to CPLEX with 1 hour time limit, within 70 seconds, the PD-SI-DC algorithm reduces the gap from 91% to 48.2%.

We also present Figure 3.7 to show the comparison of computational time and relative gap between PD variants. In summary, PD algorithm is computationally efficient and tackles the most difficult instance R5. Also, the algorithm PD-F-DC gives the overall best performance.

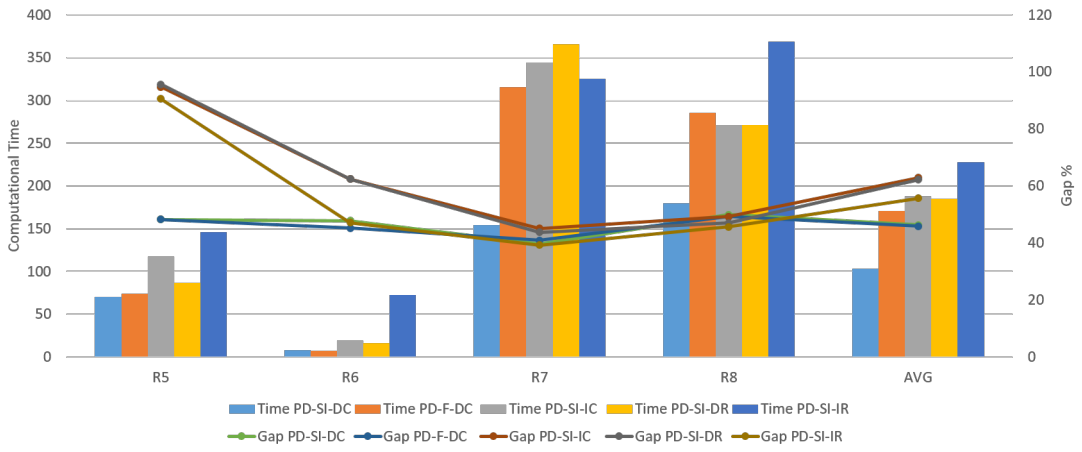


Figure 3.7: PD algorithm gap and computational time on pilot benchmark

3.4.2 Algorithm Comparison Results

In previous sections, we have presented the parameter analysis on the F&R algorithm and PD algorithm. In this section, we select following parameters which give the best overall performance for each algorithm, and perform test on benchmark IAP-B and IRG-B to evaluate their performances.

- CPLEX with 1 hour time limit (presented in Section 2.5.4).
- PD-F-DC algorithm, iteration time limit for each sub-problem as 60 seconds.
- FSH algorithm, iteration time limit as 600 seconds.
- F&O algorithm with optimization window size as 3 and step size as 1, iteration time limit as 60 seconds.

FSH results

We first present the detailed computational results of the FSH algorithm. In Table 3.6 we show the computational results on the benchmark IAP-B while in Table 3.7-3.9 we show the FSH results on benchmark IRG-B.

As we presented before, the FSH algorithm consists of two steps: *LP phase*, when a series LP relaxation problems are solved and a subset of binary variables are fixed during each iteration. *MIP phase*, the restricted MIP model is solved to obtain a feasible solution. Therefore, we present detailed results to analyze that how each component performs and contributes to the algorithm.

In the following table, for each instance, we present the obtained objective function value (Obj), total computational time (Time), the relative gap based on the best known lower bound (Gap), the number of iteration of solving LP relaxation and fixing variables (#Iter), the percentage of the number of variables fixed to 1 over the number of all binary variables (#FixedTo1), and the computational time to solve the restricted MIP problem.

Table 3.6: Computational results: FSH algorithm on IAP-B

Inst	Obj	Time	Gap	#Iter	#FixedTo1(%)	MIPTime
R5	14,160,409	216	79.0	19	1.3	3
R6	5,761,670	64	76.7	17	1.4	2
R7	8,835,804	376	74.6	21	1.5	5
R8	6,549,786	316	67.8	24	1.6	3
AVG	8,826,917	243	74.5	20	1.5	3

Based on results of real application instances IAP-B, we observe that
1) The FSH algorithm addresses the difficult instance R5. Compared to CPLEX, the objective function value is reduced from 35,511,200 to 14,160,409, while the gap is reduced from 91.6% to 79.0%. Moreover, its computational time is within 4 minutes. However, this does not apply to all instances. For relatively easier to solve instances R6, R7 and R8, the solution quality is worse than that of CPLEX.
2) On average, the average computational time is around 4 minutes to construct feasible solutions. In FSH algorithm, the most time consuming part is to solve the LP relaxation sub-problem in each iteration. Therefore, we can expect that as the problem size increases, the FSH algorithm will consume more computational time as well.
3) For the LP phase, the FSH algorithm goes through 20 iterations on average and fixes near 1.5% of binary variables to 1. However, it is enough to solve the resulted MIP in less than 5 seconds. This is mainly due to the structure of the setup carryover that if we fix all initial setup states, then all other z^0_{irt} variables are fixed to 0 accordingly. To build up the first solution, the major computational time is spent on solving LP relaxation problems. Therefore the bottleneck of the problem speed is at solving LP relaxation problems.

Table 3.7: Computational results: FSH algorithm on IRG-B (1)

Inst	Obj	Time	Gap	#Iter	#FixedTo1	MIPTime
B1	569496	261	100.0	28	1.5	2
B2	857700	252	100.0	28	1.4	2
B3	1332316	238	100.0	31	1.5	2
B4	3154699	310	95.4	31	0.9	4
B5	1627483	294	100.0	29	1.3	2
B6	2687665	315	100.0	27	1.2	3
B7	3006613	947	100.0	30	1.0	6
B8	2675949	893	100.0	32	1.1	5
B9	1925285	1040	100.0	31	0.9	6
B10	3301737	1081	100.0	30	0.8	7
B11	4475709	1242	100.0	27	0.8	7
B12	3183505	1111	100.0	27	0.8	7
B13	1421072	3480	100.0	29	0.7	20
B14	1477498	3505	100.0	31	0.8	23
B15	1004987	3530	100.0	29	0.7	22
B16	120095966	1216	100.0	0	0.0	600
B17	7721458	3932	99.8	29	0.5	25
B18	4892235	3842	99.8	30	0.5	26
AVG		1527	100	28	0.9	43
B19	1885162	328	100.0	30	1.7	2
B20	2435489	379	100.0	28	1.5	3
B21	3068139	389	100.0	28	1.5	3
B22	5863879	374	98.9	28	1.3	3
B23	6120530	372	100.0	33	1.2	3
B24	4924154	410	100.0	28	1.3	3
B25	5299304	1531	100.0	28	1.0	9
B26	4432748	1492	100.0	32	1.0	7
B27	4074232	1349	100.0	31	1.0	7
B28	10067139	1371	99.8	29	0.8	7
B29	11750265	1367	99.0	30	0.8	6
B30	10401358	1383	99.6	29	0.8	6
B31	4458564	5328	100.0	31	0.7	26
B32	4216738	4989	100.0	30	0.8	23
B33	119181317	1216	100.0	0	0.0	601
B34	13215539	4146	99.8	32	0.6	17
B35	12392793	3701	98.4	28	0.6	21
B36	114132867	1216	100.0	0	0.0	601
AVG		1741	100	26	0.9	75

Table 3.8: Computational results: FSH algorithm on IRG-B (2)

Inst	Obj	Time	Gap	#Iter	#FixedTo1	MIPTime
B37	983010	239	100.0	30	1.6	3
B38	1412074	281	100.0	31	1.4	2
B39	1255719	273	100.0	30	1.4	2
B40	3909024	312	100.0	32	1.2	2
B41	1734686	296	100.0	27	1.2	3
B42	3277156	354	99.1	29	1.1	2
B43	2011864	930	100.0	30	1.0	7
B44	1114294	782	100.0	31	1.1	8
B45	2145942	832	100.0	30	1.0	6
B46	5709567	1127	99.9	32	0.8	7
B47	3407334	1101	100.0	30	0.8	7
B48	6654652	1327	99.0	29	0.7	6
B49	1790740	3516	100.0	31	0.7	22
B50	2460050	3310	100.0	30	0.8	20
B51	2398138	3833	100.0	30	0.7	21
B52	4685339	4049	100.0	31	0.6	26
B53	5921797	3660	98.9	30	0.5	24
B54	6485383	3934	98.7	28	0.5	24
AVG		1675	100	30	1.0	11
B55	909994	259	100.0	30	1.4	2
B56	816435	247	100.0	31	1.5	2
B57	736264	243	100.0	30	1.5	2
B58	2247744	331	100.0	27	1.1	3
B59	2298256	310	100.0	28	1.1	3
B60	2278970	322	99.1	26	1.1	3
B61	1842306	934	100.0	30	1.0	5
B62	2518664	1022	100.0	31	1.0	7
B63	2300055	1088	100.0	29	1.0	7
B64	5192475	1142	100.0	29	0.7	9
B65	5255345	1204	100.0	29	0.8	8
B66	4610617	1031	99.9	27	0.7	8
B67	1937528	3542	100.0	29	0.7	22
B68	1845334	3469	100.0	29	0.7	27
B69	2948951	3554	100.0	33	0.7	21
B70	4781314	4690	100.0	30	0.6	31
B71	115751239	1216	100.0	0	0.0	600
B72	5652817	3296	100.0	28	0.5	27
AVG		1550	100	28	0.9	44

Table 3.9: Computational results: FSH algorithm on IRG-B (3)

Inst	Obj	Time	Gap	#Iter	#FixedTo1	MIPTime
B73	1027	66	100.0	44	2.2	4
B74	2166	71	100.0	42	2.3	3
B75	5079	113	100.0	44	2.2	4
B76	118275	276	100.0	37	1.6	2
B77	1290034	341	93.2	30	1.2	3
B78	626638	276	78.8	30	1.3	2
B79	11036	135	100.0	45	1.5	9
B80	19799	544	100.0	39	1.4	11
B81	3116	446	100.0	41	1.4	10
B82	495908	952	100.0	33	1.1	6
B83	1431220	1028	89.8	29	0.8	7
B84	110817	910	100.0	35	1.1	6
B85	4838	2013	100.0	43	1.1	26
B86	5088	680	100.0	42	1.1	22
B87	2812	2249	100.0	47	1.1	25
B88	13799	3327	100.0	37	0.8	21
B89	207862	4165	100.0	49	0.8	20
B90	78039	3991	100.0	38	0.7	25
AVG		1199	98	39	1.3	11
B91	870570	213	100.0	29	1.5	2
B92	460278	233	100.0	32	1.6	2
B93	776054	236	100.0	29	1.5	2
B94	1624274	310	100.0	28	1.2	3
B95	2211051	319	100.0	26	1.2	3
B96	2455140	338	100.0	26	1.1	3
B97	1625407	803	100.0	30	1.1	6
B98	2150486	966	100.0	31	1.0	4
B99	1720532	936	100.0	30	1.0	6
B100	6485075	1142	100.0	30	0.7	7
B101	9240132	1262	97.7	27	0.7	7
B102	3928379	1127	100.0	28	0.8	8
B103	1436398	3410	100.0	31	0.8	18
B104	2097677	3616	100.0	30	0.7	19
B105	1882524	3606	100.0	30	0.7	19
B106	123496124	1215	100.0	0	0.0	600
B107	4693908	4228	100.0	30	0.6	23
B108	3453669	4800	100.0	29	0.6	25
AVG		1598	100	28	0.9	42

For the test on benchmark IRG-B, first we show the computational time of FSH algorithm in Figure 3.8. The speed of the algorithm depends on the LP relaxation, therefore the curve has similar form as that of the computational time of the LP relaxation presented in Figure 2.14.

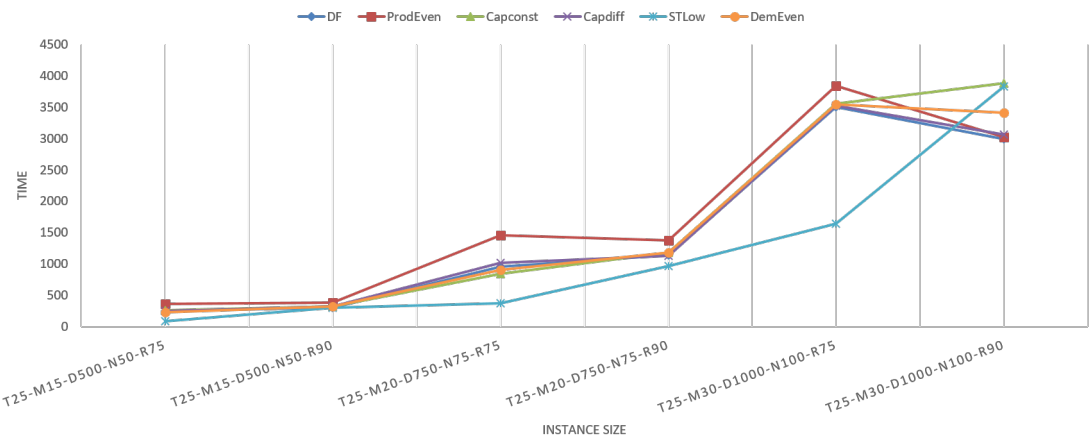


Figure 3.8: FSH algorithm computational time on different types of instances

Out of 108 instances, there are 5 instances that the FSH algorithm do not fix any variables due to the failure of solving the LP relaxation problem in the first iteration given 600 seconds time limit. Therefore, the returned results of the FSH algorithm is to solve the original model given 600 seconds time limit. For other instances, the average iterations for fixing variables are from 26 to 39 for different types of instances with around 1% of binary variables fixed to 1. This is similar to the results of the IAP-B instances. Moreover, the resulted restricted MIP model are solved within 1 minutes. Also, for tested instances, the constraint propagation manages to avoid infeasibility during the variable fixation. Without the constraint propagation, the FSH algorithm easily get into infeasibility which leads to no solution found in most cases. However, as we mentioned before, the completeness of the propagation rule is not proved. At last, for the solution quality, the relative gap is still very high based on our best known lower bounds. The further analysis are given in comparison with other heuristic algorithms in the next part.

All heuristic algorithm comparison results

Finally, we compare all developed heuristic algorithms in Table 3.10 - 3.15. The detailed results are given in Appendix Table A.6 - A.11. For CPLEX results, we present the computational time, the objective function value and the relative gap based on the best known lower bound as reference. For other heuristic algorithms, we show the computational time and the relative gap or the relative improvement defined as $impro = \max\{-1, \frac{obj^C - obj^A}{\max\{1, obj^C\}}\} \cdot 100\%$, where obj^A and obj^C are the objective function value of the corresponding algorithm and CPLEX. We compare the improvement instead of gap is because that in many cases of IRG-B, the best known lower bound is 0, which leads

to 100% gap for any positive objective function value. Therefore we use the value *impro* to measure the improvement of objective function value with reference of CPLEX.

Table 3.10: Computational results: heuristic algorithm on IAP-B

	CPLEX			FSH		PD		FSH+FO		PD+FO	
Inst	Obj	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap
R5	35,511,200	3600	91.6	216	79.0	74	48.2	662	14.2	478	35.8
R6	1,456,011	3600	7.7	64	76.7	7	45.1	492	5.9	181	19.0
R7	2,692,957	3601	16.7	376	74.6	315	40.9	823	9.2	756	12.0
R8	2,597,838	3600	18.7	316	67.8	285	49.2	762	6.9	734	17.7
AVG	10,564,501	3600	33.7	243	74.5	170	45.9	685	9.0	537	21.1

Comparison results on IAP-B instances are shown in Table 3.10 and in Figure 3.9. First, besides CLPEX, the computational time always follows the order PD < FSH < PD + F&O < FSH + F&O on these 4 instances. On the other hand, the gap always follows the order FSH > PD > PD + F&O > FSH + FO. Second, between two constructive heuristic algorithm, it seems that more computational effort does not lead to better solution quality. On average, the computational time of FSH is 243 seconds while it is 170 seconds for PD algorithm. However, the average gap of FSH algorithm is 74.5%, which is almost double of that of PD algorithm (45.9%). However, the effort pays off when the constructive algorithm is followed with the improving algorithm. With the same improving local search mechanism as F&O algorithm, FSH + F&O gives better results than PD + F&O on all instances. It implies that a better starting solution does not mean a better final solution for F&O algorithm. Third, comparing to CPLEX, both algorithm FSH + F&O and PD + F&O manages to provide better solutions in shorter computational time. The average gap of FSH + F&O algorithm is 9.0% whereas the average gap of PD + F&O is 21.1%. For CPLEX, the average gap is 33.7%. Especially for the most difficult instance R5, the gap is reduced from 91.6% (CPLEX) to 14.2% (FSH + F&O). This shows the efficiency of our developed heuristic algorithms comparing to CPLEX. For the apparel application, a decent solution is provided in a reasonable time (< 12 minutes) by our heuristic algorithm. At last, we observe that the F&O algorithm improves the solution quality quite well, especially for FSH algorithm. Therefore, F&O algorithm remains efficient for our LSP as well as for many cases in the literature in spite of its simple structure.

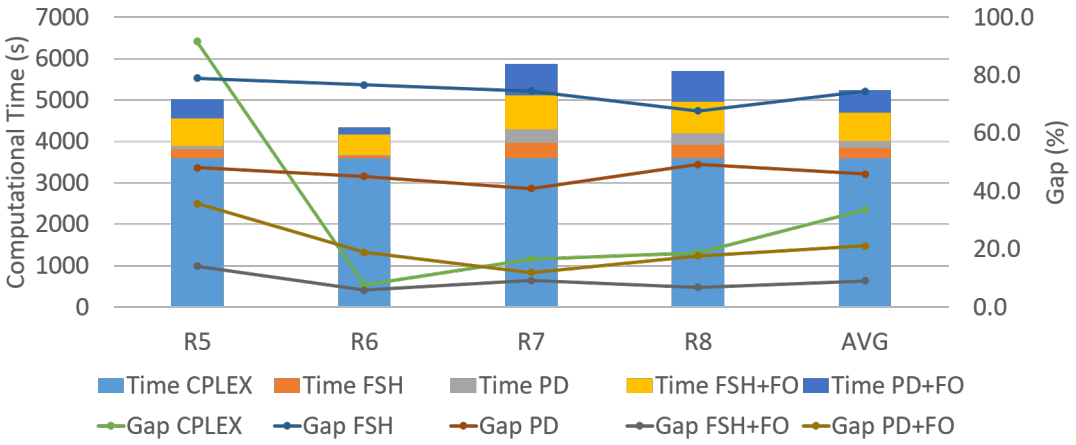


Figure 3.9: Heuristic algorithm comparison results on pilot instances

In the next, we present the results on pseudo-randomly generated testedbed IRG-B and analyze the results. We first give a summary results over all 108 instances in Table 3.11 and 3.12. The comparison results for each instance is given in Table 3.13 - 3.15.

Table 3.11: Computational results: heuristic algorithm on IRG-B summary

	CPLEX		FSH			PD			FSH+FO			PD+FO		
Inst	Time	Gap	Time	Gap	Impro	Time	Gap	Impro	Time	Gap	Impro	Time	Gap	Impro
AVG	3381	91.6	1548	99.5	44.7	248	99.6	57.8	3671	88.3	85.1	1149	90.9	77.4

Based on this summary results, we observe similar trend on computational time and solution quality (measured by value *impro*). Therefore, the performance is consistent on both benchmark instances. The only difference is that PD + F&O algorithm takes even shorter time than FSH algorithm, which is opposite for the real application instances. In terms of computational time, FSH algorithm (1548 seconds) takes much longer than PD algorithm (248 seconds). This is because the instance size of some IRG-B instances is much larger than that of IAP-B, therefore the LP relaxation problem in each iteration gets harder to solve. Even the F&O algorithm computational time are different when the initial solution is provided by FSH algorithm or by PD algorithm. FSH + F&O algorithms takes 3671 seconds on average while PD + F&O algorithm takes 985 seconds on average. Therefore, the average time of FSH + F&O is longer than that of CPLEX already, even though it provides better solution than CPLEX. In terms of solution quality, on average, all heuristic algorithms obtain better solution quality comparing to CPLEX with 1 hour time limit. However, the improvement is not well shown

by gap. One of the reason is as we have mentioned before: when the best lower bound is 0, then any positive objective function value gives a gap 100%. Another reason is that some IRG-B instances might be more difficult to solve due to the larger instance size and the instance generating algorithm. The best average gap is given by FSH + F&O algorithm, which equals 88.3%. We have gained less than 3% in terms of relative gap comparing to CPLEX. Therefore, the gap may not be a good indicator in this case. If we look at the value *impro* instead, then the improvement of solution quality by heuristic algorithms are more obvious. For example, FSH algorithm improve objective function value by 44.7% on average, which is the worst case in all 4 heuristic algorithms.

In Table 3.12, we show the summary result on different types of generated instances. Since the relative gap does not provide too much information, we only show the computational time and the *impro* value for each heuristic algorithm.

Table 3.12: Computational results: heuristic algorithm on IRG-B summary by type

	CPLEX		FSH		PD		FSH+FO		PD+FO	
Inst type	Time	Gap	Time	Impro	Time	Impro	Time	Impro	Time	Impro
DF	3600	99.9	1527	52.8	201	75.7	3818	94.8	1197	89.1
ProdEven	3600	99.9	1741	62.2	380	73.1	4202	90.9	1499	87.2
Capconst	3600	100.0	1675	59.2	176	79.2	3990	93.0	1146	90.0
Capdiff	3600	100.0	1550	52.5	266	66.6	3840	95.3	1203	80.9
STLow	2286	49.7	1199	-2.3	215	-5.8	2342	49.4	807	36.7
DemEven	3600	100.0	1598	43.8	251	58.2	3835	87.2	1042	80.3
All	3381	91.6	1548	44.7	248	57.8	3671	85.1	1149	77.4

We observe that

1), on average comparing to CPLEX, the FSH algorithm takes half of the computational time while improves at least 43% objective function value for all types but the STLow type instances. According to CPLEX results, STlow type instances are easier to solve than other types. Therefore, the FSH algorithm performs better on relatively difficult instances.

2), PD algorithm has a big advantage on short computational time comparing to other algorithms. On average, PD algorithm takes less than 5 minutes and improves the solution quality of CPLEX for most instances. On average, PD algorithm manages to improve solution quality by 57.8% comparing to CPLEX. Like FSH algorithm, PD algorithm fails to obtain better solution quality than CPLEX on STLow type instances. However, the average improvement is better than FSH on all types instances. Therefore we can conclude that PD algorithm is efficient and can provide a feasible solution with

reasonable quality in short time.

3), by combining FSH algorithm with F&O algorithm, the solution quality is further improved, the objective function value is improved by 85.1% over all instances comparing to CPLEX. Actually, for 107 instances over 108, the algorithm gives better solution than CPLEX. However, the computational time of FSH + F&O algorithm is on average around 5 minutes higher than CPLEX.

4), by combining PD algorithm with F&O algorithm, the solution quality is also improved comparing to PD algorithm. The F&O phase increases the average computational time from 248 seconds to 985 seconds, while increases the improving value from 38% to 56%.

Table 3.13: Computational results: heuristic algorithm on IRG-B (1)

	CPLEX			FSH		PD		FSH+FO		PD+FO	
Inst	Gap	Time	Obj	Time	Impro	Time	Impro	Time	Impro	Time	Impro
B1	100.0	3600	329607	261	-72.8	48	50.0	2343	95.7	1027	87.5
B2	100.0	3600	769798	252	-11.4	82	79.5	2435	96.1	1041	94.5
B3	100.0	3600	69398	238	-100.0	43	-100.0	2462	53.8	893	-11.5
B4	98.5	3600	9485931	310	66.7	16	64.4	1023	86.6	399	81.0
B5	100.0	3600	1084421	294	-50.1	39	38.4	2354	92.0	1133	79.1
B6	100.0	3600	6291857	315	57.3	110	59.3	1811	93.8	634	88.9
B7	100.0	3600	88795819	947	96.6	154	98.7	3334	99.9	1254	99.7
B8	100.0	3600	83386167	893	96.8	204	99.8	3541	100.0	937	99.9
B9	100.0	3600	85466741	1040	97.7	77	99.7	3804	99.9	1244	99.8
B10	100.0	3600	83980195	1081	96.1	217	97.9	3630	99.5	1344	99.2
B11	100.0	3600	90015737	1242	95.0	95	95.5	3528	98.5	1145	97.7
B12	100.0	3600	89549343	1111	96.4	120	95.9	3723	99.1	780	97.9
B13	100.0	3600	115106676	3480	98.8	314	99.8	6010	99.9	1474	99.9
B14	100.0	3600	121228481	3505	98.8	454	99.8	6156	99.9	1487	99.9
B15	100.0	3600	12857474	3530	92.2	406	97.8	5935	99.7	1520	98.9
B16	100.0	3600	123816283	1216	3.0	447	96.9	3658	97.0	1823	98.5
B17	100.0	3600	127298893	3932	93.9	369	93.0	6699	96.7	1713	94.8
B18	100.0	3600	116072579	3842	95.8	426	96.1	6273	98.7	1703	97.6
AVG	99.9	3600		1527	52.8	201	75.7	3818	94.8	1197	89.1
B19	100.0	3600	132638	328	-100.0	81	-100.0	2738	60.7	1007	6.7
B20	100.0	3600	58077709	379	95.8	176	99.1	3070	99.8	1426	99.7
B21	100.0	3600	8404981	389	63.5	155	76.0	2883	96.7	1236	95.5
B22	99.5	3600	13053180	374	55.1	40	53.5	1882	80.1	354	77.7
B23	100.0	3600	62081602	372	90.1	92	87.1	2210	95.4	851	94.6
B24	100.0	3600	10247860	410	51.9	82	58.3	2368	89.7	1025	85.9
B25	100.0	3600	92162919	1531	94.3	434	95.7	4294	99.3	1765	99.2
B26	100.0	3600	91068198	1492	95.1	257	97.0	4256	99.4	1525	99.2
B27	100.0	3602	67557529	1349	94.0	331	98.7	4113	99.6	1605	99.4
B28	99.9	3600	43534412	1371	76.9	288	71.2	3829	85.2	1534	84.7
B29	99.6	3600	32120464	1367	63.4	84	59.1	3290	77.1	874	73.8
B30	99.9	3600	35640301	1383	70.8	321	63.5	3510	83.1	1461	80.8
B31	100.0	3600	127510049	5328	96.5	729	98.7	8094	99.1	2072	99.2
B32	100.0	3600	120922514	4989	96.5	742	99.4	7756	99.6	1983	99.6
B33	100.0	3600	119911245	1216	0.6	1166	97.6	3983	96.4	2463	99.1
B34	100.0	3600	118721109	4146	88.9	496	86.0	6913	93.3	1853	91.4
B35	99.8	3600	120294447	3701	89.7	613	87.4	6468	92.6	1953	91.6
B36	100.0	3600	110465941	1216	-3.3	754	86.8	3982	88.7	2004	91.1
AVG	99.9	3600		1741	62.2	380	73.1	4202	90.9	1499	87.2

Table 3.14: Computational results: heuristic algorithm on IRG-B (2)

	CPLEX			FSH		PD		FSH+FO		PD+FO	
Inst	Gap	Time	Obj	Time	Impro	Time	Impro	Time	Impro	Time	Impro
B37	100.0	3600	23204	239	-100.0	69	-100.0	2433	35.6	914	19.8
B38	100.0	3600	1163883	281	-21.3	75	77.2	2657	95.1	785	88.4
B39	100.0	3600	625842	273	-100.0	40	67.9	2570	88.4	750	83.4
B40	100.0	3601	18011097	312	78.3	53	81.2	1631	93.3	453	89.1
B41	100.0	3600	5487443	296	68.4	51	94.5	2814	97.4	977	96.3
B42	99.7	3600	8834308	354	62.9	57	52.1	1544	82.0	485	71.0
B43	100.0	3600	83214545	930	97.6	190	99.8	3575	99.9	1295	99.9
B44	100.0	3600	4254029	782	73.8	80	95.5	2973	99.9	944	98.6
B45	100.0	3600	5242020	832	59.1	85	96.1	3405	99.1	1095	98.3
B46	100.0	3600	92254335	1127	93.8	82	93.7	3118	97.3	1205	96.0
B47	100.0	3600	88539706	1101	96.2	87	96.5	3581	98.7	1033	98.1
B48	99.9	3600	92440700	1327	92.8	107	90.0	3069	95.8	1242	93.2
B49	100.0	3600	121162208	3516	98.5	394	99.8	6167	100.0	1534	99.9
B50	100.0	3600	14690362	3310	83.3	255	97.9	5962	99.6	1338	98.5
B51	100.0	3600	111052268	3833	97.8	490	99.7	6600	99.9	1723	99.8
B52	100.0	3600	114397623	4049	95.9	263	95.8	6815	98.8	1469	97.7
B53	99.9	3600	114067019	3660	94.8	469	94.6	6306	97.3	1691	96.7
B54	99.9	3600	113712126	3934	94.3	323	92.7	6602	96.3	1698	95.3
AVG	100.0	3600		1675	59.2	176	79.2	3990	93.0	1146	90.0
B55	100.0	3600	257750	259	-100.0	128	43.5	2526	92.5	820	74.3
B56	100.0	3600	85133	247	-100.0	115	-100.0	2612	86.2	982	13.6
B57	100.0	3600	18704	243	-100.0	60	-100.0	2300	72.2	888	-65.0
B58	100.0	3600	7976495	331	71.8	37	78.1	1526	96.8	758	94.8
B59	100.0	3601	7333092	310	68.7	206	61.8	1420	93.0	580	80.6
B60	99.8	3600	11514296	322	80.2	54	70.8	1340	95.0	712	88.7
B61	100.0	3600	86537869	934	97.9	277	99.7	3464	99.9	1311	99.8
B62	100.0	3600	91957690	1022	97.3	301	99.6	3778	99.9	1389	99.7
B63	100.0	3600	92125122	1088	97.5	140	99.7	3735	99.9	1236	99.8
B64	100.0	3600	87013629	1142	94.0	157	93.8	3611	97.6	806	96.3
B65	100.0	3600	70307604	1204	92.5	123	90.7	3623	96.4	777	95.0
B66	100.0	3600	20872475	1031	77.9	244	77.6	3494	93.7	1127	87.1
B67	100.0	3600	123548573	3542	98.4	491	99.8	6089	99.9	1591	99.9
B68	100.0	3600	117798084	3469	98.4	398	99.8	6066	99.9	1525	99.9
B69	100.0	3600	13873785	3554	78.7	488	97.6	6207	99.4	1618	99.0
B70	100.0	3600	113178684	4690	95.8	542	96.3	7456	98.3	1892	98.2
B71	100.0	3600	117260538	1216	1.3	614	96.1	3983	96.4	2000	98.2
B72	100.0	3600	117514463	3296	95.2	411	94.0	5895	98.4	1632	96.4
AVG	100.0	3600		1550	52.5	266	66.6	3840	95.3	1203	80.9

Table 3.15: Computational results: heuristic algorithm on IRG-B (3)

	CPLEX			FSH		PD		FSH+FO		PD+FO	
Inst	Gap	Time	Obj	Time	Impro	Time	Impro	Time	Impro	Time	Impro
B73	0.0	37	0	66	-100.0	34	-100.0	77	0.0	44	0.0
B74	0.0	84	0	71	-100.0	59	-100.0	82	0.0	70	0.0
B75	0.0	277	0	113	-100.0	59	-100.0	125	0.0	555	0.0
B76	100.0	3600	5148006	276	97.7	59	93.4	2414	98.6	586	96.4
B77	98.8	3600	7241378	341	82.2	40	44.5	1516	96.7	831	81.0
B78	97.5	3601	5381298	276	88.4	44	89.4	2043	95.7	985	93.7
B79	0.0	465	0	135	-100.0	116	-100.0	200	0.0	163	0.0
B80	0.0	1200	0	544	-100.0	123	-100.0	763	0.0	761	-100.0
B81	0.0	1008	0	446	-100.0	284	-100.0	576	0.0	323	0.0
B82	100.0	3600	87599277	952	99.4	92	95.6	3628	100.0	1128	99.2
B83	99.1	3600	15890973	1028	91.0	227	76.0	3171	97.5	1301	90.9
B84	100.0	3600	88968761	910	99.9	148	97.4	3403	100.0	1181	99.7
B85	0.0	1745	0	2013	-100.0	417	-100.0	2103	0.0	475	0.0
B86	0.0	1809	0	680	-100.0	454	-100.0	769	0.0	905	-100.0
B87	0.0	2127	0	2249	-100.0	520	-100.0	2359	0.0	726	0.0
B88	100.0	3600	116164709	3327	100.0	441	99.6	5361	100.0	1434	99.9
B89	100.0	3600	120352296	4165	99.8	413	99.5	6813	99.9	1552	99.7
B90	100.0	3600	121208248	3991	99.9	347	99.5	6757	100.0	1509	99.7
AVG	49.7	2286		1199	-2.3	215	-5.8	2342	49.4	807	36.7
B91	100.0	3600	7154	213	-100.0	105	-100.0	1181	100.0	519	94.3
B92	100.0	3600	1	233	-100.0	180	-100.0	1415	-100.0	243	100.0
B93	100.0	3600	21171	236	-100.0	58	-100.0	1812	99.2	910	100.0
B94	100.0	3600	54451635	310	97.0	84	97.3	2695	99.8	1186	99.7
B95	100.0	3600	7334313	319	69.9	89	60.5	2855	97.9	1064	92.1
B96	100.0	3600	13755823	338	82.2	48	73.8	2185	96.0	794	92.3
B97	100.0	3600	83679	803	-100.0	286	91.5	1929	99.0	327	100.0
B98	100.0	3600	93744125	966	97.7	265	100.0	3730	100.0	543	100.0
B99	100.0	3600	91176566	936	98.1	201	100.0	3596	100.0	962	100.0
B100	100.0	3600	86302661	1142	92.5	199	91.0	3818	95.8	1021	95.1
B101	99.8	3600	92421684	1262	90.0	234	89.7	2725	93.1	1060	92.6
B102	100.0	3600	89646764	1127	95.6	182	95.6	3814	99.1	1477	98.1
B103	100.0	3600	121114882	3410	98.8	639	100.0	5985	100.0	1032	100.0
B104	100.0	3600	115605436	3616	98.2	472	100.0	6383	100.0	1448	100.0
B105	100.0	3600	118987924	3606	98.4	557	100.0	6373	100.0	797	100.0
B106	100.0	3600	100250337	1215	-23.2	454	93.9	3982	92.2	1840	97.2
B107	100.0	3600	114328741	4228	95.9	325	96.6	6994	98.5	1711	98.9
B108	100.0	3600	112508993	4800	96.9	437	97.4	7566	99.2	1824	99.1
AVG	100.0	3600		1598	43.8	251	58.2	3835	87.2	1042	80.3

3.5 Conclusions

In this chapter, we have developed three constructive heuristic algorithms: F&R algorithm, FSH algorithm and PD algorithm and one improvement heuristic algorithm: F&O algorithm.

First different variants of F&R algorithm and PD algorithm are tested on real-world application instances IAP-B to perform parameter analysis. The result shows that F&R algorithm is not adapted to our problem since it takes very long computational time. This is due to that the partially relaxed problem in each iteration remains hard for CPLEX to solve. Moreover, one type of PD algorithm PD-F-DC has slightly better performance than other its peers therefore is selected to be tested on a larger benchmark instances IRG-B.

All heuristic algorithms except the F&R algorithm are tested on both real application instances IAP-B and pseudo-randomly generated instance IRG-B. On both benchmark instances, all heuristic algorithms have consistent behavior on solution quality. The average *impro* values of algorithm FSH, PD, PD + F&O and FSH + F&O follows increasing order. In other words, FSH algorithm gives the worst performance whereas FSH + F&O algorithm has the best performances based on our experiments. As for the computational time, on benchmark IAP-B, the computational time of algorithm PD, FSH, PD + F&O and FSH + F&O follows increasing order. However, on benchmark IRG-B, PD + F&O algorithm even has shorter computational time than FSH algorithm.

PD algorithm has the advantage of short computational time. It seems to have better performance on relatively difficult instances, which has worse performance on relatively easy instances comparing to CPLEX. FSH algorithm has the same attribute on this that it performance better on relatively difficult instances. On average, it takes longer time than PD algorithm, and obtains worse solution quality. However, when we combine the constructive algorithm with F&O algorithm, FSH + F&O returns better solution and PD + F&O algorithm. Therefore, F&O algorithm does not guarantee a better final solution given a better initial solution. Actually, FSH + F&O algorithm almost always provide better solution than CPLEX, therefore has the best overall performance. However, the bottleneck is the computational time spent on solving LP relation problem in each iteration.

In summary, PD algorithm or PD + F&O algorithm has the advantage of speed, which can be used when computational time is rare resource. FSH algorithm and FSH + F&O algorithm has non negligible computational time, especially when the problem size gets large. However, it returns best solution over all developed algorithm.

Chapter 4

Production Planning Solution to the Apparel Application

Motivated by the apparel manufacturing application introduced in the first chapter, we extracted a complex capacitated lot sizing problem CLSC and have studied it from different points of view. In fact, this problem is constructed by simplifying constraints and aggregating products. In this chapter, we refocus on the application and display the entire production planning solution. The methodology is based on a decomposition approach, and CLSC is solved as the first step of the production planning engine.

The chapter is organized as follows: in Section 4.1, we present the decomposition approach in the application production planning engine. In Section 4.2, we use one application instance to analyze different scenarios and evaluate the system performance. Finally, we conclude in Section 4.3.

4.1 Decomposition Approach

The project scope is production planning and scheduling, and the decomposition framework is shown in Figure 4.1. The production planning problem is solved first, which is followed by a scheduling phase.

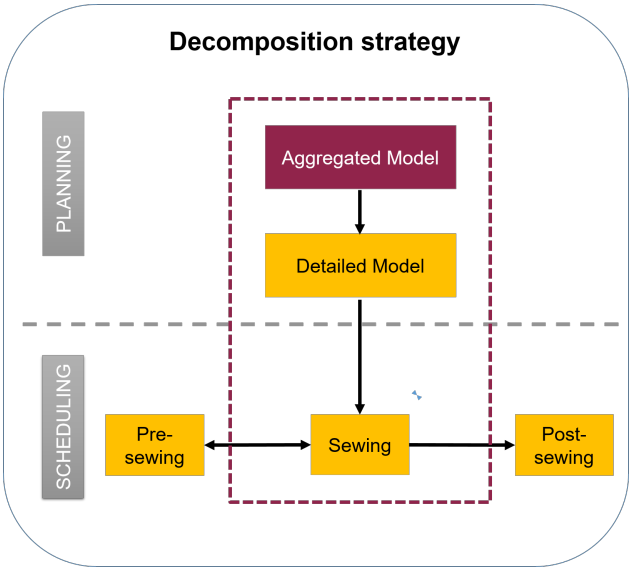


Figure 4.1: Apparel manufacturing decomposition approach

In the planning phase, we build a detailed model and simplify it into an aggregated model, which is CLSC studied in previous chapters. The detailed model has each demand corresponding to a product, but the setup still incurs between style/product families. The aggregated model is solved first. Then we fix the setup of detailed model according to the aggregated model solution. Eventually, the restricted detailed model is solved to obtain a planning solution for the apparel application.

In the scheduling phase, activities on each sewing production line are decided by the planning solution. For each demand on each machine, if the production quantity is nonzero, we combine its productions in all time buckets into one production activity in the scheduling phase. This is due to the fact that we rarely split production of a work order on one production line. Each sewing activity is projected to an activity on other process step. First, scheduling of pre-sewing steps are solved by a greedy algorithm. Then based on the pre-sewing solution, we restrict the starting date of sewing activities based on pre-sewing solution and schedule sewing activities with a commercial constraint scheduling solver CPLEX CPO optimizer. At last, we schedule after-sewing process steps and obtain a complete scheduling solution.

Since our work is mainly the production planning phase, in the following, we focus on explaining detailed steps of the production planning phase in the decomposition approach.

Planning Phase Detailed Model

The aggregated model CLSC have been defined and studied in previous chapters. Therefore, in the following, we formally define the detailed model. The input parameters of the detailed model are:

- $\mathcal{T} = \{1, 2, \ldots, T\}$: set of time buckets.
- $\mathcal{R} = \{1, 2, \ldots, R\}$: set of machines.
- $\mathcal{N} = \{1, 2, \ldots, N\}$: set of product families.
- $\mathcal{D} = \{1, 2, \ldots, D\}$: set of demands.
- cap_{rt}: capacity of machine r in time bucket t ($r \in \mathcal{R}$, $t \in \mathcal{T}$).
- st_{ir}: setup capacity for product family i on machine r ($i \in \mathcal{N}$,$r \in \mathcal{R}$).
- sc_{ir}: setup cost for product family i on machine r ($i \in \mathcal{N}$,$r \in \mathcal{R}$).
- pt_d: capacity required by unitary production of product d ($d \in \mathcal{D}$).
- pc_d: unitary production cost of product required by demand d ($d \in \mathcal{D}$).
- $p_d \in \mathcal{N}$: the required product family of the demand d ($d \in \mathcal{D}$).
- q_d: quantity of product required by demand d ($d \in \mathcal{D}$).
- b_d: release date of demand d ($d \in \mathcal{D}$).
- e_d^1: first due date of demand d ($d \in \mathcal{D}$). No extra cost in interval $[b_d, e_d^1)$.
- e_d^2: second due date of demand d ($d \in \mathcal{D}$).
- tc_d^1: unitary extra cost for demand d satisfied at or after e_d^1 ($d \in \mathcal{D}$).
- tc_d^2: unitary extra cost for demand d satisfied at or after e_d^2 ($d \in \mathcal{D}$).
- lc_d: unitary cost for unsatisfied demand d ($d \in \mathcal{D}, lc_d > tc_d^1 + tc_d^2$).
- $\mathcal{D}^i \subseteq \mathcal{D}$: the subset of demands such that $p_d = i$, i.e., $\mathcal{D}^i \subseteq \mathcal{D}$.

- θ_{dr}: minimum split size for product of demand d on machine r.

The problem is to decide for each machine $r \in \mathcal{R}$ and for each time bucket $t \in \mathcal{T}$, how much to produce of each demand $d \in \mathcal{D}$. The objective is to minimize the total cost including lost sale cost (first priority), tardiness cost (first priority), setup cost (second priority) and production cost (second priority). The restriction includes four parts: 1), the production to satisfy demand d can only start from its release date; 2), the machine capacities cap_{rt} must not be exceeded by the capacity usage for each machine r and time bucket t ($r \in \mathcal{R}$, $t \in \mathcal{T}$); 3), setup occurs for the product families and setup carryover is considered; 4), for each demand d on each machine r, the total production quality of d has to be greater than equals to its minimum split size θ_{dr} if there is a positive production. It comes from the application requirement that each demand can be produced on multiple lines. However, on each production line the production must continue for a minimum number of days. This formulation is very close to the $Form3^{FL}$ formulation of CLSC introduced in Section 2.3.

The link between the CLSC and the detailed model is shown in Table 4.1. Each style family is seen as a product in the aggregated model, whereas each demand corresponds to a product in the detailed model. Therefore, the unitary production time are defined differently. Moreover, we have minimum split size constraints introduced in the detailed model.

Table 4.1: Detailed and aggregated model in planning phase

Parameter	AggregatedModel	DetailedModel
$\mathcal{N}$	Products	Product families
$\mathcal{D}$	Demands	Demands and products
pt	$pt_i = \frac{\sum_{d \in \mathcal{D}^i} pt_d}{\lvert\mathcal{D}^i\rvert}$	pt_d
θ_{dr}	×	✓

We introduce following MIP formulation for the detailed model. For each $i \in \mathcal{N}$, $r \in \mathcal{R}$, $t \in \mathcal{T}$, $d \in \mathcal{D}$, we introduce following variables:

- $x_{drt} \in \mathbb{R}^+$: the production quantity of product i on machine r during time t.
- $z_{dr} \in \{0, 1\}$: it equals to 1 if there is positive production of demand d on machine r.
- $z^0_{irt} \in \{0, 1\}$ equals to 1 if the initial setup state is for product family i on machine r in time bucket t, implying that the final setup state for $t-1$ on r is for product family i.

- $z^+_{irt} \in \{0,1\}$ equals to 1 if there is a state switch for product family i on machine r in time bucket t.
- $w_{rt} \in [0,1]$ is zero if there is more than one product setup on machine r in time bucket t.

The formulation is formally given as follows ($\tilde{\mathcal{T}} = \mathcal{T} \setminus \{1\}$):

$$\min \quad \underbrace{First\ priority: \sum_{d\in\mathcal{D}} lc_d(q_d - \sum_{r,t} x_{drt})}_{lost} + \underbrace{\sum_{d\in\mathcal{D},r\in\mathcal{R},t\in\mathcal{T}:t\geq e^1_d} tc^1_d x_{drt} + \sum_{d\in\mathcal{D},r\in\mathcal{R},t\in\mathcal{T}:t\geq e^2_d} tc^2_d x_{drt}}_{tardiness} \tag{4.1}$$

$$\underbrace{Second\ priority: \sum_{i\in\mathcal{N},r\in\mathcal{R},t\in\mathcal{T}} sc_{ir} z^+_{irt}}_{setup} + \underbrace{\sum_{d\in\mathcal{D},r\in\mathcal{R},t\in\mathcal{T}} pc_d x_{drt}}_{production}$$

$$s.t. \quad \sum_{r\in\mathcal{R},t\in\mathcal{T}} x_{drt} \leq q_d \qquad d\in\mathcal{D} \tag{4.2}$$

$$\sum_{d\in\mathcal{D}} pt_d x_{drt} + \sum_{i\in\mathcal{N}} st_{ir} z^+_{irt} \leq cap_{rt} \qquad r\in\mathcal{R}, t\in\mathcal{T} \tag{4.3}$$

$$x_{drt} \leq \Theta_{drt}(z^0_{irt} + z^+_{irt}) \qquad d\in\mathcal{D}, r\in\mathcal{R}, t\in\mathcal{T} \tag{4.4}$$

$$\sum_{t\in\mathcal{T}} x_{drt} \leq q_d z_{dr} \qquad d\in\mathcal{D}, r\in\mathcal{R}, t\in\mathcal{T} \tag{4.5}$$

$$\sum_{t\in\mathcal{T}} x_{drt} \geq \theta_{dr} z_{dr} \qquad d\in\mathcal{D}, r\in\mathcal{R}, t\in\mathcal{T} \tag{4.6}$$

$$\sum_{i\in\mathcal{N}} z^0_{irt} = 1 \qquad r\in\mathcal{R}, t\in\mathcal{T} \tag{4.7}$$

$$z^0_{irt} \leq z^0_{ir,t-1} + z^+_{ir,t-1} \qquad i\in\mathcal{N}, r\in\mathcal{R}, t\in\tilde{\mathcal{T}} \tag{4.8}$$

$$z^0_{irt} + z^0_{ir,t-1} \leq 1 + w_{r,t-1} \qquad i\in\mathcal{N}, r\in\mathcal{R}, t\in\tilde{\mathcal{T}} \tag{4.9}$$

$$z^+_{irt} + w_{rt} \leq 1 \qquad i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T} \tag{4.10}$$

$$z^0_{irt}, z^+_{irt} \in \{0,1\} \qquad i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T} \tag{4.11}$$

$$z_{dr} \in \{0,1\} \qquad d\in\mathcal{D}, r\in\mathcal{R} \tag{4.12}$$

$$0 \leq w_{rt} \leq 1 \qquad r\in\mathcal{R}, t\in\mathcal{T} \tag{4.13}$$

$$x_{drt} \in [0, q_d] \qquad i\in\mathcal{N}, r\in\mathcal{R}, b_d \leq t\in\mathcal{T} \tag{4.14}$$

Constraints (4.2) guarantee production quantity of demand d is not greater than its required quantity. Constraints (4.3) ensure capacity usage is less than the available

capacity, where the capacity usage consists of both production and setup. Constraints (4.4) link the production x_{drt} and its setup variable z^0_{irt} and z^+_{irt}. Constraints (4.5) link total production quality of d on machine r with the binary variable z_{dr}. Constraints (4.6) force the minimum split size for each production that if there is a positive production, then it has to be equal to or greater than the given minimum split size. Constraints (4.7) - (4.10) are setup carryover related constraints for product families. Remaining constraints are to define the introduced variables.

Planning Phase Decomposition Approach

The decomposition approach is described in Algorithm 8.

Algorithm 8: Planning Phase Decomposition Approach

1 $(\bar{x}_{irt}, \bar{z}^0_{irt}, \bar{z}^+_{irt}) \leftarrow$ Solve aggregated model;
2 Fix setup variables $z^0_{irt} \leftarrow \bar{z}^0_{irt}$, $z^+_{irt} \leftarrow \bar{z}^+_{irt}$ in detailed model;
3 Solve detailed model with only first priority objective.

4.2 Application Performance Analysis

In this section, we perform computational test to evaluate the performance of the system. The test is performed on the application instance R5, which is the most difficult instance we have. The computational results are given in Table 4.2. We have tested the decomposition approach with FSH + FO algorithm developed in the previous chapter, of which the result is reported in Row DA. Moreover, the result for aggregated model and the detailed model is given in Row DA.Aggregated and DA.Detail. We have also tested to solve the instance directly by CPLEX within 2 hours time limit on the above proposed formulation. The result is given in Row Detail MIP whereas its LP relaxation solution is reported in Row Detail LP. For each approach, we present the objective function value including lost sale cost and tardiness cost in Column Obj. Lost sale cost and tardiness cost are given in Column Lost and Tardiness as well for reference. Also given are the computational time and relative Gap in percentage. To be able to compare the gap, we use the LP relaxation value LPV of the corresponding model (reported in Table 2.11) and define the gap as $\frac{Obj-LPV}{Obj}$.

Table 4.2: Application planning solution evaluation

Algorithm	Obj	Lost	Tardiness	Time	Gap (%)
DA.Aggregated	3,443,159	0	3,443,159	1208	14.74
DA.Detail	3,449,146	0	3,449,146	0	21.10
DA	3,449,146	0	3,449,146	1208	21.10
Detailed MIP	1.12E+09	1.12E+09	0	T.L.	99.76
Detailed LP	2,721,535	0	2,721,535	3295	-

*Time unit in seconds, T.L. = 7200s

We observe the efficiency of the proposed decomposition algorithm. Without applying the decomposition method and solve the detailed model directly, the gap is 99% after 2 hours of running and apparently CPLEX got stuck at the first trivial solution that all demands are lost sales. By applying decomposition approach, the gap is reduced to 21% for the detailed model. Moreover, almost all computational time is spent on the aggregated model. Once the aggregated model is solved and we fixed the setup in detailed model, the restricted detailed problem became trivial to solve to optimality. Therefore, even though we probably lose certain optimality at the detailed problem, the solution quality is much better than solving it directly.

4.3 Conclusions

To conclude this section, we have developed a decomposition strategy for the production planning phase in the apparel manufacturing application. The decomposition method provides decent planning solution, which cannot be achieved by solving the detailed model directly using standard MIP solver.

Chapter 5

Capacitated Lot Sizing Problem with A Fixed Product Sequence

In many manufacturing industry, switching production from one product to another will cause setup operations. The setup will consume limited machine capacity and/or cause a setup cost. When the setup depends on the production sequence, i.e., the setup to produce current product depends on both itself and the previous produced product, it is called sequence dependent setup [46, 63]. In this case, both lot sizing and sequencing decisions have to be made. The difficulty of this problem is the factorial number of setup sequence candidates to be chosen from. However, in certain manufacturing industries, this number may be reduced if we restrict the model based on the planners' knowledge. In this chapter, we study a special case of CLSP with sequence dependent setup, which is called capacitated lot sizing problem with a fixed product sequence.

The chapter is organized as follows: In Section 5.1, the classical CLSP with sequence dependent setup is presented with problem definition and literature review. In Section 5.2, we introduce the study motivation and formally define our problem. In Section 5.3 - 5.6, we present study results of this problem which include MIP formulations, a special case study, a column generation heuristic and computational results. Finally, we conclude in Section 5.7.

5.1 Capacitated Lot Sizing Problem with Sequence Dependent Setup

The CLSP with sequence dependent setup we address in this manuscript is defined as follows:

- $\mathcal{N} = \{1, 2, \ldots, N\}$ a set of N products.
- $\mathcal{T} = \{1, 2, \ldots, T\}$ a set of T time buckets
- cap_t: machine capacity in each time bucket $t \in \mathcal{T}$.
- d_{it}: demand of each product $i \in \mathcal{N}$ in time bucket $t \in \mathcal{T}$.
- pt_i: unitary production time of each product $i \in \mathcal{N}$.
- hc_{it}: unitary inventory cost of each product $i \in \mathcal{N}$ in time bucket $t \in \mathcal{T}$.
- b_{it}: the maximum amount of production $i \in \mathcal{N}$ that can be produced in $t \in \mathcal{T}$.
- st_{ij}: setup time from product $i \in \mathcal{N}$ to product $j \in \mathcal{N}$.
- sc_{ij}: setup cost from product $i \in \mathcal{N}$ to product $j \in \mathcal{N}$.

The problem CLSP with sequence dependent setup is to decide the production sequence and the production quantity of each product in each time bucket so that all demands are satisfied with a minimum total cost while respecting the machine capacities, which are consumed by production and setup. Moreover, to clarify the problem we are studying, following assumptions are made:

- The setup state is carried over between time buckets, even preserved over idle time.
- No setup crossover, i.e., the setup has to be finished in one time bucket.
- Only single lot is considered unless it is the first product of the sequence. This implies that one product appears at most once in the setup sequence each time bucket. However, the first product could be the same as the last product in the selected setup sequence.

Let $\mathbb{S}$ be the set of available sequences to schedule products on the machine for each time bucket. Based above assumption, the cardinality of $\mathbb{S}$ equals to $O(n!)$.

Different MIP formulations of this problem are compared in [63], here we only present one of them to further describe the problem. For each sequence $s \in \mathbb{S}$, we define its length

as $L(s)$, the associated setup cost and setup time as $sc(s)$ and $st(s)$. For each product $i \in \mathcal{N}$, each time bucket $t \in \mathcal{T}$ and each candidate sequence $s \in \mathbb{S}$, we introduce following variables:

- $x_{it} \in \mathbb{R}_+$: quantity of product i produced in time bucket t;
- $I_{it} \in \mathbb{R}_+$: inventory of product i at the end of time bucket t;
- $w_{st} \in \{0, 1\}$: it equals to 1 if sequence s is chosen for time bucket t, 0 otherwise;

Then the problem can be formulated as follows:

$$\min \quad \sum_{i\in\mathcal{N},t\in\mathcal{T}} hc_{it}I_{it} + \sum_{s\in\mathbb{S},t\in\mathcal{T}} sc(s)w_{st} \tag{5.1}$$

$$s.t. \quad I_{i,t-1} + x_{it} = I_{it} + d_{it} \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.2}$$

$$\sum_{i\in\mathcal{N}} pt_i x_{it} + \sum_{s\in\mathbb{S}} st(s)w_{st} \leq cap_t \qquad t \in \mathcal{T} \tag{5.3}$$

$$x_{it} \leq b_{it} \sum_{s\in\mathbb{S}:i\in s} w_{st} \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.4}$$

$$\sum_{s\in\mathbb{S}} w_{st} = 1 \qquad t \in \mathcal{T} \tag{5.5}$$

$$\sum_{s\in\mathbb{S}:s_1=i} w_{s,t+1} = \sum_{s\in\mathbb{S}:s_{L(s)}=i} w_{st} \qquad i \in \mathcal{N}, t \in \mathcal{T}\backslash\{T\} \tag{5.6}$$

$$x_{it}, I_{it} \geq 0,\ I_{i0} = 0 \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.7}$$

$$w_{st} \in \{0, 1\} \qquad s \in \mathbb{S}, t \in \mathcal{T} \tag{5.8}$$

The objective function (5.1) includes the inventory cost and setup cost. The material flow balance constraints are formulated as (5.2). Constraints (5.3) ensure that the used capacity does not exceed the available capacity. Constraints (5.4) express that there can be a production for product i only if there is a setup for i, which implies that a sequence containing i is selected. One and only one sub-sequence can be chosen, which is guaranteed by Constraints (5.5). At last, Constraints (5.6) express the consistence of the chosen sequences from one bucket to the next, which means that the last product of bucket t should be the same as the first product of time bucket $t + 1$.

In CLSP with sequence dependent setup, both lot sizing and sequencing decisions have to be made. Therefore, it is often classified as lot sizing and scheduling problem [51, 63]. Sequence dependent setup has been first studied with only setup costs in different context: discrete lot sizing and scheduling like problem [121], discrete lot sizing and scheduling problem [46], proportional lot sizing and scheduling problem [67], uncapacitated LSP [38] and CLSP [68]. Then setup time is also incorporated in the model and

has been studied [39, 70, 117]. Generally, introducing setup times makes the problem more difficult to solve since the feasibility depends on the sequencing decisions as well.

Copil et al. [31] present a review paper on lot sizing and scheduling problem recently. In this survey, all above mentioned models with sequence dependent setups are reviewed and classified. Specially, the literature on CLSP with sequence dependent setups is presented in Section 3.2. The research on CLSP with sequence dependent setups have two major directions: problem modeling and heuristic algorithm design. As for the problem modeling, Guimarães et al. [63] classified and compared different MIP formulations for the CLSP with sequence dependent setup. Based on the sequencing decision, they group MIP formulations into sequence-oriented and product-oriented. The sequence-oriented formulation has binary variables representing the selection of sequences explicitly. Therefore, it has exponential number of variables. The product-oriented formulation formulate the setup sequence as a path in a graph, and subtour elimination constraints are needed to prevent disconnected subtours in the chosen setup sequence. Therefore, we have polynomial number of variables but some models have exponential number of constraints. Due to the difficulty of the problem, different heuristics are developed for different variants of the problem, such as production-balancing algorithm [68, 81], tabu search algorithm [92], variable neighborhood based search algorithm [9, 11], and MIP formulation based algorithm such as fix and relax algorithm [82] and fix and optimize algorithm [93, 126] that we have used previously for CLSP-SC.

5.2 Problem Definition

As mentioned before, the difficulty of CLSP with sequence dependent setup is due to the exponential size of candidate setup sequence that can be chosen for each time bucket. However, in certain industries, this situation might be improved by considering planners knowledge to redefine the model.

We take the color change in production as an example shown in Figure 5.1. There is a most efficient production sequence as $\langle$ white, yellow, orange and black $\rangle$. When the production follows the given sequence from left to right, there are minor setups incurred due to additive color. For instance, when we switch the production from white cups to yellow cups, we need to add the yellow color to the machine. However, if we switch the production from a later product to a previous product in the sequence, there is a major setup occurred due to the machine cleaning. For instance, when the production is switched from black cups to white cups, we need to clean up the entire machine to be able to produce qualified white cups.

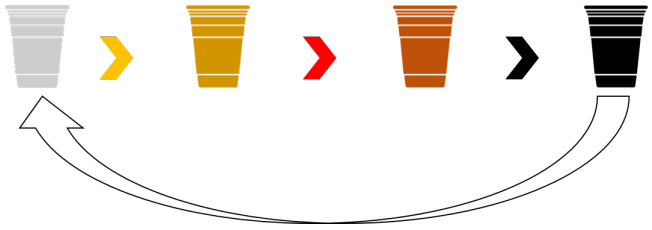


Figure 5.1: Color change in production

In this case, ideally, the chosen setup sequence should follow this given product sequence as much as possible to reduce major setups and capacity loss. In other words, the selected setup sequence should satisfy following conditions:

1. Products position follows the same order as the given sequence.
   Sequence white → orange → yellow is not efficient.

2. Allow to skip products.
   It is possible to switch production from white to black directly without producing the middle products.

3. Allow to restart the sequence.
   Sequence black → white → orange is also valid which has one major setup incurred.

This is the essential concept and motivation of our interest at this restricted model. In the next, we formally define our problem.

Definition 5.1. *Given two sequences* $\omega = \langle \omega_1, \omega_2, \ldots, \omega_n \rangle$ *and* $\alpha = \langle \alpha_1, \alpha_2, \ldots, \alpha_m \rangle$ *(*$m \leq n+1$*), we say* α *follows the order of* ω*, denoted by* $\alpha \preceq \omega$*, if*

1. $\alpha_i \in \omega$ *for* $i \in \{1, 2, \ldots, m\}$.

2. $\alpha_i \neq \alpha_j$ *for* $i \neq j \in \{1, 2, \ldots, m\}$ *and* $\{i, j\} \neq \{1, m\}$.

3. *Let* i *be the index such that* $\omega_i = \alpha_1$ *and define sequence*

$$\beta(i) = \langle \omega_i, \omega_{i+1}, \ldots, \omega_n, \omega_1, \omega_2, \ldots, \omega_{i-1}, \omega_i \rangle \tag{5.9}$$

   There exists a subset $\Omega' = \{\omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_{nm}}\}$ *such that* $\langle \alpha_{\mathbf{1}}, \omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_{n_1}}, \alpha_{\mathbf{2}}, \omega_{i_{n_1+1}}, \omega_{i_{n_1+2}}, \ldots, \omega_{i_{n_2}}, \alpha_{\mathbf{3}}, \ldots, \alpha_{\mathbf{m}}, \omega_{i_{n_{m-1}+1}}, \ldots, \omega_{i_{nm}} \rangle$ *equals to* $\beta(i)$.

The Figure 5.2 illustrates the above definition. The first condition ensures that entries of sequence α are from sequence ω. The second condition excludes repeating

element unless they are the first and the last product. For example, $(\omega_1, \omega_2, \omega_3, \omega_2)$ is not allowed. Finally, the third condition restricts α to be a "sub-sequence" of ω which can reach $\beta(i)$ at maximum. Based on the definition, $\beta(i)$ tries to keep element position as ω as much as possible while gives the possibility to revert the production with a major setup. Moreover, we keep the possibility to skip some elements. For example, given a sequence $\omega = \langle A, B, C, D, E\rangle$, the sequence $a = \langle B, C, A\rangle$ satisfies that $a \preceq \omega$. However, sequence $\langle A, E, C, B\rangle \not\preceq \omega$.

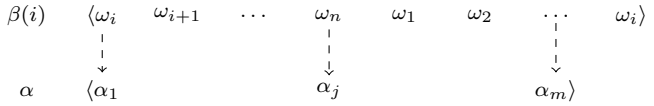


Figure 5.2: Definition 5.1 illustration

Parameters of the CLSP with a fixed product sequence are given as follows:

- $\mathcal{N} = \{1, 2, \ldots, N\}$ a set of N products.
- $\mathcal{T} = \{1, 2, \ldots, T\}$ a set of T time buckets.
- cap_t: machine capacity in each time bucket t.
- d_{it}: demand of each product i in time bucket t.
- pt_i: unitary production time of each product i.
- hc_{it}: unitary inventory cost of each product i in time bucket t.
- b_{it}: the maximum amount of production i that can be produced in t.
- st_{ij}: setup time from product i to product j.
- sc_{ij}: setup cost from product i to product j.
- A permutation of $\mathcal{N}$: $\omega = \langle \omega_1, \omega_2, \ldots, \omega_N\rangle$.

The problem CLSP with a fixed product sequence, denoted as CLSP-FS1, is to decide the production sequence and the production quantity of each product in each time bucket so that all demands are satisfied with a minimum total cost while respecting the machine capacities. Moreover, the chosen setup sequence of each time bucket has to follow the order of ω.

Example 5.1. *We consider three products and four time buckets. The capacity* cap_t *equals to 5 for all* $t = 1, 2, 3$*. The demand and setup is given in Table 5.1. For* $i \in \mathcal{N}$ *and* $t \in \mathcal{T}$*, define* $pt_i = 1$*,* $hc_{it} = 1$*,* $b_{it} = 1$*. The given sequence* $\omega = \langle 1, 2, 3 \rangle$*.*

Table 5.1: CLSP-FS1 Example 5.1 data

d_{it}	t_1	t_2	t_3	t_4
i_1	0	0	1	1
i_2	0	0	1	0
i_3	1	0	1	0

st_{ij}	i_1	i_2	i_3
i_1	0	1	1
i_2	1	0	1
i_3	1	1	0

sc_{ij}	i_1	i_2	i_3
i_1	0	10	10
i_2	10	0	10
i_3	10	10	0

If we do not consider the fixed sequence and solve the problem as a CLSP with sequence dependent setup, then the optimal objective function value is 20 with only setup cost, and the optimal solution is given in Figure 5.3.

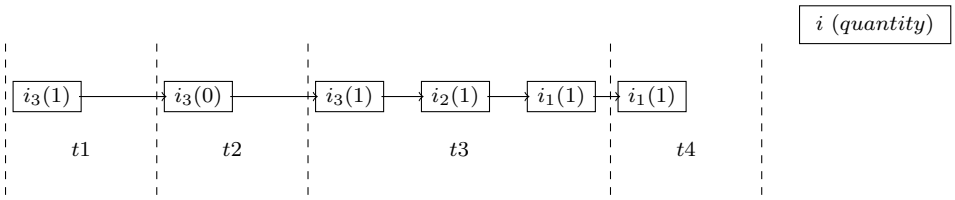


Figure 5.3: CLSP with sequence dependent setup Example 5.1 optimal solution

However, the optimal objective function value of CLSP-FS1 is 21 due to the restriction on production sequence that $\langle 3, 2, 1 \rangle$ *is not a feasible sequence. Therefore, we can not produce all products in time bucket* $t3$*. The optimal solution is given in Figure 5.4.*

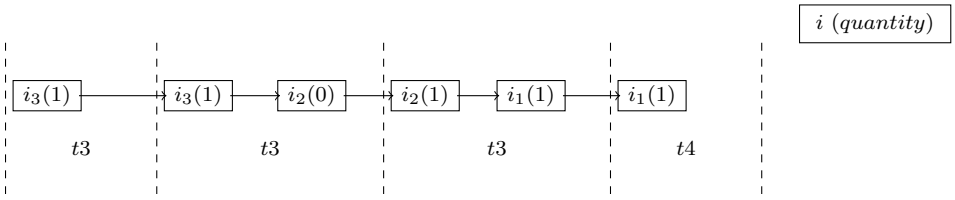


Figure 5.4: CLSP-FS1 Example 5.1 optimal solution

In the literature of CLSP with sequence dependent setup, a concept of "efficient" sequence is proposed by Haase and Kimms [70]. The similar part of two concepts is that once we decide the first product, the last product and other appearing products, the sequence itself is decided. However, the computation of this sequence is polynomial whereas the computation of the "efficient" sequences requires to solve a traveling salesman problem hence intractable.

Theorem 5.1. *CLSP-FS1 is strongly NP-hard.*

Proof. We prove the statement by reduction from CLSP defined in Section 1.2.

Given an instance P1 of CLSP defined as Section 1.2, We construct following CLSP-FS1 instance P2: there are $N + 2T$ products, T time buckets. Let the time bucket set be the same as defined in CLSP as $\mathcal{T}$ and the product set be $\mathcal{N}' = \mathcal{N} \cup \mathcal{N}^s \cup \mathcal{N}^e$ where $\mathcal{N} = \{1, 2, \ldots, N\}$, $\mathcal{N}^s = \{p_s^t : t \in \mathcal{T}\}$ and $\mathcal{N}^e = \{p_e^t : t \in \mathcal{T}\}$. We define:

- Capacity in t equals to cap_t for $t \in \mathcal{T}$.
- For $i \in \mathcal{N}$, demand of i in time bucket t equals to d_{it}. For $t \in \mathcal{T}$, demand of product p_s^t (p_e^t) in time bucket t equals to $d_{p_s^t,t} = 1$ ($d_{p_e^t,t} = 1$), otherwise 0.
- Production time of product $i \in \mathcal{N}$ equals to p_i. Production time of product $i \in \mathcal{N}^s \cup \mathcal{N}^e$ equals to 0.
- Holding cost of product $i \in \mathcal{N}$ in time bucket t equals to h_{it}. Holding cost of product $i \in \mathcal{N}^s \cup \mathcal{N}^e$ in time bucket t equals to ∞.
- Setup cost (time)

$$sc_{ij}(st_{ij}) = \begin{cases} 0 & \textit{If } j \in \mathcal{N}^e \\ \infty & \textit{If } i \in \mathcal{N} \textit{ and } j \in \mathcal{N}^s \\ 0 & \textit{If } i \in \mathcal{N}^e \cup \mathcal{N}^s \textit{ and } j \in \mathcal{N}^s \\ sc_j(st_j) & \textit{If } i \in \mathcal{N} \cup \mathcal{N}^s \textit{ and } j \in \mathcal{N} \\ \infty & \textit{If } i \in \mathcal{N}^e \textit{ and } j \in \mathcal{N} \end{cases}$$

- $\omega = \langle\ p_s^T, p_s^{T-1}, \ldots, p_s^1\ 1, 2, \ldots, N\ p_e^1, p_e^2, \ldots, p_e^T\ \rangle$

In the following, we show the optimal objective function value of P1 equals to P2 with three steps. The idea of the proof is to show that any optimal solution of P2 has structure shown in Figure 5.5, and it has a corresponding solution to P1 which shares the same objective function value.

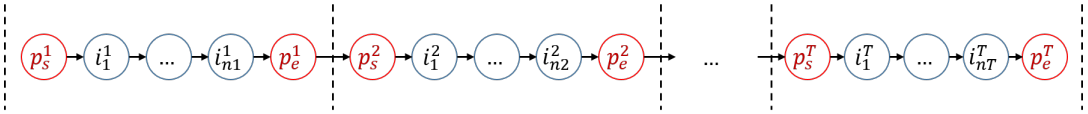


Figure 5.5: Theorem 5.1 CLSP-FS1 instance optimal solution structure

For any solution to P2, it consists of the production quantity of each product i in time bucket t represented as x_{it}, the inventory represented as I_{it} and the setup sequence s_t in time bucket t.

First, we show that any feasible solution to P1 can be transformed into a feasible solution to P2 with the same objective value. Let $S1$ be a feasible solution in the form of decision variables' values in formulation (1.2) - (1.7). For example, $S1(x_{it})$ represents the production quantity of product i in time bucket t of solution $S1$. We construct a solution $S2$ of P2 as follows: $S2(x_{it}) = S1(x_{it})$ for $i \in \mathcal{N}$ and $t \in \mathcal{T}$. $S2(I_{it}) = S1(I_{it})$ for $i \in \mathcal{N}$ and $t \in \mathcal{T}$. $S2(x_{p_s^t,t'}) = 1$ for $t' = t \in \mathcal{T}$, 0 otherwise. $S2(x_{p_e^t,t'}) = 1$ for $t' = t \in \mathcal{T}$, 0 otherwise. $S2(I_{it}) = 0$ for $i \in \mathcal{N}^s \cup \mathcal{N}^e$ and $t \in \mathcal{T}$. For $t \in \mathcal{T}$, let $\alpha = \langle \alpha_1, \alpha_2, \ldots, \alpha_n \rangle$ be a sequence such that $S1(z_{\alpha_i,t}) = 1$ for $\alpha_i \in \alpha$ and $\alpha_i < \alpha_j$ for $i < j$. The chosen sequence in $t > 1$ is $S2(s_t) = \langle p_e^{t-1}, p_s^t, \alpha, p_e^t \rangle$ and $S2(s_t) = \langle p_s^t, \alpha, p_e^t \rangle$ for $t = 1$. We claim that $S2$ is a feasible solution to P2. For product $i \in \mathcal{N}$, material flow constraints are guaranteed by the feasibility of $S1$. For product $i \in \mathcal{N}^e \cup \mathcal{N}^s$, material flow constraints are guaranteed by the construction that there is no inventory during the planning horizon. The chosen sequences' setup time (cost) equals to $\sum_{j:S1(z_{it})=1} st_j$ ($\sum_{j:S1(z_{it})=1} sc_j$) by definition of sequence $S2(s_t)$. Therefore, the capacity constraints hold since production and setup of product $i \notin \mathcal{N}$ do not consume capacities and $S1$ is a feasible solution to P1. Finally, sequence $S2(s_t)$ follows the order of ω and together form a valid planning sequence over the entire time horizon. Moreover,

$$\begin{aligned} obj(S2) &= \sum_{i \in \mathcal{N}', t \in \mathcal{T}} S2(I_{it}) + \sum_{t \in \mathcal{T}} S2(s_t) sc(s_t) \\ &= \sum_{i \in \mathcal{N}, t \in \mathcal{T}} S2(I_{it}) + \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{N}: S1(z_{it})=1} sc_i = obj(S1) \end{aligned}$$

Second, we show that any optimal solution $S2$ of P2 must either satisfy following properties or can be transformed to an equivalent optimal solution satisfying following properties:

- $S2(x_{p_t^s,t'}) = 1$ for $t = t' \in \mathcal{T}$, 0 otherwise.
- $S2(x_{p_t^e,t'}) = 1$ for $t = t' \in \mathcal{T}$, 0 otherwise.
- $S2(I_{it}) = 0$ for $i \in \mathcal{N}^s \cup \mathcal{N}^e$ and $t \in \mathcal{T}$.
- $S2(s_t) = \langle p_{t-1}^s, p_t^s, \alpha, p_t^e \rangle$ where α is a subsequence of $\mathcal{N}$ $\alpha \subseteq \mathcal{N}$ for $1 < t \in \mathcal{T}$.
- $S2(s_t) = \langle p_t^s, \alpha, p_t^e \rangle$ where α is a subsequence of $\mathcal{N}$ $\alpha \subseteq \mathcal{N}$ for $t = 1$.

The first three conditions ensure the demand satisfaction of products in $\mathcal{N}^s \cup \mathcal{N}^e$ without any inventories since the unitary inventory cost equals to ∞. To prove last two conditions hold, we first point out that first three conditions implies that we only have positive production of products in the set $\mathcal{N} \cup \{p_t^s, p_t^e\}$ in time bucket t. Moreover, having setup of products not in this set does not improve the solution. Due to the setup cost from any product $i \in \mathcal{N}$ to p_t^s is ∞ and the setup cost from p_t^e to any product $i \in \mathcal{N}$ is ∞, in the chosen sequence $S2(s_t)$, p_t^s must be before any $i \in \mathcal{N}$ while p_t^e must be after any $i \in \mathcal{N}$. In other words, $S2(s_t) = \langle ..., p_t^s, i_1, ..., i_n, p_t^e, ... \rangle$. Finally, due to the last product of $t-1$ is p_{t-1}^e, the setup sequence in t will start from p_{t-1}^e. Hence, all properties hold.

Third, any optimal solution of P2 can be transformed into a solution to P1 with the same objective value. Based on the second statement, let S2 be an optimal solution to P2 satisfying above properties. Then we construct a feasible solution $S1$ to P1 as follows: $S1(x_{it}) = S2(x_{it})$ for $i \in \mathcal{N}$ and $t \in \mathcal{T}$. $S1(I_{it}) = S2(I_{it})$ for $i \in \mathcal{N}$ and $t \in \mathcal{T}$. $S1(z_{it}) = 1$ if $i \in S2(s_t)$ for $i \in \mathcal{N}$ and $t \in \mathcal{T}$. Due to the above properties, $S1$ is feasible and share the same objective function value as $S2$.

□

CLSP with only setup cost is still NP-hard. Therefore based on our proof procedure, we have

Corollary 5.1. *CLSP-FS1 with only setup cost is NP-hard.*

Actually, CLSP can be seen as a special case of CLSP-FS1 with $sc_{ii} = sc_i$, $st_{ii} = st_i$ for all products i. Therefore, the complexity result holds directly. However, with our proof, we prove that even under the case that $sc_{ii} = 0$, $st_{ii} = 0$, the problem is still strongly NP-hard.

Corollary 5.2. *CLSP-FS1 with setup $sc_{ii} = 0$, $st_{ii} = 0$ for $i \in \mathcal{N}$ is still strongly NP-hard.*

5.3 Problem Formulation

In this section, we introduce MIP formulations for CLSP-FS1. There are two types of decisions to make: lot sizing and sequencing. For lot sizing, we have the classical aggregated formulation and facility location based formulation inherited from CLSP (see Section 1.2). For sequencing, we have product-oriented formulation with compact size and sequence-oriented formulation with exponential size. In the next, we introduce them in details.

Aggregated sequence-oriented formulation (AG-SO)

Let $\mathcal{S} = \{s : s \preceq \omega\}$. Given a sequence $s \in \mathcal{S}$ with length $L(s)$, its associated setup cost $sc(s)$ and setup time $st(t)$ are defined as follows:

$$sc(s) = \sum_{k=1}^{L(s)-1} sc_{s_k,s_{k+1}} \qquad st(s) = \sum_{k=1}^{L(s)-1} sc_{s_k,s_{k+1}}$$

We introduce the following variables for $i \in \mathcal{N}$, $t \in \mathcal{T}$ and $s \in \mathcal{S}$:

- $x_{it} \in \mathbb{R}_+$: quantity of product i produced in time bucket t;
- $I_{it} \in \mathbb{R}_+$: inventory of product i at the end of time bucket t;
- $w_{st} \in \{0, 1\}$: it equals to 1 if sequence s is chosen for time bucket t, 0 otherwise.

The problem can be formulated as follows:

$$\min \quad \sum_{i\in\mathcal{N},t\in\mathcal{T}} hc_{it}I_{it} + \sum_{s\in\mathcal{S},t\in\mathcal{T}} sc(s)w_{st} \tag{5.10}$$

$$s.t. \quad I_{i,t-1} + x_{it} = I_{it} + d_{it} \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.11}$$

$$\sum_{i\in\mathcal{N}} pt_i x_{it} + \sum_{s\in\mathcal{S}} st(s)w_{st} \le cap_t \qquad t \in \mathcal{T} \tag{5.12}$$

$$x_{it} \le b_{it} \sum_{s\in\mathcal{S}:i\in s} w_{st} \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.13}$$

$$\sum_{s\in\mathcal{S}} w_{st} = 1 \qquad t \in \mathcal{T} \tag{5.14}$$

$$\sum_{s\in\mathcal{S}} s_1 w_{s,t+1} = \sum_{s\in\mathcal{S}} s_{L(s)} w_{st} \qquad t \in \mathcal{T}\backslash\{T\} \tag{5.15}$$

$$x_{it}, I_{it} \ge 0,\ I_{i0} = 0 \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.16}$$

$$w_{st} \in \{0, 1\} \qquad s \in \mathcal{S}, t \in \mathcal{T} \tag{5.17}$$

The objective function (5.10) includes the inventory cost and the setup cost. The material flow balance constraints is formulated as (5.11). Constraints (5.12) ensure that the used capacity does not exceed the available capacity. Constraints (5.13) express that there can be a production for product i only if there is a setup for i, which implies a subsequence containing i is selected. One and only one sub-sequence can be chosen, which is guaranteed by constraints (5.14). At last, constraints (5.15) express the consistence of the chosen sequences between time buckets, which means that the last setup product of time bucket t should be the same as the first setup product of time bucket $t + 1$.

Aggregated product-oriented formulation (AG-PO)

In this section, we introduce a product-oriented formulation which does not have the sequence as a variable explicitly. The setup sequence can be represented as a path in the graph.

First of all, we have following observations regarding the valid setup sequences in CLSP-FS1:

Observation 5.1. *Given a sequence ω, any sequence α with length $L(\alpha) \leq 2$ and $\alpha_i \in \omega$ $\forall i$ follows the order of ω: $\alpha \preceq \omega$.*

Observation 5.2. *Given a sequence ω, if the first product, the last product and its appearing products of a sequence s are known, and $s \preceq \omega$, then the sequence s is fixed.*

This holds due to the definition. In other words, we only need to know all elements and the first and the last element of a sequence to build up the whole information if it follows the order of a given sequence.

Property 5.1. *CLSP with sequence dependent setup has $O(n!)$ valid sequences, while CLSP-SD has $O(n2^n)$ sequences.*

We define the CLSP-FS1 associated graph $G = (V, E)$ as follows:

- Vertex V
  - Source P and sink Q
  - $V_i^t := \{v_{i0}^t\} \cup \{v_{ij}^t : j \in \mathcal{N}\}$ for $t \in \mathcal{T}$, $i \in \mathcal{N}$.
- Edge E
  - E^1: (P, v_{i0}^1) for $i \in \mathcal{N}$.
  - E^2: (v_{ij}^t, v_{ik}^t) for $i \in \mathcal{N}$, $j < k \in \mathcal{N}$ and $t \in \mathcal{T}$.
  - E^3: (v_{ij}^t, v_{j0}^{t+1}) for $i \in \mathcal{N}$, $j \in \mathcal{N}$ and $t \in \mathcal{T} \backslash \{T\}$.
  - E^4: (v_{ij}^T, Q) for $i \in \mathcal{N}$ and $j \in \mathcal{N}$.

Except the source and the sink node, each node v represents a product $p(v)$ and belongs to a time bucket $t(v)$ which is defined as follows:

$$p(v_{i0}^t) = \omega_i \quad p(v_{ij}^t) = (\beta(i))_{j+1} \quad t(v_{i0}^t) = t \quad t(v_{ij}^t) = t \quad \forall j \in \mathcal{N}$$

where $\beta(i)$ is defined as (5.9) and $(\beta(i))_j$ is the jth element of $\beta(i)$. We also introduce following notations to link products $i \in \mathcal{N}$ and time buckets $t \in \mathcal{T}$ with the graph:

$$E(i,t) := \{(u,v) \in E | p(u) = i \text{ and } t(u) = t\}$$
$$E(t) := \{(u,v) | t(u) = t\}$$

Moreover, we define the setup cost/time of each edge $(u,v) \in E$ accordingly as follows:

$$sc(u,v) = \begin{cases} sc_{p(v),p(v)} & (u,v) \in E^1 \\ sc_{p(u),p(v)} & (u,v) \in E^2, E^3 \\ 0 & (u,v) \in E^4 \end{cases} \tag{5.18}$$

$$st(u,v) = \begin{cases} st_{p(v),p(v)} & (u,v) \in E^1 \\ st_{p(u),p(v)} & (u,v) \in E^2, E^3 \\ 0 & (u,v) \in E^4 \end{cases} \tag{5.19}$$

An example with 3 products A, B, C and 2 time buckets is given in Figure 5.6 for illustration:

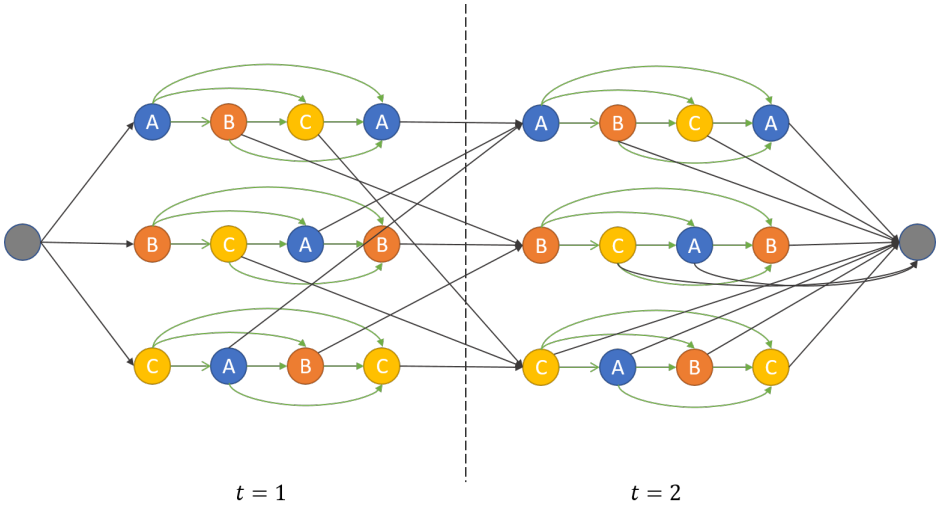


Figure 5.6: CLSP-FS1 graph representation of setup sequence

Property 5.2. *A valid setup sequence of the entire planning horizon for CLSP-FS1 is a path from source node P to sink node Q in the graph $G = (V, E)$.*

According to the Property 5.2, we decide the setup sequence by forming a path in the product-oriented formulation. Let $G = (V, E)$ be the induced graph presented before. We introduce following variables for each edge $(u,v) \in E$:

- $T_{uv} \in \{0,1\}$ equals to 1 if the edge $(u,v) \in E$ is selected, 0 otherwise.

Then the product oriented formulation (AG-PO) can be formally formulated as follows:

$$\min \quad \sum_{i\in\mathcal{N},t\in\mathcal{T}} hc_{it}I_{it} + \sum_{(u,v)\in E} sc(u,v)T_{uv} \tag{5.20}$$

$$s.t. \quad I_{i,t-1} + x_{it} = I_{it} + d_{it} \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.21}$$

$$\sum_{i\in\mathcal{N}} pt_i x_{it} + \sum_{(u,v)\in E(t)} st(u,v)T_{uv} \leq cap_t \qquad t \in \mathcal{T} \tag{5.22}$$

$$x_{it} \leq b_{it} \sum_{(u,v)\in E(i,t)} T_{uv} \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.23}$$

$$\sum_{(P,v)\in E} T_{Pv} = 1 \tag{5.24}$$

$$\sum_{(v,Q)\in E} T_{vQ} = 1 \tag{5.25}$$

$$\sum_{(u,v)\in E} T_{uv} = \sum_{(v,u)\in E} T_{vu} \qquad v \in V\backslash\{P,Q\} \tag{5.26}$$

$$x_{it}, I_{it} \geq 0,\ I_{i0} = 0 \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.27}$$

$$T_{uv} \in \{0,1\} \qquad (u,v) \in E \tag{5.28}$$

Like the previous formulation, the objective is to minimize the total cost including inventory cost and setup cost. Specially, the setup cost is defined as the total cost of selected edges. Constraints (5.21) are to maintain material flow balance as in previous formulation. Constraints (5.22) guarantee the machine capacity is not exceeded by the used capacity. The setup time is defined similarly as setup cost. Constraints (5.23) link production and setup. Constraints (5.24) - (5.26) are flow balance constraints for each node in graph $G = (V, E)$ to form a path, which is a valid setup sequence for entire time horizon.

Facility location based sequence-oriented formulation (FL-SO)

Another straightforward formulation for the CLSP-SD is the facility location based formulation. Instead of production variables x_{it}, we introduce following variables for each product $i \in \mathcal{N}$ and time bucket $t \leq k \in \mathcal{T}$:

- x_{itk}: the production quantity of product i in time bucket t to satisfy the demand in time bucket k.

Then the FL sequence-oriented formulation (FL-SO) is formally defined as follows:

$$\min \quad \sum_{i\in\mathcal{N},t\in\mathcal{T}} hc_{it}\sum_{k=1}^{t}\sum_{l=t+1}^{T} x_{ikl} + \sum_{s\in\mathcal{S},t\in\mathcal{T}} sc(s)w_{st} \tag{5.29}$$

$$s.t. \quad \sum_{k=1}^{t} x_{ikt} = d_{it} \qquad i\in\mathcal{N}, t\in\mathcal{T} \tag{5.30}$$

$$\sum_{i\in\mathcal{N}}\sum_{k=t}^{T} pt_i x_{itk} + \sum_{s\in\mathcal{S}} st(s)w_{st} \le cap_t \qquad t\in\mathcal{T} \tag{5.31}$$

$$x_{itk} \le \min\{b_{it}, d_{ik}\}\sum_{s\in\mathcal{S}:i\in s} w_{st} \qquad i\in\mathcal{N}, t\le k\in\mathcal{T} \tag{5.32}$$

$$\sum_{s\in\mathcal{S}} w_{st} = 1 \qquad t\in\mathcal{T} \tag{5.33}$$

$$\sum_{s\in\mathcal{S}} s_1 w_{s,t+1} = \sum_{s\in\mathcal{S}} s_{L(s)} w_{st} \qquad t\in\mathcal{T}\backslash\{T\} \tag{5.34}$$

$$x_{itk} \ge 0 \qquad i\in\mathcal{N}, t\le k\in\mathcal{T} \tag{5.35}$$

$$w_{st}\in\{0,1\} \qquad s\in\mathcal{S}, t\in\mathcal{T} \tag{5.36}$$

There is a direct relation between variables x_{it}, I_{it} and newly introduced variable x_{itk} that $x_{it} = \sum_{k\ge t}^{T} x_{itk}$ and $I_{it} = \sum_{k=1}^{t}\sum_{l=t+1}^{T} x_{ikl}$. Therefore, the objective function (5.29) is to minimize the inventory cost and setup cost by substitution. Constraints (5.30) ensure demands satisfaction.

Facility location based product-oriented formulation (FL-PO)

The previous formulation is based on the sequence-oriented formulation, therefore another formulation will be combining aggregated lot sizing decision with product-oriented sequencing formulation. The facility location based product-oriented formulation (FL-PO) is given as follows:

$$\min \quad \sum_{i\in\mathcal{N},t\in\mathcal{T}} hc_{it}\sum_{k=1}^{t}\sum_{l=t+1}^{T} x_{ikl} + \sum_{(u,v)\in E} sc(u,v)T_{uv} \tag{5.37}$$

$$s.t. \quad \sum_{k=1}^{t} x_{ikt} = d_{it} \qquad i\in\mathcal{N}, t\in\mathcal{T} \tag{5.38}$$

$$\sum_{i\in\mathcal{N}}\sum_{k=t}^{T} pt_i x_{itk} + \sum_{(u,v)\in E} st(u,v)T_{uv} \le cap_t \qquad t\in\mathcal{T} \tag{5.39}$$

$$x_{itk} \le \min\{b_{it}, d_{ik}\}\sum_{(u,v)\in E(i)} T_{uv} \qquad i\in\mathcal{N}, t\le k\in\mathcal{T} \tag{5.40}$$

$$\sum_{(P,v)\in E} T_{Pv} = 1 \tag{5.41}$$

$$\sum_{(v,Q)\in E} T_{vQ} = 1 \tag{5.42}$$

$$\sum_{(u,v)\in E} T_{uv} = \sum_{(v,u)\in E} T_{vu} \qquad v \in V\backslash\{P,Q\} \tag{5.43}$$

$$x_{itk} \geq 0 \qquad i \in \mathcal{N}, t \leq k \in \mathcal{T} \tag{5.44}$$

$$T_{uv} \in \{0,1\} \qquad (u,v) \in E \tag{5.45}$$

5.4 A Special Case Study

Given a product sequence, if the production follows the given sequence, the setup is minor. However, when we need to reverse products in the sequence, there will be a major setup occurred. Here we study an extreme case to have the minor setup as zero and the major setup as a positive number. More specifically, we define

$$st_{\omega_i,\omega_j} = \begin{cases} 0 & i \leq j \\ \Delta_{st} & otherwise \end{cases} \qquad sc_{\omega_i,\omega_j} = \begin{cases} 0 & i \leq j \\ \Delta_{sc} & otherwise \end{cases}$$

where $\Delta_{st} > 0$ and $\Delta_{sc} > 0$. Without loss of generality, we can assume the fixed sequence is $\langle 1, 2, \ldots, N\rangle$ by reindexing. Then the setup matrices $\{st_{ij}\}_{i,j\in\mathcal{N}}$, $\{sc_{ij}\}_{i,j\in\mathcal{N}}$ have following structure:

$$\begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 \\ \Delta_{st} & 0 & 0 & \ldots & 0 & 0 \\ \Delta_{st} & \Delta_{st} & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ \Delta_{st} & \Delta_{st} & \Delta_{st} & \ddots & 0 & 0 \\ \Delta_{st} & \Delta_{st} & \Delta_{st} & \ldots & \Delta_{st} & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 \\ \Delta_{sc} & 0 & 0 & \ldots & 0 & 0 \\ \Delta_{sc} & \Delta_{sc} & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ \Delta_{sc} & \Delta_{sc} & \Delta_{sc} & \ddots & 0 & 0 \\ \Delta_{sc} & \Delta_{sc} & \Delta_{sc} & \ldots & \Delta_{sc} & 0 \end{bmatrix}$$

We refer this special case of CLSP-FS1 as CLSP-FS1-LT since its setup matrices are lower triangle. In the following theorem, CLSP-FS1-LT is shown to be NP-hard.

Theorem 5.2. *CLSP-FS1-LT is NP-hard.*

Proof. The proof is based on reduction from CLSP with single product.

Given a CLSP instance P1 with one product and T' time buckets, for each time bucket t, the demand is d'_t, the unitary holding cost is hc'_t, the unitary processing time is pt' and the setup cost is sc'. The capacity for each time bucket t is cap'_t. Without

loss of generality, we assume $pt' > 0$, otherwise the solution is trivial. This problem is NP-hard [20].

Now we build a CLSP-FS1-LT instance P2 based on $P1$. Let $\mathcal{N} = \{1, 2\}$ and $\mathcal{T} = \{1, 2, \ldots, 2T'\}$. Define $\mathcal{T}^1 = \{2t - 1 \in \mathcal{T} : 1 \leq t \leq T'\}$ and $\mathcal{T}^2 = \{2t \in \mathcal{T} : 1 \leq t \leq T'\}$. If $t \in \mathcal{T}^1$, $cap_t = 0$; otherwise, $cap_t = cap'_{t/2}$. Other parameters of P2 are summarized in Table 5.2. The given product sequence is $\langle 1, 2\rangle$. Therefore, all sequences are feasible, including $\langle 1\rangle$, $\langle 1, 2\rangle$, $\langle 1, 2, 1\rangle$, $\langle 2\rangle$, $\langle 2, 1\rangle$ and $\langle 2, 1, 2\rangle$.

Table 5.2: Theorem 5.2 proof CLSP-FS1-LT instance parameters

	$i = 1$		$i = 2$	
Parameter	$t \in \mathcal{T}^1$	$t \in \mathcal{T}^2$ $(u = \frac{t}{2})$	$t \in \mathcal{T}^1$	$t \in \mathcal{T}^2$ $(u = \frac{t}{2})$
d_{it}	0	d'_u	1	0
h_{it}	0	h'_u	∞	∞
pt_i	pt'		0	

st_{ij}	1	2
1	0	0
2	st'	0

sc_{ij}	1	2
1	0	0
2	sc'	0

We claim that P1 and P2 are equivalent in the sense that P1 is feasible if and only if P2 is feasible and its optimal objective value is less than ∞. The idea of the problem is to show that any optimal solution of P2 has structure shown in Figure 5.7, and it corresponds to a solution to P1 which shares the same objective function value.

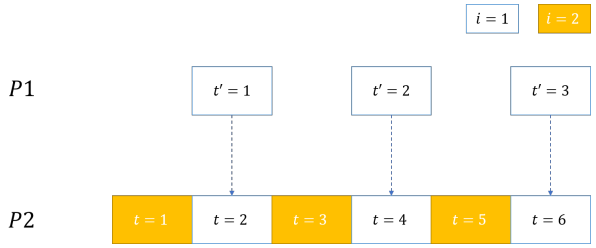


Figure 5.7: Theorem 5.2 reduction from CLSP with single product to CLSP-FS1-LT

This statement is proved by following 3 arguments.

1. Any feasible solution of P1 corresponds to a feasible solution S2 to P2 and they share the same objective function values. Let $S1$ be an optimal solution of P1, $S1(x_t)$, $S1(z_t)$ and $S1(I_t)$ represent the solution value of the production quantity, the setup and the inventory in time bucket t. Then we construct solution $S2$ to problem P2 in the form of decision variables shown in formulation AG-SO as follows: $S2(x_{1,2t}) = S1(x_t)$, $S2(x_{1,2t-1}) = 0$ for $1 \leq t \leq T'$. $S2(x_{2,2t}) = 0$, $S2(x_{2,2t+1}) = 1$ for $1 \leq t \leq T'$. $S2(I_{1,2t}) = S2(I_{1,2t-1}) = S1(I_t)$ for $1 \leq t \leq T'$. $S2(I_{1,1}) = 0$. $S2(I_{2,t}) = 0$ for all $t \in \mathcal{T}$. If $t \in \mathcal{T}^1$, $S2(s_t) = \langle 2\rangle$. If $t \in \mathcal{T}^2$, $S2(s_t) = \langle 2, 1, 2\rangle$ if $S1(z_t) = 1$; $S2(s_t) = \langle 2\rangle$ if

$S1(z_t) = 0$. First, by construction, we have

$$\begin{aligned} Obj(S2) &= \sum_{i\in\mathcal{N},t\in\mathcal{T}} hc_{it}S2(I_{it}) + \sum_{t\in\mathcal{T}} sc(S2(s_t)) \\ &= \sum_{1\le t\le T'} hc'_t S1(I_t) + \sum_{t\in\mathcal{T}} sc(S2(s_t)) \\ &= \sum_{1\le t\le T'} hc'_t S1(I_t) + \sum_{t\in\mathcal{T}^2} sc(S2(s_t)) \\ &= \sum_{1\le t\le T'} hc'_t S1(I_t) + \sum_{1\le t\le T', S1(z_t)=1} sc' = Obj(S1) \end{aligned}$$

The second equation is by construction of S2. The third and fourth equation is because of the sequence construction in S2. For $t \in \mathcal{T}^1$ there is no setup since the selected sequence is always $\langle 2\rangle$, while for $t \in \mathcal{T}^2$ there is a setup if and only if the selected sequence is $\langle 2, 1, 2\rangle$, which implies $S1(z_t) = 1$ by construction. Therefore, the objective function value of $S2$ equals to that of SI. Moreover, $S2$ is a feasible solution of P2. First, the capacity restriction is respected on $t \in \mathcal{T}^1$ due to that the only production is one unit of product 2 which consumes 0 capacity and there is no setup. For $t \in \mathcal{T}^2$, we have $S2(x_{1,t})pt_1 + S2(x_{2,t}pt_2) = S2(x_{1,t})pt_1 = S1(x_t)pt' \le cap'_t = cap_t$. Second, the production sequence is consistent from one time bucket to the next one due to construction.

2. Given an optimal solution $S2$ to P2, $S2$ satisfies following properties or it can be transformed to an equivalent optimal solution with the same objective function value that follows properties:

- For $t \in \mathcal{T}^1$, $S2(x_{2,t}) = 1$; for $t \in \mathcal{T}^2$, $S2(x_{2,t}) = 0$.
- For $t \in \mathcal{T}$, $S2(I_{2,t}) = 0$.
- For $t \in \mathcal{T}^1$, $S2(x_{1,t}) = 0$.
- For $t \in \mathcal{T}^1$, $S2(s_t) = \langle 2\rangle$.
- For $t \in \mathcal{T}^2$, $S2(s_t) = \langle 2\rangle$ if $S2(x_{1,t}) = 0$ or $S2(s_t) = \langle 2, 1, 2\rangle$ otherwise.

The first two properties guarantee that product 2 is produced only in the demanding time bucket without any inventory since the unitary holding cost is ∞. The third property holds because there is no capacity to perform production of product 1 in time bucket $t \in \mathcal{T}^1$. Based on the first three properties, the nonzero production in $t \in \mathcal{T}^1$ can only be product 2 and the nonzero production in $t \in \mathcal{T}^2$ can only be product 1.

Therefore, we can always have the setup sequence $S2(s_t) = \langle 2 \rangle$ in $t \in \mathcal{T}^1$ and $S2(s_t) = \langle 2 \rangle$ or $\langle 2, 1, 2 \rangle$ in $t \in \mathcal{T}^2$. Other sequences are either equivalent or not optimal.

3. An optimal solution $S2$ to P2 corresponds to a feasible solution $S1$ to P1 and they share the same objective function value. Due to the second argument, we assume that S2 satisfies above properties as an optimal solution. Then we can build a solution S1 of problem P1 as follows: for $t \in \mathcal{T}'$, $S1(x_t) = S2(x_{1,2t})$, $S1(I_t) = S2(I_{1,2t})$, $S1(z_t) = 1$ if $S1(x_t) > 0$.

First, by construction, we have

$$\begin{aligned} Obj(S1) &= \sum_{t \in \mathcal{T}'} hc'_t S1(I_t) + \sum_{t \in \mathcal{T}'} sc' S1(z_t) \\ &= \sum_{t \in \mathcal{T}^2} hc_{1,t} S2(I_{1,t}) + \sum_{t \in \mathcal{T}^2 : S2(x_{1,t}) > 0} sc' \\ &= \sum_{i \in \mathcal{N}, t \in \mathcal{T}} hc_{it} S2(I_{it}) + \sum_{t \in \mathcal{T}} sc(S2(s_t)) = Obj(S2) \end{aligned}$$

Therefore, the objective function value of $S2$ equals to that of $S1$. The feasibility of $S1$ comes from the feasibility of $S2$ that capacity is respected in time bucket $t \in \mathcal{T}^2$ and the flow balance is conserved on product 1.

From above arguments, P1 and P2 are equivalent in the sense that the optimal objective function value of P1 equals to that of P2. Therefore, CLSP-FS1-LT is NP-hard.

□

From the proof procedure, we have following result holds:

Corollary 5.3. *In the Definition 5.1, if we require α to be a sub-sequence of $\beta(i)$, i.e., without the possibility to skip some products in the middle, the problem is still NP-hard.*

Due to the special structure of the setup matrix and the definition of the problem, we have one direct observation

Observation 5.3. *Given a feasible solution to CLSP-FS1-LT, there is maximum one nonzero setup occurred in each time bucket. Moreover, if a nonzero setup occurs, then all products can be setup in this time bucket.*

In this special case, the sequencing decision to make is simplified to decide the first (last) product of each time bucket and to decide whether we perform a major (nonzero) setup or not in this time bucket. Therefore, we can reformulate it to a simplified model. For each product $i \in \mathcal{N}$ and time bucket $t \in \mathcal{T}$, we introduce following sequencing variables:

- $z_t \in \{0,1\}$: it equals to 1 if there is a major setup in time bucket t.
- $f_{it} \in \{0,1\}$: it equals to 1 if product i is the first product in the setup sequence in time bucket t. It also represents the last product in the time bucket $t-1$.
- $z_{it} \in \{0,1\}$: it equals to 1 if there is a setup for product i in time bucket t.

Together with variables x_{it} and I_{it} as introduced before, the reformulation of CLSP-FS1-LT is given as follows:

$$\min \quad \sum_{i\in\mathcal{N},t\in\mathcal{T}} hc_{it}I_{it} + \sum_{t\in\mathcal{T}} \Delta_{sc} z_t \tag{5.46}$$

$$s.t. \quad I_{i,t-1} + x_{it} = I_{it} + d_{it} \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.47}$$

$$\sum_{i\in\mathcal{N}} pt_i x_{it} + \Delta_{st} z_t \leq cap_t \qquad t \in \mathcal{T} \tag{5.48}$$

$$x_{it} \leq b_{it} z_{it} \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.49}$$

$$f_{i,t+1} \leq \sum_{j\in\mathcal{N}:j\leq i} f_{jt} + z_t \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.50}$$

$$f_{i,t+1} \leq \sum_{j\in\mathcal{N}:j\geq i} f_{jt} + (1 - z_t) \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.51}$$

$$z_{it} \leq (z_t - 1) + (\sum_{j\in\mathcal{N}:j\leq i} f_{jt} + \sum_{j\in\mathcal{N}:j\geq i} f_{j,t+1}) \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.52}$$

$$z_{it} \leq (1 - z_t) + (\sum_{j\in\mathcal{N}:j\leq i} f_{jt} + \sum_{j\in\mathcal{N}:j\geq i} f_{j,t+1}) \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.53}$$

$$\sum_{i\in\mathcal{N}} f_{it} = 1 \qquad t \in \mathcal{T} \cup \{T+1\} \tag{5.54}$$

$$x_{it}, I_{it} \geq 0,\ I_{i0} = 0 \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.55}$$

$$z_t \in \{0,1\} \qquad t \in \mathcal{T} \tag{5.56}$$

$$z_{it}, f_{it} \in \{0,1\} \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{5.57}$$

Recall that we assume $\omega = \langle 1, 2, \ldots, N \rangle$. Constraints (5.50) and (5.51) link the last product in time bucket t $f_{i,t+1}$ (i.e., the first product in time bucket $t+1$) with the first product in t f_{it} and the major setup decision z_t. If there is no major setup ($z_t = 0$), the last product can only be a product after i in the sequence ω. In other words, when there is no major setup occurred in t, i might be last product in t only if a product j before i is the first product in t, which is stated by (5.50). If there is a major setup ($z_t = 1$), the last product can only be a product before i in the sequence ω to trigger a major setup, which is stated by (5.51). Constraints (5.52) and (5.53) link the product setups z_{it} with

the first setup product f_{it}, the last setup product $f_{i,t+1}$ and the major setup decision z_t. If there is no major setup ($z_t = 0$), product i can have a setup only if it is between the first product and the last product, which is ensured by constraints (5.52). If there is a major setup ($z_t = 1$) and the first product is j, then all products in ω after j can be included in the setup sequence. If there is a major setup ($z_t = 1$) and the last product is j, then all products in ω products from 1 to j can be included in the setup sequence. In other words, if there is a major setup in t, product i can be setup if it is after the first product or before the last product in the ω. This is ensured by constraints (5.53). Other constraints are explained as before.

5.5 Column Generation Approach

In this section, we develop a heuristic algorithm for the problem CLSP-FS1. This algorithm is based on the formulation AG-SO (5.10) - (5.17). Due to the exponential number of variables, we use column generation to solve the LP relaxation. Then the problem is solved as MIP with generated columns to obtain a feasible solution to CLSP-FS1.

Given subset $\mathcal{S}' \subseteq \mathcal{S}$, the master problem and its dual problem are defined as follows:

$$
\begin{aligned}
(FS^M(\mathcal{S}')) \quad \min \quad & \sum_{i\in\mathcal{N},t\in\mathcal{T}} hc_{it}I_{it} + \sum_{s\in\mathcal{S}',t\in\mathcal{T}} sc(s)w_{st} & & \\
s.t. \quad & I_{i,t-1} + x_{it} = I_{it} + d_{it} & & i\in\mathcal{N}, t\in\mathcal{T} \\
& \sum_{i\in\mathcal{N}} pt_i x_{it} + \sum_{s\in\mathcal{S}'} st(s)w_{st} \le cap_t & & t\in\mathcal{T} \\
& x_{it} \le b_{it} \sum_{s\in\mathcal{S}':i\in s} w_{st} & & i\in\mathcal{N}, t\in\mathcal{T} \\
& \sum_{s\in\mathcal{S}'} w_{st} = 1 & & t\in\mathcal{T} \\
& \sum_{s\in\mathcal{S}':s_1=i} w_{s,t+1} = \sum_{s\in\mathcal{S}':s_{L(s)}=i} w_{st} & & i\in\mathcal{N}, t\in\mathcal{T}\backslash\{T\} \\
& x_{it}, I_{it} \ge 0,\ I_{i0} = 0 & & i\in\mathcal{N}, t\in\mathcal{T} \\
& w_{st} \in \{0,1\} & & s\in\mathcal{S}', t\in\mathcal{T}
\end{aligned}
$$

$$
\begin{aligned}
(FS^D(\mathcal{S}')) \quad \min \quad & \sum_{i\in\mathcal{N},t\in\mathcal{T}} d_{it}\alpha_{it} + \sum_{t\in\mathcal{T}} cap_t\beta_t + \sum_{t\in\mathcal{T}} \sigma_t & & \\
s.t. \quad & \alpha_{it} + pt_i\beta_t + \gamma_{it} \le 0 & & i\in\mathcal{N}, t\in\mathcal{T}
\end{aligned}
$$

$$
\begin{aligned}
& -\alpha_{it} + \alpha_{i,t+1} \leq hc_{it} && i \in \mathcal{N}, t \in \mathcal{T} \\
& st(s)\beta_t - \sum_{i \in s} b_{it}\gamma_{it} + \sigma_t + \zeta_{s_{L(s)},t} - \zeta_{s_1,t-1} \leq sc(s) && s \in \mathcal{S}', t \in \mathcal{T} \\
& \beta_t \leq 0 && t \in \mathcal{T} \\
& \gamma_{it} \leq 0 && i \in \mathcal{N}, t \in \mathcal{T}
\end{aligned}
$$

The pricing problem is to use reduced cost to price out a new setup pattern $s \in \mathcal{S}$ to be able to add to the master problem. It is actually the graph presented with AG-PO formulation considering only one time bucket. Let graph $G' = (V', E')$, where V' is the vertex set defined as $\{v_{ij} : i \in \mathcal{N}, j \in \mathcal{N} \cup \{0\}\} \cup \{P, Q\}$, and E' is the edge set defined as $E' = E'^1 \cup E'^2 \cup E'^4$. $E'^1 = \{(P, v_{i0}) : i \in \mathcal{N}\}$, $E'^2 = \{(v_{ij}, v_{ik}) : i \in \mathcal{N}, j < k \in \mathcal{N}\}$ and $E'^4 = \{(v_{ij}, Q) : i, j \in \mathcal{N}\}$. The produce presented by each node and the setup cost/time for each edge are also defined as (5.18), (5.19). The network representation of the pricing problem example with 3 products is given in the Figure 5.8.

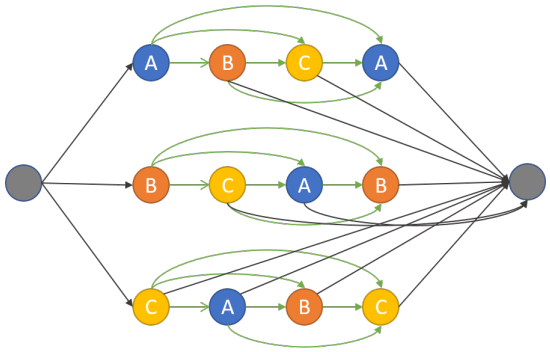


Figure 5.8: CLSP-FS1 network representation of the pricing problem

As we have shown in Section 5.3, the path from source P to Q will form a setup sequence, in this case, a valid setup sequence for one time bucket. To formulate the pricing problem, we introduce the following variables for $(u, v) \in E'$:

- $T_{uv} \in \{0, 1\}$: it equals to 1 if the edge (u, v) is selected, 0 otherwise.
- $z_i \in \{0, 1\}$: it equals to 1 if there exists an edge (u, v) is selected such that $p(u) = i$ or $p(v) = i$.

Given an optimal solution of dual problem $(\alpha^*, \beta^*, \gamma^*, \sigma^*, \zeta^*)$, we have the pricing

problem defined as follows:

$$(FS^P(\mathcal{S}',t)) \quad \max \quad \sum_{(u,v)\in E'} c_{uv}T_{uv} + \sum_{i\in\mathcal{N}} c_i z_i + \sigma_t^* \tag{5.58}$$

$$s.t. \quad \sum_{(P,v)\in E'} T_{Pv} = 1 \tag{5.59}$$

$$\sum_{(v,Q)\in E'} T_{vQ} = 1 \tag{5.60}$$

$$\sum_{u\in V':(u,v)\in E'} T_{uv} = \sum_{u\in V':(v,u)\in E'} T_{vu} \quad v \in V'\backslash\{P,Q\} \tag{5.61}$$

$$z_i \le \sum_{(u,v)\in E'(i)} T_{uv} \quad i \in \mathcal{N} \tag{5.62}$$

$$T_{uv} \in \{0,1\} \quad (u,v) \in E' \tag{5.63}$$

$$z_i \in \{0,1\} \quad i \in \mathcal{N} \tag{5.64}$$

where the cost is defined as follows:

$$c_i = -b_{it}\gamma_{it}^* \tag{5.65}$$

$$c_{uv} = \begin{cases} -\zeta^*_{p(v),t-1} & (u,v) \in E^1 \\ st(u,v)\beta_t^* - sc(u,v) & (u,v) \in E^2 \\ \zeta^*_{p(u),t} & (u,v) \in E^4 \end{cases}$$

If the optimal objective function value of above pricing problem is positive, the sequence defined by the optimal solution should be added into the master problem. We solve the LP relaxation of the master problem with newly added columns and repeat this process until no column is pricing out. This implies that the LP relaxation of the master problem is optimally solved. Based on generated columns, we solve the MIP model of the master problem to obtain a feasible solution to the original problem. The column generation heuristic is formally defined in Algorithm 9.

Algorithm 9: Column generation heuristic

1 Initialize the set $\mathcal{S}'$ with the initial heuristic solution;
2 Solve the $FS^M(\mathcal{S}')$ by column generation, and update set $\mathcal{S}'$ by adding the generated columns (setup sequences);
3 Solve $FS^M(\mathcal{S}')$ with an MIP solver by considering the subset of variables $w_{st},\ s \in \mathcal{S}'$.

5.6 Computational Results

CLSP-FS1 Computational Results

We first compare four formulations that we have developed: AG-SO, AG-PO, FL-SO, FL-PO. The benchmark instances consists of 10 instances from [63], with only the first 10 products and 10 time buckets. The summary result is given in Table 5.3, while detailed results for each instance is given in Table A.12 - A.15.

For the MIP formulation, we present the objective function value (MIP/Obj), the computational time (MIP/Time), the exit gap when CPLEX terminates (E.Gap), the relative gap comparing to the best known lower bound defined as $\frac{Obj-BestKnownLB}{Obj}$ (R.Gap), the number of columns (Cols), the number of binary variables (Bin), the number of constraints (Rows) and number of explored nodes for each formulation (Nodes). Moreover, for the LP relaxation of each MIP formulation, the optimal objective function value (LP/Obj) and its computational time (LP/Time) are also presented.

Table 5.3: Computational results: CLSP-FS1 formulation comparison

	MIP									LP	
Form.	Obj	Time	E.Gap(%)	R.Gap(%)	LB	Cols	Bin	Rows	Nodes	Obj	Time
AG-PO	42,235	569	2.8	1.7	41,038	6,710	6,510	1,312	133,075	9,038	0.1
AG-SO	42,817	600	12.2	3.1	37,582	102,500	102,300	310	15,622	11,486	2.0
FL-PO	42,118	524	1.5	1.4	41,498	7,060	6,510	1,762	132,359	37,138	0.3
FL-SO	42,252	600	4.1	1.7	40,496	102,850	102,300	760	27,175	39,991	8.3

According to the average results over 10 tested instances, we observe that
1) Formulation FL-PO has the best solution quality, which gives the lowest average relative gap as 1.4%. On the other hand, formulation AG-SO shows the worst solution quality that its average relative gap is 3.1%, which is largest among all four formulations.
2) Formulation FL-PO also has the shortest computational time comparing to other formulations.
3) Although the exit gap and the relative gap are consistent for all formulations, we still observe that there is a difference between them. For instance, the formulation AG-SO has average exit gap as large as 12.2% whereas its relative gap is 3.1%. It implies that this formulation obtains better solution quality than what its optimality gap has shown.
4) For the lower bound given by the LP relaxation, formulation FL-SO gives the best bound while formulation AG-PO gives the worst lower bound. There seems to be a dominance relationship of the formulations strength in terms of lower bounds given by the LP relaxations. However, it is not proven yet.

Next, we present our column generation algorithm based on the formulation AG-SO in Table 5.4. One observation based on our experiments is that the column generation heuristic gives average gap of 12.9%, which is worse than solving the problem directly by a standard MIP solver CPLEX. This is mainly due to the lower bound given by the linear relaxation is not strong enough based on AG-SO algorithm. This is because that when capacity is not tight, and we just need to generate all the single-product sequence $\langle i \rangle$ for $i \in \mathcal{N}$. The linear relaxation will chose a fraction of each singe-product sequence so that the total setup cost is zero and many products can be produced.

Table 5.4: Computational results: column generation heuristic based on AG-SO

Inst	Obj	Time	R.Gap (%)	Generated Cols	Pricing time	Master time	MIP time
A1	47742	23	12.4	600	22	0.1	1
A2	44705	22	9.9	603	20	0.1	2
A3	50550	22	12.2	591	20	0.2	1
A4	43576	19	14.9	568	18	0.1	1
A5	49064	21	13.6	575	19	0.1	1
A6	48396	21	14.9	575	19	0.1	1
A7	47776	22	13.5	576	20	0.1	1
A8	48615	22	13.3	572	20	0.1	1
A9	49797	21	12.5	580	19	0.1	1
A10	43286	20	11.5	584	18	0.1	1
AVG	47351	21	12.9	582	20	0.1	1

CLSP-FS1-LT Computational Results

To compare different formulations on CLSP-FS1-LT, we also generate instances based on the above tested instances. For each instance, we modify the setup values as follows to transform CLSP-FS1 instances into CLSP-FS1-LT instances:

1. $\omega = \langle 1, 2, \ldots, N \rangle$
2. $\Delta_{st} = \lfloor \frac{cap_{max}}{2} \rfloor + 1$ where $cap_{max} = \max_{t \in \mathcal{T}} cap_t$
3. $\Delta_{sc} = \frac{\sum_{i,j \in \mathcal{N}} sc_{ij}}{N^2}$
4. $st_{ij} = \Delta_{st}$ if $i > j$, 0 otherwise.
5. $sc_{ij} = \Delta_{sc}$ if $i > j$, 0 otherwise.

Moreover, in such instance, at most one nonzero setup can be performed in each time bucket. Therefore, we can also use the MIP formulation for CLSP with sequence dependent setup to solve the instances. According to the results presented in [63], we choose

the product-oriented formulation with single commodity flow, which is represented as SCM.

In Table 5.5, we present the formulation comparison results for all eight formulations by combining 2 lot sizing formulations (AG and FL) and 4 sequencing formulation (LT, PO, SO and SCM) where LT represents for the tailored formulation of CLSP-FS1-LT. Detailed results are given in Table A.16 - A.23.

Table 5.5: Computational results: CLSP-FS1-LT formulation comparison

	MIP									LP	
Form.	Obj	Time	E.Gap(%)	R.Gap(%)	LB	Cols	Bin	Rows	Nodes	Obj	Time
AG-LT	10,623	0.3	0.00	0.00	10,623	420	220	621	278	1,243	0.0
AG-PO	10,623	56.4	0.01	0.00	10,623	6,710	6,510	1,312	29,993	882	0.1
AG-SO	10,630	253.6	2.68	0.05	10,291	102,500	102,300	310	17,703	1,245	2.0
AG-SCM	10,623	3.9	0.00	0.00	10,623	2,510	1,210	1,731	3,914	0	0.0
FL-LT	10,623	0.2	0.00	0.00	10,623	770	220	1,071	-	6,586	0.0
FL-PO	10,623	145.6	0.30	0.00	10,588	7,060	6,510	1,762	53,507	6,419	0.3
FL-SO	10,623	29.3	0.00	0.00	10,623	102,850	102,300	760	775	6,586	2.2
FL-SCM	10,623	3.0	0.00	0.00	10,623	2,860	1,210	2,181	2,219	0	0.0

We have following observations:

1) Out of 8 formulations, 6 of them solve all the tested instances to prove optimality. In fact, the maximum relative gap we obtained is less than 0.01%. Therefore, the problem CLSP-FS1-LT seems easier to solve than the general version CLSP-FS1.

2) Overall, the FL formulations provide better solution quality than the AG formulations on tested instances. Based on te relative gap, all of the FL formulations have solved tested instances to optimality (may not be proven by the corresponding formulation), while one AG formulation (AG-SO) fails to solve 2 instances to optimality.

3) We measure the formulation performance based on the lexicographical order of relative gap, exit gap and computational time. Then for the AG formulations, we have formulation AG-LT, AG-SCM, AG-PO and AG-SO in the order of decreasing performances. For the FL formulations, we have FL-LT, FL-SCM, FL-SO and FL-PO in the order of decreasing performance. In other words, the LT formulation always gives best performance as expected. Moreover, the SCM formulation also gives the second best performance in our experiments. This might be due to the smaller size of the SCM formulation, which has less binary variables comparing to the PO and SO formulations. In summary, for the special case CLSP-FS1-LT problem, the tailored formulation LT is most promising among all tested formulations. Moreover, the SCM formulation, which

was developed for general CLSP with sequence dependent setups, can also solves CLSP-FS1-LT efficiently in comparison to our developed PO and SO formulation.

5.7 Conclusions

In this chapter, we have studied a restricted model of CLSP with sequence dependent setup. The problem reduces the number of candidate setup sequences from $O(n!)$ to $O(n2^n)$. However, the sequencing decision is still exponential and the problem is proven to be strongly NP-hard. We have studied two types of formulations including exponential size sequence-oriented formulation and polynomial size product-oriented formulation. Moreover, both formulations are reformulated according to the facility location based model. We perform preliminary experiments on developed formulations and column generation heuristic. Formulation FL-SO gives the overall best lower bound from LP relaxation and FL-PO gives the best MIP solution quality.

A special case is studied, which considers only two setup values. However, it still captures the essence of our problem with a major setup and a minor setup. It is proven to be NP-hard as well. Due to the special structure, we simplify the MIP formulation for this special case, and comparing to the product-oriented formulation for the general problem, the number of binary variables is reduced from $O(N^3T)$ to $O(NT)$.

However, more experiments need to be done to further evaluate formulation performance, and more efficient heuristic algorithms are to be designed.

Chapter 6

General Conclusion and Future Work

In this manuscript, we have presented our research motivated by challenging real-world applications. We can summarize the main results in two areas.

In the first part of the manuscript, a challenging production planning problem brought to our attention by an apparel manufacturing project is studied. We designed an optimized software tool to efficiently tackle this industrial problem. A decomposition framework has been developed, which solves an aggregated model and a detailed model in sequence. The aggregated problem, called for brevity CLSC, is shown to be the bottleneck of the approach and it has been studied from different perspectives. CLSC corresponds to a complex capacitated lot sizing problem, and it has been shown to be NP-hard even without the setup costs. Several Mixed Integer Programming (MIP) formulations are developed for CLSC. To computationally evaluate the different MIP formulations, two sets of benchmark instances have been designed. The first set consists of realistic data while the second set consists of pseudo-randomly generated instances with realistic characteristics and different features. Thanks to extensive computational tests, we have identified that one formulation computationally outperforms the others. Average-size instances can be solved directly using CPLEX, but large scale instances cannot be solved to proven optimality within short computational time. Therefore, several effective heuristic algorithms are developed based on constructive phases and enhanced by local search phases. We designed a Fix & Relax (F&R) heuristic algorithm based on the Linear Programming (LP) relaxation of the compact formulations. This heuristic provides good quality solutions but it requires long computational times. Then, in

order to obtain good quality solutions in short computational time, we designed a Product Decomposition (PD) algorithm based on the observation that 20% of the product families covers 80% of the demands (for realistic instances). We experienced a natural trade-off of solution quality and computational time when comparing the performances of F&R and PD. In addition, a constructive heuristic is developed and is called First Solution Heuristic (FSH). The FSH algorithm is based on the LP relaxation of the compact model and variable fixing with the goal of constructing good quality feasible solutions. Thanks to extensive computational tests, we observed that the PD algorithm outperforms the FSH algorithm in terms of computational time and solution quality (for the tested benchmark instances). However, we observed that the positive effect of the local search phase is stronger for the FSH algorithm than for the PD algorithm. Finally, combining FSH and F&O allows us to achieve overall best performance. For real-world instances, a maximum optimality gap of 15% is observed between the feasible solutions and the LP relaxation values. These results outperforms the optimality gap of CPLEX directly applied to the compact formulations, which is greater than 30% on average considering within similar time limit of 1 hour CPU time. As far as the randomly generated benchmarks are concerned, if we compare the solution of CPLEX and the ones of the algorithm FSH + F&O, we observed an improvement in their quality of 85%. As far as the other heuristics are concerned, they also outperform CPLEX in computing feasible solutions within short computational time. All the developed heuristic algorithms have been included into the production planning tool of DecisionBrain improving in this way the efficiency of the optimization system.

In the second part the manuscript, we studied a restricted version of the capacitated lot sizing problem with sequence dependent setups, where the setup sequences for each time bucket have to follow the order of a given sequence. This problem is called capacitated lot sizing problem with a fixed product sequence (CLSP-FS1). Also this problem comes from a real-world application. Compared to the capacitated lot sizing problem with sequence dependent setups, CLSP-FS1 reduces the number of candidate sequences from $O(n!)$ to $O(n2^n)$. In many real-world applications, an "ideal" sequence is known and only sequences following that order can be chosen. This problem is shown to be NP-hard and four MIP models are developed based on sequence-oriented and product-oriented (setup) formulations. We performed preliminary computational tests to compare these formulations to a classical reformulation. We observed that one newly proposed formulation guarantees the best performance overall for the tested benchmark instances. For the sequence-oriented (setup) formulation, a simple column generation heuristic has been developed and tested. Even if the quality of the LP relaxation bound is

superior to its counter part of the compact formulations, the feasible solutions computed by the column generation heuristic are inferior to those computed by CPLEX applied to the compact formulation. Moreover, we studied a special case of CLSP-FS1, which has a lower triangle form of setup matrices and is called CLSP-FS1-LT. For this variant of the problem, if the production plan follows the order of the given sequence, then no setup incurs. However, if the production plan reverses the order of products in the given sequence, a big setup has to be paid. Hence, what has to be decided is when to perform the major (nonzero) setup. Also, this problem variant is shown to be NP-hard and a tailored MIP formulation is developed. Comparing to the product-oriented formulation with $O(N^3T)$ binary variables, the tailored MIP formulation of CLSP-FS1-LT has only $O(NT)$ binary variables.

Summarizing this thesis, we tacked challenging production planning problems and we designed advanced mathematical models and effective heuristic algorithms. These tools allow us to compute good quality feasible solution, however, several possible future lines of research remain open.

Regarding the CLSC problem, we observed a large optimality gap larger than 50% on difficult benchmark instances IRG-B. Therefore, other families of heuristic algorithms can be developed possibly based on stronger formulations. More in details, a promising line of research can be the study of the network reformulation of the CLSC, or a hybrid MIP formulation could also be developed based on recently proposed in the literature.

With regards to the problem CLSP-FS1, our study offers a first glance into this problem and only preliminary experiments have been conducted thus far. Therefore, we would like to test developed formulations on a larger scale instances especially on structured instances where our model applies, and they should be compared with the classical sequence dependent model.

At last, the study presented in Chapter 2 have been published in the conference paper [48]. We have also started another project regarding Temporal Bin Pacing Problem (TBPP). It is an extension of the Bin Packing Problem, where items consume the bin capacity during a time window only. Both a polynomial-size formulation and an extensive formulation are studied. Moreover, various heuristic algorithms are developed and compared, including greedy-type heuristics and a column generation based heuristic. The study of TBPP has been published in the conference paper [49]. Since TBPP does not follow the line of production planning, we have not included it in this manuscript.

Appendix A

Data Analysis and Computational Results

A.1 CLSC Computational Results

Table A.1: CLSP-SC real-world instance R5 analysis: product feature

Product i	$\lvert\mathcal{D}^i\rvert$	UnitReqCap	ReqCaps	Percentage	AvgSetupCap	AvgSetupCost
1	7	1529	19,474,873	1.2	1,146,038	234,686
2	26	1207	8,917,432	0.5	1,031,701	211,272
3	1	1852	18,520	0.0	1,202,585	246,266
4	123	1599	324,863,641	19.8	1,016,619	208,183
5	4	1637	4,764,544	0.3	1,078,964	220,950
6	6	1894	3,678,084	0.2	1,133,420	232,102
7	15	1608	23,920,684	1.5	1,099,995	225,257
8	67	616	85,304,296	5.2	1,710,857	350,349
9	4	1952	3,733,368	0.2	1,204,352	246,627
10	65	1343	34,583,566	2.1	1,205,671	246,897
11	6	1523	722,308	0.0	1,123,788	230,130
12	6	1539	14,443,330	0.9	1,101,936	225,655
13	3	1747	13,718,184	0.8	1,030,382	211,002
14	21	806	4,238,754	0.3	1,707,149	349,590
15	9	1526	5,347,104	0.3	1,155,397	236,602
16	4	1308	673,663	0.0	1,004,075	205,615
17	3	1289	7,898,992	0.5	1,362,518	279,017
18	8	1673	25,095,000	1.5	1,159,951	237,535
19	15	1696	11,983,155	0.7	1,096,162	224,472
20	3	1295	2,101,154	0.1	1,392,260	285,107
21	4	1253	5,042,094	0.3	1,162,490	238,055
22	8	1553	9,054,286	0.6	1,741,569	356,638
23	19	1174	3,995,880	0.2	1,127,024	230,792
24	2	1967	1,384,768	0.1	1,135,386	232,504
25	1	1527	679,515	0.0	1,323,618	271,050
26	23	1898	67,254,286	4.1	1,187,801	243,238
27	16	1775	97,858,468	6.0	1,133,544	232,127
28	2	2030	2,261,951	0.1	1,210,176	247,820
29	29	2028	17,982,756	1.1	1,327,923	271,932
30	1	1015	1,023,120	0.1	1,151,738	235,853
31	8	2128	4,917,220	0.3	1,313,164	268,910
32	2	1235	1,993,290	0.1	1,052,408	215,512
33	109	1513	748,092,322	45.5	1,120,403	229,436
34	2	1960	5,064,640	0.3	1,070,079	219,131
35	11	1636	1,884,672	0.1	1,224,910	250,837
36	4	1994	16,488,386	1.0	1,129,438	231,286
37	1	1732	1,584,780	0.1	1,103,703	226,016
38	4	1777	11,865,029	0.7	1,125,257	230,430
39	2	1296	177,552	0.0	1,213,387	248,477
40	6	1792	9,533,067	0.6	1,130,135	231,429
41	4	1240	10,549,920	0.6	1,051,413	215,308
42	6	1734	12,220,270	0.7	996,260	204,014
43	2	2005	2,860,220	0.2	1,145,018	234,477
44	2	1468	3,376,944	0.2	1,141,011	233,656
45	2	2657	5,446,850	0.3	1,435,590	293,980
46	2	1431	5,276,868	0.3	1,448,209	296,564

Table A.2: CLSP-SC real-world instance R5 analysis: demand feature

TimeBucket	ReleaseDate	FirstDueDate	SecondDueDate
1	38	24	0
2	24	12	1
3	14	5	5
4	32	18	4
5	33	15	14
6	78	46	2
7	87	45	9
8	98	47	22
9	34	36	8
10	41	44	59
11	38	50	41
12	19	87	44
13	31	15	31
14	67	54	72
15	29	37	48
16	0	18	69
17	1	37	9
18	0	48	61
19	2	25	26
20	0	0	21
21	2	1	39
22	0	0	55
23	0	2	13
24	0	0	0
25	0	2	1

Table A.3: CLSP-SC real-world instance R5 analysis: demand feature

Length	first - release	second-first
< 0	16	4
0	56	10
1	91	1
2	55	5
3	114	181
4	303	458
5	25	0
6	7	9
7	1	0

Table A.4: CLSP-SC real-world instance R5 analysis: capacity feature

Interval	NumDemand	ReqCapRatio
[t1,t4]	7.63	1.15
[t2,t5]	4.49	0.53
[t3,t6]	6.89	0.53
[t4,t7]	11.68	0.59
[t5,t8]	14.07	0.72
[t6,t9]	14.22	0.47
[t7,t10]	9.73	0.31
[t8,t11]	7.49	0.27
[t9,t12]	4.34	0.11
[t10,t13]	1.35	0.08
[t11,t14]	2.84	0.11
[t12,t15]	2.69	0.13
[t13,t16]	2.54	0.18
[t14,t17]	3.44	0.25
[t15,t18]	0.9	0.07
[t16,t19]	0	0
[t17,t20]	0	0
[t18,t21]	0	0
[t19,t22]	0	0
[t20,t23]	0	0
[t21,t24]	0	0
[t22,t25]	0	0

Table A.5: CLSP-SC real-world instance R5 analysis: capacity feature

Machine	avgCap	minCap	maxCap	avgSetupCap	minSetupCap	maxSetupCap
r1	2,329,699	1,428,300	2,768,970	1,060,070	880,638	1,539,450
r2	2,089,478	1,273,049	2,225,250	996,465	827,800	1,447,083
r3	2,217,635	1,366,200	2,333,925	932,861	774,962	1,354,716
r4	3,487,490	2,181,780	3,643,200	1,441,695	1,197,668	2,093,652
r5	3,574,976	2,252,160	3,848,130	1,441,695	1,197,668	2,093,652
r6	3,008,788	1,821,600	3,449,655	1,314,486	1,091,991	1,908,918
r7	3,215,400	1,987,200	3,415,500	1,272,084	1,056,766	1,847,340
r8	3,334,425	2,111,400	3,622,500	1,349,179	1,120,812	1,959,300
r9	2,320,183	1,424,160	2,539,200	1,017,667	845,413	1,477,872
r10	2,584,539	1,583,550	2,925,600	1,166,077	968,702	1,693,395
r11	2,896,103	1,676,700	3,187,800	1,229,680	1,021,540	1,785,761
r12	4,079,481	2,527,469	4,340,100	1,616,446	1,342,840	2,347,428
r13	2,916,000	1,242,000	3,105,000	1,272,084	1,056,766	1,847,340
r14	1,912,162	1,117,799	2,421,900	954,063	792,575	1,385,505
r15	3,112,130	1,970,640	3,381,000	1,259,235	1,046,092	1,828,680
r16	2,426,845	1,523,520	2,784,840	1,060,070	880,638	1,539,450
r17	4,031,382	2,527,469	4,257,990	1,590,104	1,320,957	2,309,175
r18	2,188,478	1,366,200	2,539,200	1,017,667	845,413	1,477,872
r19	2,137,045	1,366,200	2,428,110	932,861	774,962	1,354,716
r20	2,516,028	1,475,910	2,815,200	1,166,077	968,702	1,693,395
r21	3,008,788	1,821,600	3,449,655	1,314,486	1,091,991	1,908,918
r22	6,557,759	6,557,759	6,557,759	3,053,000	2,536,238	4,433,616
r23	2,426,845	1,523,520	2,784,840	1,060,070	880,638	1,539,450
r24	2,804,821	1,753,290	2,998,050	1,116,607	927,605	1,621,554
r25	1,981,479	1,140,570	2,372,910	890,459	739,736	1,293,138
r26	1,390,278	870,435	1,450,725	699,646	581,221	1,016,037
r27	1,878,438	1,170,585	2,051,024	954,063	792,575	1,385,505
r28	2,392,143	1,490,399	2,860,050	1,102,472	915,864	1,601,028
r29	1,529,040	950,130	2,119,680	848,056	704,511	1,231,560
r30	1,893,877	1,229,579	2,231,460	848,056	704,511	1,231,560

Table A.6: Computational results: FSH and FO algorithm on IRG-B (1)

	FSH						FSH+FO		
Inst	Obj	Time	Gap	#Iter	#FixedTo1	MIPTime	Obj	Time	Gap
B1	569496	261	100.0	28	1.5	2	14220	2343	100.0
B2	857700	252	100.0	28	1.4	2	30347	2435	100.0
B3	1332316	238	100.0	31	1.5	2	32043	2462	100.0
B4	3154699	310	95.4	31	0.9	4	1268281	1023	88.6
B5	1627483	294	100.0	29	1.3	2	86697	2354	100.0
B6	2687665	315	100.0	27	1.2	3	389124	1811	100.0
B7	3006613	947	100.0	30	1.0	6	121681	3334	100.0
B8	2675949	893	100.0	32	1.1	5	8182	3541	100.0
B9	1925285	1040	100.0	31	0.9	6	97418	3804	100.0
B10	3301737	1081	100.0	30	0.8	7	381594	3630	100.0
B11	4475709	1242	100.0	27	0.8	7	1393145	3528	100.0
B12	3183505	1111	100.0	27	0.8	7	774781	3723	100.0
B13	1421072	3480	100.0	29	0.7	20	62269	6010	100.0
B14	1477498	3505	100.0	31	0.8	23	78058	6156	100.0
B15	1004987	3530	100.0	29	0.7	22	38147	5935	100.0
B16	120095966	1216	100.0	0	0.0	600	3726046	3658	100.0
B17	7721458	3932	99.8	29	0.5	25	4137600	6699	99.7
B18	4892235	3842	99.8	30	0.5	26	1463073	6273	99.2
AVG	9189521	1527	100	28	0.9	43	783484	3818	99.3
B19	1885162	328	100.0	30	1.7	2	52074	2738	100.0
B20	2435489	379	100.0	28	1.5	3	135957	3070	100.0
B21	3068139	389	100.0	28	1.5	3	276295	2883	100.0
B22	5863879	374	98.9	28	1.3	3	2597011	1882	97.4
B23	6120530	372	100.0	33	1.2	3	2862285	2210	100.0
B24	4924154	410	100.0	28	1.3	3	1059845	2368	100.0
B25	5299304	1531	100.0	28	1.0	9	607896	4294	100.0
B26	4432748	1492	100.0	32	1.0	7	511110	4256	100.0
B27	4074232	1349	100.0	31	1.0	7	303425	4113	100.0
B28	10067139	1371	99.8	29	0.8	7	6458050	3829	99.6
B29	11750265	1367	99.0	30	0.8	6	7363777	3290	98.5
B30	10401358	1383	99.6	29	0.8	6	6025335	3510	99.4
B31	4458564	5328	100.0	31	0.7	26	1180039	8094	100.0
B32	4216738	4989	100.0	30	0.8	23	466260	7756	100.0
B33	119181317	1216	100.0	0	0.0	601	4321073	3983	100.0
B34	13215539	4146	99.8	32	0.6	17	7946626	6913	99.7
B35	12392793	3701	98.4	28	0.6	21	8878757	6468	97.7
B36	114132867	1216	100.0	0	0.0	601	12516305	3982	100.0
AVG	18773345	1741	100	26	0.9	75	3531229	4202	99.6

Table A.7: Computational results: FSH and FO algorithm on IRG-B (2)

	FSH						FSH+FO		
Inst	Obj	Time	Gap	#Iter	#FixedTo1	MIPTime	Obj	Time	Gap
B37	983010	239	100.0	30	1.6	3	14942	2433	100.0
B38	1412074	281	100.0	31	1.4	2	56629	2657	100.0
B39	1255719	273	100.0	30	1.4	2	72866	2570	100.0
B40	3909024	312	100.0	32	1.2	2	1212755	1631	100.0
B41	1734686	296	100.0	27	1.2	3	143503	2814	100.0
B42	3277156	354	99.1	29	1.1	2	1591917	1544	98.1
B43	2011864	930	100.0	30	1.0	7	50310	3575	100.0
B44	1114294	782	100.0	31	1.1	8	2302	2973	100.0
B45	2145942	832	100.0	30	1.0	6	45059	3405	100.0
B46	5709567	1127	99.9	32	0.8	7	2508003	3118	99.8
B47	3407334	1101	100.0	30	0.8	7	1168334	3581	100.0
B48	6654652	1327	99.0	29	0.7	6	3902549	3069	98.3
B49	1790740	3516	100.0	31	0.7	22	28248	6167	100.0
B50	2460050	3310	100.0	30	0.8	20	54707	5962	100.0
B51	2398138	3833	100.0	30	0.7	21	154890	6600	100.0
B52	4685339	4049	100.0	31	0.6	26	1399951	6815	100.0
B53	5921797	3660	98.9	30	0.5	24	3055235	6306	97.9
B54	6485383	3934	98.7	28	0.5	24	4203031	6602	98.0
AVG	3186487	1675	100	30	1.0	11	1092513	3990	99.6
B55	909994	259	100.0	30	1.4	2	19346	2526	100.0
B56	816435	247	100.0	31	1.5	2	11763	2612	100.0
B57	736264	243	100.0	30	1.5	2	5208	2300	100.0
B58	2247744	331	100.0	27	1.1	3	257911	1526	100.0
B59	2298256	310	100.0	28	1.1	3	511325	1420	100.0
B60	2278970	322	99.1	26	1.1	3	578207	1340	96.3
B61	1842306	934	100.0	30	1.0	5	69887	3464	100.0
B62	2518664	1022	100.0	31	1.0	7	90160	3778	100.0
B63	2300055	1088	100.0	29	1.0	7	91142	3735	100.0
B64	5192475	1142	100.0	29	0.7	9	2084179	3611	100.0
B65	5255345	1204	100.0	29	0.8	8	2548560	3623	100.0
B66	4610617	1031	99.9	27	0.7	8	1321694	3494	99.7
B67	1937528	3542	100.0	29	0.7	22	62085	6089	100.0
B68	1845334	3469	100.0	29	0.7	27	82770	6066	100.0
B69	2948951	3554	100.0	33	0.7	21	83002	6207	100.0
B70	4781314	4690	100.0	30	0.6	31	1880378	7456	100.0
B71	115751239	1216	100.0	0	0.0	600	4244626	3983	100.0
B72	5652817	3296	100.0	28	0.5	27	1881746	5895	99.9
AVG	9106906	1550	100	28	0.9	44	879111	3840	99.8

Table A.8: Computational results: FSH and FO algorithm on IRG-B (3)

	FSH						FSH+FO		
Inst	Obj	Time	Gap	#Iter	#FixedTo1	MIPTime	Obj	Time	Gap
B73	1027	66	100.0	44	2.2	4	0	77	0.0
B74	2166	71	100.0	42	2.3	3	0	82	0.0
B75	5079	113	100.0	44	2.2	4	0	125	0.0
B76	118275	276	100.0	37	1.6	2	69647	2414	100.0
B77	1290034	341	93.2	30	1.2	3	238513	1516	63.5
B78	626638	276	78.8	30	1.3	2	230945	2043	42.4
B79	11036	135	100.0	45	1.5	9	0	200	0.0
B80	19799	544	100.0	39	1.4	11	0	763	0.0
B81	3116	446	100.0	41	1.4	10	0	576	0.0
B82	495908	952	100.0	33	1.1	6	4519	3628	100.0
B83	1431220	1028	89.8	29	0.8	7	398953	3171	63.4
B84	110817	910	100.0	35	1.1	6	3169	3403	100.0
B85	4838	2013	100.0	43	1.1	26	0	2103	0.0
B86	5088	680	100.0	42	1.1	22	0	769	0.0
B87	2812	2249	100.0	47	1.1	25	0	2359	0.0
B88	13799	3327	100.0	37	0.8	21	0	5361	0.0
B89	207862	4165	100.0	49	0.8	20	66902	6813	100.0
B90	78039	3991	100.0	38	0.7	25	17031	6757	100.0
AVG	245975	1199	98	39	1.3	11	57204	2342	37.2
B91	870570	213	100.0	29	1.5	2	0	1181	0.0
B92	460278	233	100.0	32	1.6	2	5755	1415	100.0
B93	776054	236	100.0	29	1.5	2	166	1812	100.0
B94	1624274	310	100.0	28	1.2	3	121190	2695	100.0
B95	2211051	319	100.0	26	1.2	3	152603	2855	100.0
B96	2455140	338	100.0	26	1.1	3	544955	2185	100.0
B97	1625407	803	100.0	30	1.1	6	842	1929	100.0
B98	2150486	966	100.0	31	1.0	4	6905	3730	100.0
B99	1720532	936	100.0	30	1.0	6	29277	3596	100.0
B100	6485075	1142	100.0	30	0.7	7	3623037	3818	100.0
B101	9240132	1262	97.7	27	0.7	7	6400344	2725	96.7
B102	3928379	1127	100.0	28	0.8	8	789585	3814	100.0
B103	1436398	3410	100.0	31	0.8	18	3155	5985	100.0
B104	2097677	3616	100.0	30	0.7	19	35075	6383	100.0
B105	1882524	3606	100.0	30	0.7	19	43186	6373	100.0
B106	123496124	1215	100.0	0	0.0	600	7824869	3982	100.0
B107	4693908	4228	100.0	30	0.6	23	1671870	6994	100.0
B108	3453669	4800	100.0	29	0.6	25	911956	7566	100.0
AVG	9478204	1598	100	28	0.9	42	1231376	3835	94.3

Table A.9: Computational results: PD and FO algorithm on IRG-B (1)

	PD			PD+FO		
Inst	Obj	Time	Gap	Obj	Time	Gap
B1	164824	48	100.0	41083	1027	100.0
B2	157580	82	100.0	42608	1041	100.0
B3	145923	43	100.0	77369	893	100.0
B4	3378237	16	95.7	1802414	399	92.0
B5	668287	39	100.0	226697	1133	100.0
B6	2563782	110	100.0	700792	634	100.0
B7	1113790	154	100.0	265497	1254	100.0
B8	200636	204	100.0	101699	937	100.0
B9	229043	77	100.0	142183	1244	100.0
B10	1777419	217	100.0	651238	1344	100.0
B11	4083545	95	100.0	2049581	1145	100.0
B12	3693503	120	100.0	1900927	780	100.0
B13	223826	314	100.0	115884	1474	100.0
B14	258130	454	100.0	117655	1487	100.0
B15	284868	406	100.0	142262	1520	100.0
B16	3820282	447	100.0	1863076	1823	100.0
B17	8876278	369	99.9	6655287	1713	99.8
B18	4501086	426	99.7	2831314	1703	99.6
AVG	2007836	201	99.7	1095976	1197	99.5
B19	278413	81	100.0	123732	1007	100.0
B20	497369	176	100.0	189411	1426	100.0
B21	2019686	155	100.0	378180	1236	100.0
B22	6064520	40	98.9	2909949	354	97.7
B23	7991949	92	100.0	3374718	851	100.0
B24	4275062	82	100.0	1440811	1025	100.0
B25	3925330	434	100.0	752207	1765	100.0
B26	2719359	257	100.0	692292	1525	100.0
B27	889276	331	100.0	386731	1605	100.0
B28	12551842	288	99.8	6653551	1534	99.6
B29	13123882	84	99.1	8404438	874	98.7
B30	13000493	321	99.7	6837244	1461	99.5
B31	1660838	729	100.0	1007841	2072	100.0
B32	667081	742	100.0	480884	1983	100.0
B33	2861996	1166	100.0	1128450	2463	100.0
B34	16566442	496	99.8	10165015	1853	99.7
B35	15187035	613	98.7	10143799	1953	98.0
B36	14542025	754	100.0	9856305	2004	100.0
AVG	6601255	380	99.8	3606976	1499	99.6

Table A.10: Computational results: PD and FO algorithm on IRG-B (2)

	PD			PD+FO		
Inst	Obj	Time	Gap	Obj	Time	Gap
B37	71152	69	100.0	18614	914	100.0
B38	265139	75	100.0	135173	785	100.0
B39	200942	40	100.0	104088	750	100.0
B40	3390405	53	100.0	1966819	453	100.0
B41	301904	51	100.0	203349	977	100.0
B42	4235634	57	99.3	2559574	485	98.8
B43	201591	190	100.0	92852	1295	100.0
B44	190307	80	100.0	59365	944	100.0
B45	203421	85	100.0	88307	1095	100.0
B46	5778783	82	99.9	3664870	1205	99.8
B47	3072701	87	100.0	1661221	1033	100.0
B48	9289926	107	99.3	6260173	1242	98.9
B49	203402	394	100.0	77507	1534	100.0
B50	313893	255	100.0	213961	1338	100.0
B51	382684	490	100.0	238909	1723	100.0
B52	4759802	263	100.0	2589047	1469	100.0
B53	6199666	469	98.9	3786746	1691	98.3
B54	8327873	323	99.0	5326919	1698	98.4
AVG	2632735	176	99.8	1613750	1146	99.7
B55	145649	128	100.0	66342	820	100.0
B56	203327	115	100.0	73563	982	100.0
B57	115540	60	100.0	30857	888	100.0
B58	1743807	37	100.0	416272	758	100.0
B59	2797638	206	100.0	1422255	580	100.0
B60	3365225	54	99.4	1301320	712	98.4
B61	277617	277	100.0	169776	1311	100.0
B62	384096	301	100.0	244576	1389	100.0
B63	307630	140	100.0	155711	1236	100.0
B64	5408696	157	100.0	3203130	806	100.0
B65	6551910	123	100.0	3533088	777	100.0
B66	4671602	244	99.9	2692260	1127	99.9
B67	255589	491	100.0	146812	1591	100.0
B68	264806	398	100.0	125522	1525	100.0
B69	338952	488	100.0	139664	1618	100.0
B70	4163398	542	100.0	2015908	1892	100.0
B71	4576418	614	100.0	2142634	2000	100.0
B72	7051053	411	100.0	4260084	1632	100.0
AVG	2367942	266	100.0	1229987	1203	99.9

Table A.11: Computational results: PD and FO algorithm on IRG-B (3)

	PD			PD+FO		
Inst	Obj	Time	Gap	Obj	Time	Gap
B73	13650	34	100.0	0	44	0.0
B74	23520	59	100.0	0	70	0.0
B75	163447	59	100.0	0	555	0.0
B76	341374	59	100.0	184993	586	100.0
B77	4020239	40	97.8	1373028	831	93.7
B78	569385	44	76.7	339286	985	60.8
B79	143936	116	100.0	0	163	0.0
B80	202415	123	100.0	4840	761	100.0
B81	94001	284	100.0	0	323	0.0
B82	3860227	92	100.0	682297	1128	100.0
B83	3806156	227	96.2	1452931	1301	89.9
B84	2270256	148	100.0	242469	1181	100.0
B85	19310	417	100.0	0	475	0.0
B86	248104	454	100.0	90	905	100.0
B87	144974	520	100.0	0	726	0.0
B88	521868	441	100.0	168839	1434	100.0
B89	614865	413	100.0	321989	1552	100.0
B90	580113	347	100.0	303851	1509	100.0
AVG	979880	215	98.4	281923	807	58.0
B91	74582	105	100.0	410	519	100.0
B92	11195	180	100.0	0	243	0.0
B93	84762	58	100.0	3	910	100.0
B94	1472665	84	100.0	176225	1186	100.0
B95	2897150	89	100.0	579829	1064	100.0
B96	3598958	48	100.0	1059709	794	100.0
B97	7120	286	100.0	0	327	0.0
B98	17238	265	100.0	427	543	100.0
B99	38003	201	100.0	1098	962	100.0
B100	7808451	199	100.0	4190866	1021	100.0
B101	9554091	234	97.8	6836055	1060	96.9
B102	3977100	182	100.0	1718978	1477	100.0
B103	5691	639	100.0	1074	1032	100.0
B104	51029	472	100.0	3601	1448	100.0
B105	37558	557	100.0	204	797	100.0
B106	6150546	454	100.0	2835681	1840	100.0
B107	3937592	325	100.0	1238442	1711	100.0
B108	2904031	437	100.0	969104	1824	100.0
AVG	2368209	268	99.9	1089539	1042	88.7

A.2 CLSP-FS1 Computational Results

Table A.12: Computational results of CLSP-FS1 formulation: AG-SO

	MIP									LP	
Inst.	Obj	Time	E.Gap(%)	R.Gap(%)	LB	Cols	Bin	Rows	Nodes	Obj	Time
A1	43,411	600	9.68	2.32	39,210	102,500	102,300	310	14,500	11,262	2.1
A2	41,612	600	14.03	3.17	35,772	102,500	102,300	310	14,298	11,494	2.1
A3	46,398	600	11.59	3.78	41,021	102,500	102,300	310	17,652	11,977	2.7
A4	38,081	600	12.13	2.67	33,461	102,500	102,300	310	6,357	11,186	1.7
A5	44,182	600	12.52	3.77	38,650	102,500	102,300	310	15,095	11,817	2.6
A6	43,659	600	15.04	2.02	37,093	102,500	102,300	310	13,273	11,697	2.0
A7	42,899	600	13.47	2.99	37,120	102,500	102,300	310	16,594	11,339	1.8
A8	43,495	600	13.96	4.73	37,423	102,500	102,300	310	20,133	11,567	1.6
A9	45,056	600	8.26	2.31	41,334	102,500	102,300	310	22,191	11,654	2.0
A10	39,377	600	11.79	2.85	34,736	102,500	102,300	310	16,130	10,869	1.8
AVG	42,817	600	12.25	3.06	37,582	102,500	102,300	310	15,622	11,486	2.0

Table A.13: Computational results of CLSP-FS1 formulation: AG-PO

	MIP									LP	
Inst.	Obj	Time	E.Gap(%)	R.Gap(%)	LB	Cols	Bin	Rows	Nodes	Obj	Time
A1	42,873	600	2.65	1.10	41,735	6,710	6,510	1,312	136,294	8,742	0.1
A2	40,751	600	2.94	1.12	39,552	6,710	6,510	1,312	130,389	9,069	0.1
A3	45,514	600	3.39	1.91	43,973	6,710	6,510	1,312	149,275	9,456	0.1
A4	37,069	286	0.01	0.01	37,065	6,710	6,510	1,312	64,068	8,609	0.1
A5	43,231	600	2.45	1.66	42,170	6,710	6,510	1,312	115,580	9,315	0.1
A6	43,023	600	2.36	0.57	42,008	6,710	6,510	1,312	147,601	9,301	0.1
A7	42,274	600	2.50	1.55	41,217	6,710	6,510	1,312	138,486	9,024	0.1
A8	43,030	600	3.70	3.70	41,439	6,710	6,510	1,312	125,851	9,079	0.1
A9	44,982	600	3.20	2.15	43,544	6,710	6,510	1,312	168,123	9,253	0.1
A10	39,599	600	4.85	3.40	37,680	6,710	6,510	1,312	155,080	8,530	0.1
AVG	42,235	569	2.80	1.72	41,038	6,710	6,510	1,312	133,075	9,038	0.1

Table A.14: Computational results of CLSP-FS1 formulation: FL-SO

	MIP									LP	
Inst.	Obj	Time	E.Gap(%)	R.Gap(%)	LB	Cols	Bin	Rows	Nodes	Obj	Time
A1	42,960	600	3.41	1.30	41,493	102,850	102,300	760	29,400	41,162	7.5
A2	40,504	600	3.42	0.52	39,118	102,850	102,300	760	34,560	38,542	4.8
A3	45,904	600	5.37	2.74	43,441	102,850	102,300	760	21,408	43,044	9.9
A4	37,069	600	0.80	0.01	36,771	102,850	102,300	760	36,018	35,532	6.0
A5	43,231	600	3.97	1.66	41,514	102,850	102,300	760	22,567	41,202	9.5
A6	43,363	600	5.04	1.35	41,177	102,850	102,300	760	21,394	40,738	10.3
A7	42,152	600	4.56	1.27	40,231	102,850	102,300	760	31,603	39,711	9.2
A8	42,820	600	4.64	3.23	40,835	102,850	102,300	760	25,358	40,261	9.1
A9	44,854	600	4.15	1.87	42,991	102,850	102,300	760	24,064	42,665	9.1
A10	39,665	600	5.74	3.56	37,388	102,850	102,300	760	25,373	37,057	8.1
AVG	42,252	600	4.11	1.75	40,496	102,850	102,300	760	27,175	39,991	8.3

Table A.15: Computational results of CLSP-FS1 formulation: FL-PO

	MIP									LP	
Inst.	Obj	Time	E.Gap(%)	R.Gap(%)	LB	Cols	Bin	Rows	Nodes	Obj	Time
A1	42,658	600	0.60	0.60	42,403	7,060	6,510	1,762	147,216	38,244	0.3
A2	40,299	366	0.01	0.01	40,295	7,060	6,510	1,762	92,967	36,006	0.3
A3	45,557	600	2.00	2.00	44,644	7,060	6,510	1,762	154,849	40,090	0.3
A4	37,069	114	0.01	0.01	37,066	7,060	6,510	1,762	27,597	32,316	0.3
A5	43,231	600	1.66	1.66	42,515	7,060	6,510	1,762	169,998	38,403	0.3
A6	42,782	564	0.01	0.01	42,778	7,060	6,510	1,762	153,500	37,910	0.3
A7	42,232	600	1.45	1.45	41,618	7,060	6,510	1,762	155,701	36,977	0.3
A8	42,820	600	3.34	3.23	41,388	7,060	6,510	1,762	106,436	37,198	0.3
A9	44,854	600	1.87	1.87	44,017	7,060	6,510	1,762	145,601	39,813	0.3
A10	39,678	600	3.59	3.59	38,254	7,060	6,510	1,762	169,723	34,420	0.3
AVG	42,118	524	1.45	1.44	41,498	7,060	6,510	1,762	132,359	37,138	0.3

Table A.16: Computational results of CLSP-FS1-LT formulation: AG-LT

	MIP									LP	
Inst.	Obj	Time	E.Gap(%)	R.Gap(%)	LB	Cols	Bin	Rows	Nodes	Obj	Time
A1-LT	8,987	0.2	0.00	0.00	8,987	420	220	621	216	1,217	0.0
A2-LT	8,969	0.3	0.00	0.00	8,969	420	220	621	265	1,227	0.0
A3-LT	14,204	0.3	0.00	0.00	14,204	420	220	621	374	1,286	0.0
A4-LT	7,871	0.2	0.00	0.00	7,871	420	220	621	89	1,217	0.0
A5-LT	9,179	0.2	0.00	0.00	9,179	420	220	621	325	1,288	0.0
A6-LT	12,833	0.4	0.00	0.00	12,833	420	220	621	551	1,241	0.0
A7-LT	9,088	0.2	0.00	0.00	9,088	420	220	621	392	1,271	0.0
A8-LT	10,247	0.2	0.00	0.00	10,247	420	220	621	80	1,253	0.0
A9-LT	11,798	0.3	0.00	0.00	11,798	420	220	621	182	1,237	0.0
A10-LT	13,058	0.2	0.00	0.00	13,058	420	220	621	305	1,197	0.0
AVG	10,623	0.3	0.00	0.00	10,623	420	220	621	278	1,243	0.0

Table A.17: Computational results of CLSP-FS1-LT formulation: AG-PO

	MIP									LP	
Inst.	Obj	Time	E.Gap(%)	R.Gap(%)	LB	Cols	Bin	Rows	Nodes	Obj	Time
A1-LT	8,987	18.6	0.00	0.00	8,987	6,710	6,510	1,312	4,268	858	0.2
A2-LT	8,969	22.7	0.00	0.00	8,969	6,710	6,510	1,312	9,353	848	0.1
A3-LT	14,204	31.1	0.01	0.00	14,203	6,710	6,510	1,312	15,796	934	0.1
A4-LT	7,871	3.4	0.00	0.00	7,871	6,710	6,510	1,312	1,821	891	0.1
A5-LT	9,179	16.9	0.00	0.00	9,179	6,710	6,510	1,312	5,150	876	0.1
A6-LT	12,833	373.8	0.01	0.00	12,832	6,710	6,510	1,312	214,606	868	0.1
A7-LT	9,088	27.6	0.01	0.00	9,087	6,710	6,510	1,312	13,224	901	0.2
A8-LT	10,247	3.7	0.00	0.00	10,247	6,710	6,510	1,312	3,248	905	0.1
A9-LT	11,798	55.6	0.01	0.00	11,797	6,710	6,510	1,312	28,387	889	0.1
A10-LT	13,058	10.8	0.01	0.00	13,057	6,710	6,510	1,312	4,077	848	0.2
AVG	10,623	56.4	0.01	0.00	10,623	6,710	6,510	1,312	29,993	882	0.1

Table A.18: Computational results of CLSP-FS1-LT formulation: AG-SO

	MIP									LP	
Inst.	Obj	Time	E.Gap(%)	R.Gap(%)	LB	Cols	Bin	Rows	Nodes	Obj	Time
A1-LT	8,987	80.6	0.00	0.00	8,987	102,500	102,300	310	6,675	1,217	2.1
A2-LT	8,969	274.3	0.01	0.00	8,968	102,500	102,300	310	28,498	1,227	2.2
A3-LT	14,273	600.1	3.79	0.48	13,733	102,500	102,300	310	35,556	1,287	2.2
A4-LT	7,871	31.4	0.00	0.00	7,871	102,500	102,300	310	2,531	1,218	1.9
A5-LT	9,179	106.4	0.00	0.00	9,179	102,500	102,300	310	4,411	1,292	2.2
A6-LT	12,833	600.1	13.11	0.00	11,151	102,500	102,300	310	36,286	1,242	1.4
A7-LT	9,088	139.3	0.01	0.00	9,087	102,500	102,300	310	16,909	1,271	1.8
A8-LT	10,247	62.1	0.00	0.00	10,247	102,500	102,300	310	6,495	1,257	1.6
A9-LT	11,798	600.2	9.88	0.00	10,633	102,500	102,300	310	36,275	1,240	2.2
A10-LT	13,058	41.7	0.00	0.00	13,058	102,500	102,300	310	3,396	1,199	2.1
AVG	10,630	253.6	2.68	0.05	10,291	102,500	102,300	310	17,703	1,245	2.0

Table A.19: Computational results of CLSP-FS1-LT formulation: AG-SCM

	MIP									LP	
Inst.	Obj	Time	E.Gap(%)	R.Gap(%)	LB	Cols	Bin	Rows	Nodes	Obj	Time
A1-LT	8,987	1.9	0.00	0.00	8,987	2,510	1,210	1,731	2,416	0	0.0
A2-LT	8,969	2.9	0.01	0.00	8,969	2,510	1,210	1,731	4,130	0	0.0
A3-LT	14,204	2.7	0.00	0.00	14,204	2,510	1,210	1,731	3,774	0	0.0
A4-LT	7,871	1.1	0.00	0.00	7,871	2,510	1,210	1,731	642	0	0.0
A5-LT	9,179	1.7	0.00	0.00	9,179	2,510	1,210	1,731	1,104	0	0.0
A6-LT	12,833	15.6	0.01	0.00	12,832	2,510	1,210	1,731	15,018	0	0.0
A7-LT	9,088	2.7	0.00	0.00	9,088	2,510	1,210	1,731	2,271	0	0.0
A8-LT	10,247	1.1	0.00	0.00	10,247	2,510	1,210	1,731	464	0	0.0
A9-LT	11,798	7.4	0.01	0.00	11,797	2,510	1,210	1,731	8,066	0	0.0
A10-LT	13,058	1.5	0.00	0.00	13,058	2,510	1,210	1,731	1,258	0	0.0
AVG	10,623	3.9	0.00	0.00	10,623	2,510	1,210	1,731	3,914	0	0.0

Table A.20: Computational results of CLSP-FS1-LT formulation: AG-LT

	MIP									LP	
Inst.	Obj	Time	E.Gap(%)	R.Gap(%)	LB	Cols	Bin	Rows	Nodes	Obj	Time
A1-LT	8,987	0.2	0.00	0.00	8,987	770	220	1,071	0	6,081	0.0
A2-LT	8,969	0.2	0.00	0.00	8,969	770	220	1,071	0	6,036	0.0
A3-LT	14,204	0.2	0.00	0.00	14,204	770	220	1,071	0	7,719	0.0
A4-LT	7,871	0.1	0.00	0.00	7,871	770	220	1,071	0	5,892	0.0
A5-LT	9,179	0.1	0.00	0.00	9,179	770	220	1,071	0	6,289	0.0
A6-LT	12,833	0.3	0.00	0.00	12,833	770	220	1,071	0	7,246	0.0
A7-LT	9,088	0.2	0.00	0.00	9,088	770	220	1,071	0	6,240	0.0
A8-LT	10,247	0.1	0.00	0.00	10,247	770	220	1,071	0	6,363	0.0
A9-LT	11,798	0.2	0.00	0.00	11,798	770	220	1,071	0	7,062	0.0
A10-LT	13,058	0.1	0.00	0.00	13,058	770	220	1,071	0	6,936	0.0
AVG	10,623	0.2	0.00	0.00	10,623	770	220	1,071	0	6,586	0.0

Table A.21: Computational results of CLSP-FS1-LT formulation: AG-PO

	MIP									LP	
Inst.	Obj	Time	E.Gap(%)	R.Gap(%)	LB	Cols	Bin	Rows	Nodes	Obj	Time
A1-LT	8,987	50.6	0.01	0.00	8,986	7,060	6,510	1,762	27,784	6,023	0.3
A2-LT	8,969	79.3	0.01	0.00	8,968	7,060	6,510	1,762	33,716	5,894	0.3
A3-LT	14,204	69.1	0.01	0.00	14,203	7,060	6,510	1,762	48,768	7,309	0.3
A4-LT	7,871	4.1	0.00	0.00	7,871	7,060	6,510	1,762	2,395	5,880	0.3
A5-LT	9,179	28.5	0.01	0.00	9,179	7,060	6,510	1,762	9,282	6,227	0.3
A6-LT	12,833	555.6	0.01	0.00	12,832	7,060	6,510	1,762	219,805	6,929	0.3
A7-LT	9,088	55.3	0.01	0.00	9,087	7,060	6,510	1,762	21,037	6,173	0.3
A8-LT	10,247	6.7	0.01	0.00	10,246	7,060	6,510	1,762	5,855	6,292	0.3
A9-LT	11,798	600.0	2.98	0.00	11,446	7,060	6,510	1,762	160,870	6,796	0.3
A10-LT	13,058	6.9	0.00	0.00	13,058	7,060	6,510	1,762	5,559	6,670	0.3
AVG	10,623	145.6	0.30	0.00	10,588	7,060	6,510	1,762	53,507	6,419	0.3

Table A.22: Computational results of CLSP-FS1-LT formulation: AG-SO

	MIP									LP	
Inst.	Obj	Time	E.Gap(%)	R.Gap(%)	LB	Cols	Bin	Rows	Nodes	Obj	Time
A1-LT	8,987	34.4	0.00	0.00	8,987	102,850	102,300	760	416	6,081	1.9
A2-LT	8,969	49.9	0.00	0.00	8,969	102,850	102,300	760	3,868	6,036	2.1
A3-LT	14,204	26.5	0.00	0.00	14,204	102,850	102,300	760	119	7,719	2.3
A4-LT	7,871	13.3	0.00	0.00	7,871	102,850	102,300	760	0	5,892	2.0
A5-LT	9,179	27.6	0.00	0.00	9,179	102,850	102,300	760	0	6,289	1.6
A6-LT	12,833	47.3	0.00	0.00	12,833	102,850	102,300	760	1,893	7,246	2.7
A7-LT	9,088	29.8	0.00	0.00	9,088	102,850	102,300	760	573	6,240	2.0
A8-LT	10,247	11.6	0.00	0.00	10,247	102,850	102,300	760	0	6,363	2.1
A9-LT	11,798	33.7	0.00	0.00	11,798	102,850	102,300	760	880	7,062	2.7
A10-LT	13,058	18.9	0.00	0.00	13,058	102,850	102,300	760	0	6,936	2.5
AVG	10,623	29.3	0.00	0.00	10,623	102,850	102,300	760	775	6,586	2.2

Table A.23: Computational results of CLSP-FS1-LT formulation: AG-SCM

	MIP									LP	
Inst.	Obj	Time	E.Gap(%)	R.Gap(%)	LB	Cols	Bin	Rows	Nodes	Obj	Time
A1-LT	8,987	1.3	0.01	0.00	8,986	2,860	1,210	2,181	976	0	0.0
A2-LT	8,969	1.8	0.00	0.00	8,969	2,860	1,210	2,181	813	0	0.0
A3-LT	14,204	2.4	0.00	0.00	14,204	2,860	1,210	2,181	2,341	0	0.0
A4-LT	7,871	1.0	0.00	0.00	7,871	2,860	1,210	2,181	384	0	0.0
A5-LT	9,179	1.5	0.00	0.00	9,179	2,860	1,210	2,181	352	0	0.0
A6-LT	12,833	14.1	0.00	0.00	12,833	2,860	1,210	2,181	10,667	0	0.0
A7-LT	9,088	1.6	0.00	0.00	9,088	2,860	1,210	2,181	1,566	0	0.0
A8-LT	10,247	1.2	0.00	0.00	10,247	2,860	1,210	2,181	319	0	0.0
A9-LT	11,798	2.7	0.01	0.00	11,797	2,860	1,210	2,181	3,559	0	0.0
A10-LT	13,058	2.0	0.00	0.00	13,058	2,860	1,210	2,181	1,212	0	0.0
AVG	10,623	3.0	0.00	0.00	10,623	2,860	1,210	2,181	2,219	0	0.0

Bibliography

[1] Abad, P. (2003). Optimal pricing and lot-sizing under conditions of perishability, finite production and partial backordering and lost sale. *European Journal of Operational Research*, 144(3):677–685.

[2] Abad, P. L. (2008). Optimal price and order size under partial backordering incorporating shortage, backorder and lost sale costs. *International Journal of Production Economics*, 114(1):179–186.

[3] Absi, N. and Kedad-Sidhoum, S. (2007). MIP-based heuristics for multi-item capacitated lot-sizing problem with setup times and shortage costs. *RAIRO - Operations Research*, 41(2):171–192.

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

[5] Aggarwal, S. C. (1974). A review of current inventory theory and its applications. *International Journal of Production Research*, 12(4):443–482.

[6] Aksen, D., Altinkemer, K., and Chand, S. (2003). The single-item lot-sizing problem with immediate lost sales. *European Journal of Operational Research*, 147(3):558–566.

[7] Aksoy, Y. and Selcuk Erenguc, S. (1988). MultiItem Inventory Models with Coordinated Replenishments: A Survey. *International Journal of Operations & Production Management*, 8(1):63–73.

[8] Allahverdi, A., Ng, C. T., Cheng, T. C. E., and Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187:985–1032.

[9] Almada-Lobo, B., Carravilla, M., and Oliveira, J. (2008). Production planning and scheduling in the glass con- tainer industry: a vns approach. *Int J Prod Econ*, 114(1):363375.

[10] Almada-Lobo, B., Clark, A., Guimarães, L., Figueira, G., and Amorim, P. (2015). INDUSTRIAL INSIGHTS INTO LOT SIZING AND SCHEDULING MODELING. *Pesquisa Operacional*, 35(3):439–464.

[11] Almada-Lobo, B. and James, R. (2010). Neighbourhood search meta-heuristics for capacitated lot-sizing with sequence-dependent setups. *Int J Prod Res*, 48(3):861878.

[12] Atkins, D. and Sun, D. (1995). 98%-Effective Lot Sizing for Series Inventory Systems with Backlogging. *Operations Research*, 43(2):335–345.

[13] Bahl, H. C., Ritzman, L. P., and Gupta, J. N. D. (1987). OR Practice–Determining Lot Sizes and Resource Requirements: A Review. *Operations Research*, 35(3):329–345.

[14] Belvaux, G. and Wolsey, L. A. (2000). bc - prod: A Specialized Branch-and-Cut System for Lot-Sizing Problems. *Management Science*, 46(5):724–738.

[15] Belvaux, G. and Wolsey, L. A. (2001). Modelling Practical Lot-Sizing Problems as Mixed-Integer Programs. *Management Science*, 47(7):993–1007.

[16] Benton, W. C. and Park, S. (1996). A classification of literature on determining the lot size under quantity discounts. *European Journal of Operational Research*, 92(2):219–238.

[17] Bitran, G. R. and Yanasse, H. H. (1982). Computational Complexity of the Capacitated Lot Size Problem. *Management Science*, 28(10):1174–1186.

[18] Blackburn, J. D. and Kunreuther, H. (1974). Planning Horizons for the Dynamic Lot Size Model with Backlogging. *Management Science*, 21(3):251–255.

[19] Brahimi, N. (2004). Planification de la production: modéles et algorithmes pour les problems de demensionnementde lots. *PhD thesis, Université Nants*.

[20] Brahimi, N., Absi, N., Dauzere-Peres, S., and Nordli, A. (2017). Single-item dynamic lot-sizing problems: An updated survey. *European Journal of Operational Research*, pages –.

[21] Brahimi, N., Dauzère-Pérès, S., and Najid, N. M. (2006a). Capacitated Multi-Item Lot-Sizing Problems with Time Windows. *Operations Research*, 54(5):951–967.

[22] Brahimi, N., Dauzere-Peres, S., Najid, N. M., and Nordli, A. (2006b). Single item lot sizing problems. *European Journal of Operational Research*, 168(1):1–16.

[23] Brahimi, N., Dauzere-Peres, S., and Wolsey, L. A. (2010). Polyhedral and Lagrangian approaches for lot sizing with production time windows and setup times. *Computers & Operations Research*, 37(1):182–188.

[24] Briskorn, D. (2006). A note on capacitated lot sizing with setup carry over. *IIE Transactions*, 38(11):1045–1047.

[25] Buschkühl, L., Sahling, F., Helber, S., and Tempelmeier, H. (2010). *Dynamic capacitated lot-sizing problems: A classification and review of solution approaches*, volume 32.

[26] Charrua, F., Brojo, F., and M., P. (2012). Lot Sizing and Scheduling in Parallel Uniform Machines A Case Study. In *Simulated Annealing - Advances, Applications and Hybridizations*. InTech.

[27] Chen, H.-D., Hearn, D. W., and Lee, C.-Y. (1994). A dynamic programming algorithm for dynamic lot size models with piecewise linear costs. *Journal of Global Optimization*, 4(4):397–413.

[28] Chen, W. H. and Thizy, J. M. (1990). Analysis of relaxations for the multi-item capacitated lot-sizing problem. *Annals of Operations Research*, 26(1):29–72.

[29] Choo, E. U. and Chan, G. H. (1990). Two-way eyeballing heuristics in dynamic lot sizing with backlogging. *Computers & Operations Research*, 17(4):359–363.

[30] CIFRE. http://www.anrt.asso.fr/.

[31] Copil, K., Wörbelauer, M., Meyr, H., and Tempelmeier, H. (2017). Simultaneous lotsizing and scheduling problems: a classification and review of models. *OR Spectrum*, 39(1):1–64.

[32] Dauzère-Pérès, S., Brahimi, N., Najid, N. M., and Nordli, A. (2005). Uncapacitated Lot-Sizing Problems with Time windows. *Technical report, Ecole des Mines de Saint-Etienne*.

[33] De Bodt, M. A., Gelders, L. F., and Van Wassenhove, L. N. (1984). Lot sizing under dynamic demand conditions: A review. *Engineering Costs and Production Economics*, 8(3):165–187.

[34] Denizel, M., Altekin, F. T., Süral, H., and Stadtler, H. (2008). Equivalence of the LP relaxations of two strong formulations for the capacitated lot-sizing problem with setup times. *OR Spectrum*, 30(4):773–785.

[35] Diaby, M., Bahl, H. C., Karwan, M. H., and Zionts, S. (1992). A Lagrangean Relaxation Approach for Very-Large-Scale Capacitated Lot-Sizing. *Management Science*, 38(9):1329–1340.

[36] Dillenberger, C., Escudero, L. F., Wollensak, A., and Zhang, W. (1993). On Solving a Large-Scale Resource Allocation Problem in Production Planning. In *Operations Research in Production Planning and Control*, number 2, pages 105–119. Springer Berlin Heidelberg, Berlin, Heidelberg.

[37] Dillenberger, C., Escudero, L. F., Wollensak, A., and Zhang, W. (1994). On practical resource allocation for production planning and scheduling with period overlapping setups. *European Journal of Operational Research*, 75(2):275–286.

[38] Dilts, D. M. and Ramsing, K. D. (1989). Joint Lot Sizing and Scheduling of Multiple Items with Sequence Dependent Setup Costs. *Decision Sciences*, 20(1):120–133.

[39] Dobson, G. (1992). The cyclic lot scheduling problem with sequence-dependent setups. *Operations Research*, 40:736–749.

[40] Drexl, a. and Kimms, a. (1997). Lot sizing and scheduling Survey and extensions. *European Journal of Operational Research*, 99(97):221–235.

[41] Duda, J. (2005). Lot-Sizing in a Foundry Using Genetic Algorithm and Repair Functions. In *EvoCOP 2005, Lecture Notes in Computer Science 3448*, pages 101–111.

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

[43] Federgruen, A. and Tzur, M. (1993). The dynamic lot-sizing model with backlogging: A simpleo(n logn) algorithm and minimal forecast horizon procedure. *Naval Research Logistics*, 40(4):459–478.

[44] Fiorotto, D. J. and de Araujo, S. A. (2014). Reformulation and a Lagrangian heuristic for lot sizing problem on parallel machines. *Annals of Operations Research*, 217(1):213–231.

[45] Fiorotto, D. J., de Araujo, S. A., and Jans, R. (2015). Hybrid methods for lot sizing on parallel machines. *Computers & Operations Research*, 63(May 2015):136–148.

[46] Fleischmann, B. (1994). The discrete lot-sizing and scheduling problem with sequence-dependent setup costs. *European Journal of Operational Research*, 75(2):395–404.

[47] Florian, M., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1980). Deterministic Production Planning: Algorithms and Complexity. *Management Science*, 26(7):669–679.

[48] Focacci, F., Furini, F., Gabrel, V., Godard, D., and Shen, X. (2016). *MIP Formulations for a Rich Real-World Lot-Sizing Problem with Setup Carryover*, pages 123–134. Springer International Publishing, Cham.

[49] Furini, F. and Shen, X. (2018). Matheuristics for the temporal bin packing problem. *In: Amodeo L., Talbi EG., Yalaoui F. (eds) Recent Developments in Metaheuristics. Operations Research/Computer Science Interfaces Series*, 62.

[50] Gaafar, L. (2006). Applying genetic algorithms to dynamic lot sizing with batch ordering. *Computers and Industrial Engineering*, 51(3):433–444.

[51] Gagne, C., Price, W., and Gravel, M. (2002). Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times. *The Journal of the Operational Research Society*, 53:895–906.

[52] Gallego, G. and Shaw, D. (1997). Complexity of the elsp with general cyclic schedules. *IEEE Transactions*, 29:109–130.

[53] Gelders, L. F. L. N. W. (1981). Production planning: a review. *European Journal of Operational Research*, 7(2):101–110.

[54] Ghosh, S., Khanra, S., and Chaudhuri, K. (2011). Optimal price and lot size determination for a perishable product under conditions of finite production, partial backordering and lost sale. *Applied Mathematics and Computation*, 217(13):6047–6053.

[55] Glock, C. H., Grosse, E. H., and Ries, J. M. (2014). The lot sizing problem: A tertiary study. *International Journal of Production Economics*, 155(September):39–51.

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

[57] Gopalakrishnan, M., Ding, K., Bourjolly, J.-M., and Mohan, S. (2001). A Tabu-Search Heuristic for the Capacitated Lot-Sizing Problem with Set-up Carryover. *Management Science*, 47(6):851–863.

[58] Gopalakrishnan, M., Miller, D. M., and Schmidt, C. P. (1995). A framework for modelling setup carryover in the capacitated lot sizing problem.

[59] Gören, H. G. and Tunal, S. (2015). Solving the capacitated lot sizing problem with setup carryover using a new sequential hybrid approach. *Applied Intelligence*, 42(4):805–816.

[60] Goren, H. G., Tunali, S., and Jans, R. (2012). A hybrid approach for the capacitated lot sizing problem with setup carryover. *International Journal of Production Research*, 50(6):1582–1597.

[61] Goyal, S. K. and Giri, B. C. (2001). Recent trends in modeling of deteriorating inventory. *European Journal of Operational Research*, 134(1):1–16.

[62] Guiffrida, M. A. B., Alfred, Jaber, M., and Khan, M. (2015). A review of inventory lot sizing review papers. *Management Research Review*, 38(3):283–298.

[63] Guimarães, L., Klabjan, D., and Almada-Lobo, B. (2014). Modeling lotsizing and scheduling problems with sequence dependent setups. *European Journal of Operational Research*, 239(3):644–662.

[64] Guner Goren, H., Tunali, S., and Jans, R. (2010). A review of applications of genetic algorithms in lot sizing. *Journal of Intelligent Manufacturing*, 21(4):575–590.

[65] Gupta, S. and Brennan, L. (1992). Lot Sizing and Backordering in Multi-Level Product Structures. *Production and Inventory Management Journal*, 33(1):27.

[66] Gupta, Y. P. and Keung, Y. (1990). A Review of Multistage Lotsizing Models. *International Journal of Operations & Production Management*, 10(9):57–73.

[67] Haase, K. (1994). *Lotsizing and Scheduling for Production Planning*, volume 408 of *Lecture Notes in Economics and Mathematical Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[68] Haase, K. (1996). Capacitated lot-sizing with sequence dependent setup costs. *Operations-Research-Spektrum*, 18(1):51–59.

[69] Haase, K. (1998). Capacitated Lot-Sizing with Linked Production Quantities of Adjacent Periods 1 Introduction. *Beyond Manufacturing Resource Planning (MRP II)*, pages 127–146.

[70] Haase, K. and Kimms, A. (2000). Lot sizing and scheduling with sequence-dependent setup costs and times and efficient rescheduling opportunities. *International Journal of Production Economics*, 66(2):159–169.

[71] Harris, F. (1913). How many parts to make at once. *Factory - The Magazine of Management*, 10(2):135–136, 152.

[72] Helber, S. and Sahling, F. (2010). A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *International Journal of Production Economics*, 123(2):247 – 256.

[73] Hindi, K. S. (1995). Algorithms for Capacitated, Multi-Item Lot-Sizing without Set-Ups. *The Journal of the Operational Research Society*, 46(4):465–472.

[74] Hsieh, H., Lam, K. F., and Choo, E. U. (1992). Comparative study of dynamic lot sizing heuristics with backlogging. *Computers and Operations Research*, 19(5):393–407.

[75] Hsu, V. N. and Lowe, T. J. (2001). Dynamic Economic Lot Size Models with Period-Pair-Dependent Backorder and Inventory Costs. *Operations Research*, 49(2):316–321.

[76] Huai-En Chiao, Hui-Ming Wee, and Po-Chung Yang (2008). A deteriorating inventory model with tow storage facilities, partial backordering and quantity discount. In *2008 International Conference on Machine Learning and Cybernetics*, volume 7, pages 3903–3908. IEEE.

[77] Hung, Y. F., Chen, C. P., Shih, C. C., and Hung, M. H. (2003). Using tabu search with ranking candidate list to solve production planning problems with setups. *Computers and Industrial Engineering*, 45(4):615–634.

[78] Hung, Y.-F. and Chien, K.-L. (2000). A Multi-Class Multi-Level Capacitated Lot Sizing Model. *The Journal of the Operational Research Society*, 51(11):1309.

[79] Hung, Y.-f., Shih, C.-c., and Chen, C.-p. (1999). Evolutionary Algorithms for Production Planning Problems with Setup Decisions. *The Journal of the Operational Research Society*, 50(8):857–866.

[80] Hwang, H.-C. (2007). Dynamic lot-sizing model with production time windows. *Naval Research Logistics*, 54(6):692–701.

[81] I, S., H, K., H, D., and D, L. (2011). A two-stage heuristic for single machine capacitated lot-sizing and scheduling with sequence-dependent setup costs. *Comput Ind Eng*, 61(4):920929.

[82] James, R. J. W. and Almada-Lobo, B. (2011). Single and parallel machine capacitated lotsizing and scheduling: New iterative MIP-based neighborhood search heuristics. *Computers and Operations Research*, 38(12):1816–1825.

[83] Jans, R. and Degraeve, Z. (2004). An industrial extension of the discrete lot-sizing and scheduling problem. *IIE Transactions*, 36(1):47–58.

[84] Jans, R. and Degraeve, Z. (2007). Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research*, 177(3):1855–1875.

[85] Jans, R. and Degraeve, Z. (2008). Modeling industrial lot sizing problems: a review. *International Journal of Production Research*, 46(March 2015):1619–1643.

[86] Kämpf, M. and Köchel, P. (2006). Simulation-based sequencing and lot size optimisation for a production-and-inventory system with multiple items. *International Journal of Production Economics*, 104(1):191–200.

[87] Kang, S., Malik, K., and Thomas, L. J. (1999). Lotsizing and Scheduling on Parallel Machines with Sequence-Dependent Setup Costs. *Management Science*, 45(2):273–289.

[88] Karimi, B., Fatemi Ghomi, S. M. T., and Wilson, J. M. (2003). The capacitated lot sizing problem: A review of models and algorithms. *Omega*, 31(5):365–378.

[89] Karimi, B., Ghomi, S. M. T. F., and Wilson, J. M. (2006). A tabu search heuristic for solving the CLSP with backlogging and set-up carry-over. *Journal of the Operational Research Society*, 57(2):140–147.

[90] Krarup, J. and Bilde, . (1977). Plant location, set covering and economic lot size: An $o(mn)$ algorithm for structural problems. *Numerische Methoden bei Optimierungsaufgabeen, Band 3: Optimierung bei graphentheoretischen und ganzzahligen Probleme*, 36.

[91] Kuik, R., Salomon, M., and van Wassenhove, L. N. (1994). Batching decisions: structure and models. *European Journal of Operational Research*, 75(2):243–263.

[92] Laguna, M. (1999). A heuristic for production scheduling and inventory control in the presence of sequence-dependent setup time. *IIE Trans*, 31(2):125134.

[93] Lang, J. and Shen, Z. (2011). Fix-and-optimize heuristics for capacitated lot-sizing with sequence-dependent setups and substitutions. *Eur J Oper Res*, 214(3):595605.

[94] Love, S. F. (1973). Bounded production and inventory models with piecewise concave costs. *Management Science*, 20(3):313–318.

[95] Maes, J. and Wassenhove, L. V. (1988). Multi-Item Single-Level Capacitated Dynamic Lot-Sizing Heuristics: A General Review. *The Journal of the Operational Research Society*, 39(11):991–1004.

[96] Manne, A. (1958). Programming of economic lot sizes. *Management Science*, 4:115–135.

[97] Megala, N. and Jawahar, N. (2006). Genetic algorithm and Hopfield neural network for a dynamic lot sizing problem. *International Journal of Advanced Manufacturing Technology*, 27(11-12):1178–1191.

[98] Meyr, H. (2002). Simultaneous lotsizing and scheduling on parallel machines. *European Journal of Operational Research*, 139(2):277–292.

[99] Millar, H. H. and Yang, M. (1993). An application of Lagrangean decomposition to the capacitated multi-item lot sizing proble. *Computers and Operations Research*, 20(4):409–420.

[100] Millar, H. H. and Yang, M. (1994). Lagrangian heuristics for the capacitated multi-item lot-sizing problem with backordering. *International Journal of Production Economics*, 34(1):1–15.

[101] Nahmias, S. (1982). Perishable inventory theory: A review. *Operations Research*, 30(4):680–708.

[102] Nascimento, M. C. V. and Toledo, F. M. B. (2008). A hybrid heuristic for the multi-plant capacitated lot sizing problem with setup carry-over. *Journal of the Brazilian Computer Society*, 14(4):7–15.

[103] Nattaf, M., Artigues, C., Lopez, P., Medina, R., Parada, V., and Pradenas, L. (2015). A batch sizing and scheduling problem on parallel machines with different speeds, maintenance operations, setup times and energy costs. In *2015 International Conference on Industrial Engineering and Systems Management (IESM)*, pages 883–891. IEEE.

[104] Nemhauser, G. and Wosley, L. (1988). Integer and combinartorial optimization. *Wiley, New York*.

[105] Ozdamar, L. and Barbarosoglu, G. (1999). Hybrid Heuristics for the Multi-Stage Capacitated Lot Sizing and Loading Problem. *The Journal of the Operational Research Society*, 50(8):810–825.

[106] Özdamar, L. and Birbil, e. l. (1998). Hybrid heuristics for the capacitated lot sizing and loading problem with setup times and overtime decisions. *European Journal of Operational Research*, 110(3):525–547.

[107] Pochet, Y. and Wolsey, L. A. (1988). Lot-size models with backlogging: Strong reformulations and cutting planes. *Mathematical Programming*, 40-40(1-3):317–335.

[108] Pochet, Y. and Wolsey, L. A. (2006). *Production Planning by Mixed Integer Programming*. Springer-Verlag New York.

[109] Quadt, D. and Kuhn, H. (2008). Capacitated lot-sizing with extensions: A review. *4or*, 6:61–83.

[110] Quadt, D. and Kuhn, H. (2009). Capacitated lot-sizing and scheduling with parallel machines, back-orders, and setup carry-over. *Naval Research Logistics*, 56(4):366–384.

[111] Raafat, F. (1991). Survey of literature on continuously deteriorating inventory models. *Journal of the Operational Research Society*, 42(1):27–37.

[112] Richter, K. and Sombrutzki, M. (2000). Remanufacturing planning for the reverse wagner/whitin models. *European Journal of Operational Research*, 121(2):304 – 315.

[113] Rizk, N. and Martel, a. (2001). Supply chain flow planning methods: a review of the lot-sizing literature. *Quebec City, QC, Canada: Université Laval*, (January):1 – 66.

[114] Robinson, P., Narayanan, A., and Sahin, F. (2009). Coordinated deterministic dynamic demand lot-sizing problem: A review of models and algorithms. *Omega*, 37(1):3–15.

[115] Robinson Jr., E. P. and Gao, L.-L. (1996). A dual ascent procedure for multi-product dynamic demand coordinated replenishment with backlogging. *Management Science*, 42(11):1556–1564.

[116] Sahling, F., Buschkühl, L., Tempelmeier, H., and Helber, S. (2009). Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a fix-and-optimize heuristic. *Computers & Operations Research*, 36(9):2546–2553.

[117] Salomon, M., Solomon, M., Wassenhove, L. V., Dumas, Y., and Dauzere-Peres, S. (1997). Solving the discrete lotsizing and scheduling with sequence dependent set-up costs and set-up times using the travelling salesman problem with time windows. *European Journal of Operational Research 100*, 100:494–513.

[118] Sandbothe, R. A. and Thompson, G. L. (1990). A Forward Algorithm for the Capacitated Lot Size Model with Stockouts. *Operations Research*, 38(3):474–486.

[119] Sandbothe, R. A. and Thompson, G. L. (1993). Decision horizons for the capacitated lot size model with inventory bounds and stockouts. *Computers and Operations Research*, 20(5):455–465.

[120] Sargut, F. Z. and Romeijn, H. E. (2007). Lot-sizing with non-stationary cumulative capacities. *Operations Research Letters*, 35(4):549 – 557.

[121] Schrage, L. (1982). The multiproduct lot scheduling problem. *M.A.H. Dempster et al. (Eds.), Deterministic and Stocastic Scheduling, Dordrecht/Holland*, pages 233–244.

[122] Sox, C. and Gao, Y. (1999). The capacitated lot sizing problem with setup carry-over. *IIE Transactions*, 31(2):173–181.

[123] Staggemeier, A. T. and Clark, A. (2001). A survey of lot-sizing and scheduling models.

[124] Suerie, C. and Stadtler, H. (2003). The Capacitated Lot-Sizing Problem with Linked Lot Sizes. *Management Science*, 49(8):1039–1054.

[125] Sun, D. and Atkins, D. (1997). 98%-Effective Lot-Sizing for Assembly Inventory Systems with Backlogging. *Operations Research*, 45(6):940–951.

[126] Tempelmeier, H. and Copil, K. (2015). Capacitated lot sizing with parallel machines, sequence-dependent setups, and a common setup operator. *OR Spectrum*.

[127] Teng, J.-T., Ouyang, L.-Y., and Chen, L.-H. (2007). A comparison between two pricing and lot-sizing models with partial backlogging and deteriorated items. *International Journal of Production Economics*, 105(1):190–203.

[128] Toledo, F. M. B. and Armentano, V. A. (2006). A Lagrangian-based heuristic for the capacitated lot-sizing problem in parallel machines. *European Journal of Operational Research*, 175(2):1070–1083.

[129] Trigeiro, W. W., Thomas, L. J., and McClain, J. O. (1989). Capacitated Lot Sizing with Setup Times. *Management Science*, 35(3):353–366.

[130] van den Heuvel, W. and Wagelmans, A. P. (2008). Four equivalent lot-sizing models. *Operations Research Letters*, 36(4):465–470.

[131] Wagner, H. M. and Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management Science*, 5(1):89–96.

[132] Wolsey, L. A. (1995). Progress with single-item lot-sizing. *European Journal of Operational Research*, 86(3):395–401.

[133] Wolsey, L. A. (2006). Lot-sizing with production and delivery time windows. *Mathematical Programming*, 107(3):471–489.

[134] Zangwill, W. I. (1966). A Deterministic Multi-Period Production Scheduling Model with Backlogging. *Management Science*, 13(1):105–119.

[135] Zangwill, W. I. (1969). A Backlogging Model and a Multi-Echelon Model of a Dynamic Economic Lot Size Production SystemA Network Approach. *Management Science*, 15(9):506–527.

[136] Zhu, X. and Wilhelm, W. E. (2006). Scheduling and lot sizing with sequence-dependent setup: A literature review. *IIE Transactions*, 38(11):987–1007.

[137] Zoller, K. and Robrade, A. (1988). Dynamic lot sizing techniques: Survey and comparison. *Journal of Operations Management*, 7(3-4):125–148.

PROBLÈMES COMPLEXES DE DIMENSIONNEMENT DE LOTS DE PRODUCTION AVEC MACHINES PARALLÈLES ET REPORT DE CONFIGURATION

Chapitre 1

Introduction

Nos recherches sont réalisées dans le cadre du programme CIFRE (Conventions Industrielles de Formation par la REcherche) [30]. C'est donc une collaboration entre l'Université Paris Dauphine et DecisionBrain (https ://www.decisionbrain.com). DecisionBrain est une société de logiciels qui fournit des solutions avancées d'analyse et d'optimisation pour les entreprises innovantes qui souhaitent mettre en œuvre une démarche scientifique à la prise de décisions. La réalisation de solution de planification et d'ordonnancement de la production fait partie de l'expertise de DecisionBrain. Grâce à ce contexte industriel, nous avons été confronté à différents types d'applications réelles. Dans cette thèse, nous étudions deux problèmes de planification de la production motivés par des applications réelles complexes.

Dans la première partie de ce manuscrit, nous étudions un problème de planification de la production pour une entreprise de fabrication de vêtements et mettons au point un outil d'optimisation pour le résoudre. Nous proposons un framework de décomposition composé d'un modèle agrégé et d'un modèle détaillé, qui sont résolus en séquence. Le problème agrégé est le goulot d'étranglement de l'approche, et correspond à un problème de lot-sizing à capacité finie avec report de setup, machines parallèles, délais de production, arriérés et pertes de ventes. Ce problème s'est avéré NP-difficile, même sans les coûts de setup. Plusieurs formulations de programmation linéaire mixte (MIP) sont proposées et comparées d'un point de vue théorique et expérimental. De plus, plusieurs heuristiques de recherche constructives et locales sont développées pour trouver des solutions de bonne qualité sur les instances de grande taille. Nous proposons deux ensembles d'instances de benchmark pour évaluer les performances des modèles et des heuristiques. Grâce à des tests expérimentaux approfondis, nous avons montré que l'heuristique constructive (appelée Heuristic First-Solution) associée à un algorithme Fix&Optimize détermine les solutions approchées s'écartant le moins des valeurs optimales. Enfin, l'ensemble de la

démarche de planification de la production est présentée et sa performance est analysée.

Dans la deuxième partie de ce manuscrit, nous étudions une version restreinte du problème de lot-sizing à capacité finie avec setups dépendants de la séquence, où les séquences de setup pour chaque période doivent suivre l'ordre d'une séquence donnée. Par rapport au problème de lot-sizing à capacité finie avec setups dépendantes de la séquence, le nouveau modèle réduit le nombre de séquences de setup candidates de $O(n!)$ à $O(n2^n)$. Ce problème s'avère être NP-difficile. Un cas particulier avec seulement deux valeurs de setup possibles est étudié : nous prouvons que, dans ce cas également, le problème reste NP-difficile. De plus, des formulations MIP orientées produits et séquentielles sont développées. Une heuristique de génération de colonnes est également proposée à partir des formulations séquentielles. Enfin, nous effectuons des tests de calcul pour évaluer leurs performances respectives.

Chapitre 2

Problème complexe de lot-sizing : formulations et benchmarks

Nous construisons un problème complexe de lot-sizing à capacité finie basé sur une application de fabrication de vêtements. Ce problème de lot-sizing à capacité finie est composé d'éléments complexes tels que les machines parallèles, les fenêtres de temps de production, l'arriéré, les pertes de ventes et le report de setup [48]. Ces caractéristiques ont été étudiées dans différents contextes de problèmes de lot-sizing. Cependant, à notre connaissance, ils sont pour la première fois considérés ensemble dans cette application. Dans ce chapitre, nous définissons, formulons et analysons formellement le problème.

Les paramètres d'entrée du problème sont :

— $\mathcal{T} = \{1, 2, \ldots, T\}$: ensemble des périodes.
— $\mathcal{R} = \{1, 2, \ldots, R\}$: ensemble des ressources/machines.
— $\mathcal{N} = \{1, 2, \ldots, N\}$: ensemble des produits.
— $\mathcal{D} = \{1, 2, \ldots, D\}$: ensemble des demandes.
— cap_{rt} : capacité de la machine r dans la période t ($r \in \mathcal{R}$, $t \in \mathcal{T}$).
— pt_i : temps de traitement unitaire du produit i ($i \in \mathcal{N}$).
— st_{ir} : capacité de setup pour le produit i sur la machine r ($i \in \mathcal{N}$,$r \in \mathcal{R}$).
— sc_{ir} : coût de setup du produit i sur la machine r ($i \in \mathcal{N}$,$r \in \mathcal{R}$).
— $p_d \in \mathcal{N}$: produit requis par la demande d ($d \in \mathcal{D}$).
— q_d : quantité de produit p_d requise par la demande d ($d \in \mathcal{D}$).
— $b_d \in \mathcal{T}$: date de début de la demande d ($d \in \mathcal{D}$).
— $e_d^1 \in \mathcal{T}$: première échéance de la demande d ($d \in \mathcal{D}$). Pas de frais supplémentaires dans l'intervalle $[b_d, e_d^1)$.
— $e_d^2 \in \mathcal{T}$: deuxième échéance de la demande d ($d \in \mathcal{D}$).
— tc_d^1 : coût unitaire de retard de la demande d satisfait à ou après e_d^1 ($d \in \mathcal{D}$).

— tc_d^2 : coût unitaire de retard de la demande d satisfait à ou après e_d^2 ($d \in \mathcal{D}$).
— lc_d : coût de vente perdu unitaire de la demande d ($d \in \mathcal{D}, lc_d > tc_d^1 + tc_d^2$).
— $\mathcal{D}^i \subseteq \mathcal{D}$: le sous-ensemble des demandes telles que $p_d = i$, i.e., $\mathcal{D}^i := \{d \in \mathcal{D} | p_d = i\}$.

Le problème est de décider pour chaque machine $r \in \mathcal{R}$ et dans chaque période $t \in \mathcal{T}$, la quantité à produire de chaque produit $i \in \mathcal{N}$. L'objectif est de minimiser le coût total, y compris les frais de vente perdus et le coût des retards. Le coût de setup est secondaire par rapport aux autres coûts. Les contraintes sont de trois types : premièrement, il existe des contraintes de capacité limitée cap_{rt} sur chaque machine $r \in \mathcal{R}$ et période $t \in \mathcal{T}$; deuxièmement, la production pour satisfaire la demande d ne peut commencer qu'à partir de sa date de début ; troisièmement, les contraintes concernant le report de setup. Les contraintes de setup exprime le fait que, pour produire le produit i sur la machine r pendant la période t, il doit y avoir un setup pour i sur r pendant t. Cependant, si le produit i est le dernier produit fabriqué dans la periode précédente $t-1$ sur la machine r, il n'y a plus besoin de setup pour produire le produit i sur la machine r pendant la periode t. Nous supposons qu'il existe au plus un setup par produit sur chaque machine pendant chaque période.

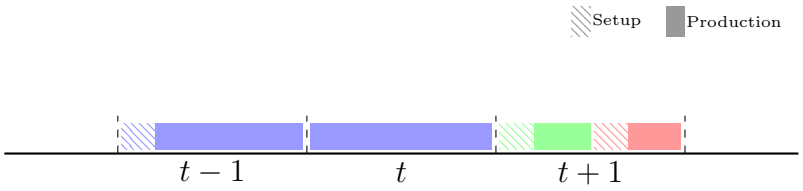


FIGURE 2.1 – Report de setup

Une pseudo-formulation peut servir à résumer le problème comme suit. Au meilleur de notre connaissance, c'est la première fois que ce problème de lot-sizing est étudié, nous l'appelons *CLSC* pour plus de simplicité.

$$
\begin{aligned}
(CLSC) \quad \min \quad & \textit{Coût des ventes perdues} + \textit{Coût de retard} \ \ (+ \textit{Coût de setup}) \\
s.t. \quad & \textit{Contraintes liées à la conservation du flux de matières premières} \\
& \textit{Contraintes de capacité des machines} \\
& \textit{Fenêtres temporelles des demandes} \\
& \textit{Report de setup}
\end{aligned}
$$

En se basant sur la définition, nous remarquons que le CLSC est différent du CLSP classique par rapport à la définition de la demande. Dans CLSP, les demandes sont

généralement agrégées par produits et par périodes. Ainsi, une demande est définie pour chaque produit dans chaque période. Toutefois, dans notre cas, il est important de tenir compte de la fenêtre temporelle individuelle de chaque demande en fonction de sa date de début et des dates d'échéance. Par conséquent, nous séparons le concept de produit et de demande. Chaque demande d concerne un seul produit p_d à produire en quantité q_d à une date r_d, accompagnée de deux dates d'échéance e_d^1, e_d^2 et leurs coûts de retard associés tc_d^1, tc_d^2 et lc_d. En conséquence, un produit peut être requis par un ensemble de demandes.

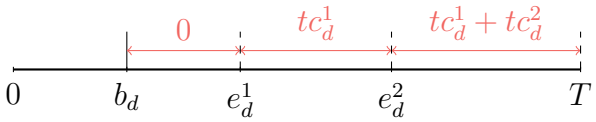


FIGURE 2.2 – Fenêtre Temporelle de la demande

Une autre différence concerne le stockage : il n' y a pas de problème de stockage, et donc pas de coût de stockage à considérer. Un produit fabriqué est directement utilisé pour satisfaire les demandes, c'est-à-dire que la livraison est immédiate.

Quatre formules de programmation linéaire mixte (MIP), appelées $Form1$, $Form2$, $Form3$ et $Form3^{FL}$, ont été développées pour modéliser CLSC. Dans ce résumé, nous présentons uniquement la formulation $Form3$.

$$\min \quad \sum_{d\in\mathcal{D}} lc_d y_d + \sum_{d\in\mathcal{D},t\in\mathcal{T}:t\geq e_d^1} tc_d^1 y_{dt} + \sum_{d\in\mathcal{D},t\in\mathcal{T}:t\geq e_d^2} tc_d^2 y_{dt} \quad (+ \sum_{i\in\mathcal{N},r\in\mathcal{R},t\in\mathcal{T}} sc_{ir} z_{irt}^+) \tag{2.1}$$

$$s.t. \quad \sum_{r\in\mathcal{R}} x_{irt} = \sum_{d\in\mathcal{D}^i,t\geq b_d} y_{dt} \qquad i\in\mathcal{N}, t\in\mathcal{T} \tag{2.2}$$

$$\sum_{b_d\leq t\in\mathcal{T}} y_{dt} + y_d = q_d \qquad d\in\mathcal{D} \tag{2.3}$$

$$\sum_{i\in\mathcal{N}} pt_i x_{irt} + \sum_{i\in\mathcal{N}} st_{ir} z_{irt}^+ \leq cap_{rt} \qquad r\in\mathcal{R}, t\in\mathcal{T} \tag{2.4}$$

$$x_{irt} \leq \Theta_{irt}(z_{irt}^0 + z_{irt}^+) \qquad i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T} \tag{2.5}$$

$$\sum_{i\in\mathcal{N}} z_{irt}^0 = 1 \qquad r\in\mathcal{R}, t\in\mathcal{T} \tag{2.6}$$

$$z_{irt}^0 \leq z_{ir,t-1}^0 + z_{ir,t-1}^+ \qquad i\in\mathcal{N}, r\in\mathcal{R}, t\in\tilde{\mathcal{T}} \tag{2.7}$$

$$z_{irt}^0 + z_{ir,t-1}^0 \leq 1 + w_{r,t-1} \qquad i\in\mathcal{N}, r\in\mathcal{R}, t\in\tilde{\mathcal{T}} \tag{2.8}$$

$$z_{irt}^+ + w_{rt} \leq 1 \qquad i\in\mathcal{N}, r\in\mathcal{R}, t\in\mathcal{T} \tag{2.9}$$

$$0 \leq x_{irt} \qquad i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T} \tag{2.10}$$

$$0 \leq y_{dt}, y_d \leq q_d \qquad d \in \mathcal{D}, b_d \leq t \in \mathcal{T} \tag{2.11}$$

$$z^0_{irt}, z^+_{irt} \in \{0,1\} \qquad i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T} \tag{2.12}$$

$$0 \leq w_{rt} \leq 1 \qquad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{2.13}$$

Pour comparer ces quatre formulations, nous avons prouvé de thèorème ci-dessous.

Theorem 2.1. *Les valeurs optimales des fonctions objectives des relaxations linéaires des formulations* $Form1$*,* $Form2$*,* $Form3$ *et* $Form3^{FL}$*, notées* $Obj^*_{LP}(Form1)$*,* $Obj^*_{LP}(Form2)$*,* $Obj^*_{LP}(Form3)$ *et* $Obj^*_{LP}(Form3^{FL})$*, vérifie*

$$Obj^*_{LP}(Form3^{FL}) \geq Obj^*_{LP}(Form3) = Obj^*_{LP}(Form2) \geq Obj^*_{LP}(Form1)$$

Pour pouvoir étudier expérimentalement le problème CLSC, nous générons deux ensembles d'instances de référence. L'un des ensembles est constitué d'instances réelles de l'application de fabrication de vêtements, tandis que l'autre provient d'un générateur d'instances pseudo-aléatoires conçu pour simuler des problèmes réels. Les instances de référence sont résumées dans le Tableau 2.1, dans lequel nous présentons le type d'instances (Type), sa notation (Notation), le nombre d'instances qu'il contient (Taille) et quelques commentaires. Les détails de chaque ensemble d'instances de benchmark sont donnés dans le Tableau 2.2 et le Tableau 2.3.

TABLE 2.1 – Résumé des instances de référence du CLSC

Type	Notation	Taille	Commentaire
Instances de l'application (IAP)	IAP-A	3	
	IAP-B	4	
Instances générées aléatoirement (IRG)	IRG-A	810	petite taille
	IRG-B	108	taille moyenne et grande

TABLE 2.2 – Instances de référence du CLSC provenant de l'application

Type	Instance	T	R	N	D	Γ(%)	Commentaire
IAP-A	R1	27	3	3	313	99	
	R2	36	28	18	1188	30	
	R3	30	29	1	595	33	
IAP-B	R5	25	30	46	668	91	
	R6	25	30	36	431	74	R5 avec horizon gelé
	R7	20	31	80	1428	40	
	R8	20	31	73	1404	41	R7 avec horizon gelé

TABLE 2.3 – Instances de référence du CLSC générées pseudo-aléatoirement

Notation	Taille	T	N	M	D	Γ(%)
IRG-A	810	{4,9,13}	{4,8,12}	{1,5,10}	{50,100,200}	{75,90}
IRG-B	108	{25}	{50,75,100}	{15,20,30}	{500,750,1000}	{75,90}

Tout d'abord, différentes formulations MIP sont comparées pour le CLSC avec et sans le coût de setup. Toutes les formulations développées sont résolues pour le benchmark IRG-A et IAP-A avec le solveur standard MIP CPLEX 12.6.1 et avec une limite de temps de 10 minutes. Le résumé des résultats est donné dans le tableau 2.4 et dans le tableau 2.5. Dans les tableaux, le temps de calcul est exprimé en secondes. Pour chaque paramètre d'instance (T, R, D, D, N, Γ) et pour chaque valeur, nous donnons les résultats moyens sur toutes les instances correspondantes. Dans la rangée T/A, les valeurs moyennes sur toutes les instances testées sont rapportées tandis que sa colonne Opt indique le nombre total d'instances résolues de manière optimale pour chaque formulation. Dans les deux premières colonnes, nous présentons les paramètres et leurs valeurs. Par exemple, pour le nombre de périodes T, il y a trois valeurs $4, 9, 13$ pour les instances IRG-A. Dans les colonnes Opt et Temps, nous rapportons le nombre et le temps de calcul moyen sur toutes les instances, avec les valeurs des paramètres donnés, résolues pour prouver l'optimalité dans le délai imparti. Dans les colonnes Nœuds et Écart, nous signalons le nombre de nœuds explorés et l'écart de sortie lorsque CPLEX termine. Cet écart représente la différence relative entre les limites primale et duale calculées par CPLEX à la date limite. Dans la colonne LPT, nous rapportons le temps de calcul moyen pour résoudre la relaxation linéaire sur toutes les instances partageant cette valeur de paramètre. En Colonne LPG, nous mesurons la qualité de la relaxation linéaire qui est calculée comme suit

$$LPG = \frac{BestMip - LPVal}{BestMip}, \tag{2.14}$$

où $bestMip$ est la meilleure solution entière (parmi toutes les formulations) et $LPVal$ est la valeur optimale de la relaxation linéaire de la formulation correspondante. Dans tous les cas sauf deux, les trois formulations $Form1$, $Form2$ et $Form3$ aboutissent à la même valeur $LPVal$ (dans les deux cas la différence est inférieure à 0,001 !). Nous ne présentons le LPG qu'une seule fois sous la colonne $Form3$. Pour les instances IAP-A, nous reportons, dans la colonne Obj, les valeurs de la fonction objectif obtenues par le solveur.

Nous observons que la formulation $Form3$ donne la meilleure performance globale. Par exemple, sur les 810 instances d'IRG-A avec coûts de setup, $Form3$ résout (avec preuve d'optimalité) 416 instances alors que $Form2$ n'en résout que 411 instances, $Form1$ 379

instances et $form3^{FL}$ 365 instances. Sur les 810 instances d'IRG-A sans coût de setup, $Form3$ résoud à l'optimalité 476 instances alors que $Form1$, $Form2$ et $Form3^{FL}$ n'en résolvent respectivement que 460, 468 et 424.

TABLE 2.4 – Comparaison des formulations CLSC avec coût de setup

		$Form1$					$Form2$					$Form3$						$Form3^{FL}$					
Para	Val	Opt	Temps	Noeuds	Écart	LPT	Opt	Temps	Noeuds	Écart	LPT	Opt	Temps	Noeuds	Écart	LPT	LPG	Opt	Temps	Noeuds	Écart	LPT	LPG
T	4	165	32	85970	4.64	0.0	175	35	49088	4.16	0.0	178	36	55347	3.89	0.0	31.01	158	39	18641	5.66	0.5	16.20
	9	123	69	41168	15.81	0.2	128	61	26469	15.28	0.2	131	65	31207	14.95	0.2	51.19	115	89	17907	18.98	1.6	35.92
	13	91	112	35839	24.17	0.5	108	102	20961	23.34	0.5	107	97	25667	23.13	0.4	59.98	92	105	16089	27.11	2.7	48.74
R	1	212	48	18748	2.70	0.0	217	55	12988	2.50	0.0	218	53	13563	2.75	0.0	65.30	226	48	21390	2.34	0.0	32.36
	5	97	68	50892	23.18	0.2	101	46	30784	22.46	0.2	105	58	37983	21.93	0.1	48.48	83	93	16101	27.17	1.1	41.29
	10	70	104	93338	18.74	0.5	93	91	52746	17.82	0.6	93	81	60674	17.29	0.5	28.40	56	136	15145	22.25	3.7	27.21
N	4	231	47	58790	1.40	0.1	258	47	25497	0.51	0.0	258	44	22985	0.59	0.0	35.07	212	59	9025	2.98	0.6	23.91
	8	87	95	64440	17.23	0.2	94	87	46402	16.48	0.2	94	86	55293	16.25	0.2	49.80	89	104	21291	19.95	1.6	35.04
	12	61	83	39746	25.99	0.4	59	81	24619	25.78	0.5	64	93	33943	25.12	0.4	57.31	64	68	22321	28.83	2.6	41.92
D	50	151	66	63995	11.75	0.2	156	53	35211	11.13	0.2	159	58	43104	10.64	0.1	46.81	149	56	23920	13.58	0.4	36.91
	100	126	56	47988	15.78	0.2	135	60	32839	15.06	0.3	137	57	37585	14.76	0.2	48.18	129	89	18361	17.67	1.0	33.79
	200	102	68	50994	17.09	0.3	120	72	28469	16.60	0.4	120	70	31532	16.56	0.3	47.19	87	73	10355	20.51	3.4	30.17
Γ	75	211	60	41299	14.53	0.2	226	50	23728	14.01	0.2	230	52	29214	13.60	0.1	49.07	206	67	13839	16.77	1.2	33.88
	90	168	68	67353	15.22	0.3	185	74	40617	14.51	0.3	186	72	45600	14.37	0.3	45.72	159	77	21252	17.74	2.0	33.36
T/A		379	63	54326	14.87	0.2	411	61	32173	14.24	0.3	416	61	37407	13.99	0.2	47.39	365	72	17546	17.25	1.6	33.62
Inst		Obj	Temps	Noeuds	Écart	LPT	Obj	Temps	Noeuds	Écart	LPT	Obj	Temps	Noeuds	Écart	LPT	LPG	Obj	Temps	Noeuds	Écart	LPT	LPG
R1		654,807	12	1827	0.00	0.1	654,807	2.6	434	0.00	0.0	654,807	3.1	404	0.00	0.0	14.25	654,807	34.7	469	0.00	0.5	14.03
R2		2,239,793	600	2	97.36	107.9	29,619,416	600	0	99.80	395.8	1,086,637	600	452	94.56	78.2	100	1E9	600	0	100	600	-
R3		10,672	0	0	0.00	0.0	10,672	0.0	0	0.00	0.0	10,672	0.0	0	0.00	0.0	0.00	10,672	2.1	0	0.00	0.9	0.00

TABLE 2.5 – Comparaison des formulations CLSC sans coût de setup

		$Form1$					$Form2$					$Form3$						$Form3^{FL}$					
Para	Val	Opt	Temps	Noeuds	Écart	LPT	Opt	Temps	Noeuds	Écart	LPT	Opt	Temps	Noeuds	Écart	LPT	LPG	Opt	Temps	Noeuds	Écart	LPT	LPG
T	4	205	33.0	57230	0.34	0.0	211	45.0	33768	0.32	0.0	214	39.4	36530	0.29	0.0	16.84	186	60.9	16830	0.58	0.2	6.9
	9	127	75.6	40526	5.10	0.1	123	40.1	26566	5.20	0.3	127	58.4	29364	5.11	0.2	25.58	104	47.2	16316	7.06	1.0	15.3
	13	128	35.8	31595	7.08	0.3	134	49.5	22915	6.92	0.6	135	42.4	25327	6.79	0.4	24.35	134	45.2	14521	9.73	1.5	17.2
R	1	214	37.2	18963	3.57	0.0	213	41.4	16839	3.81	0.0	215	37.8	17276	3.82	0.0	53.99	222	31.8	21773	3.43	0.0	25.3
	5	137	56.0	54654	5.44	0.1	145	59.5	29227	5.17	0.2	149	61.8	31683	5.05	0.2	8.78	119	84.4	13909	7.15	0.7	8.4
	10	109	48.7	55735	3.50	0.4	110	32.8	37183	3.46	0.7	112	37.7	42261	3.32	0.5	3.99	83	62.6	11985	6.79	2.0	5.7
N	4	219	24.8	33367	0.46	0.1	224	20.7	23845	0.35	0.0	227	29.3	23535	0.37	0.1	14.38	194	32.2	8974	1.33	0.4	8.1
	8	143	54.8	48945	4.49	0.2	147	54.0	30907	4.27	0.2	148	48.0	31840	4.24	0.2	23.91	130	68.4	17905	5.77	0.8	13.9
	12	98	78.3	47039	7.56	0.3	97	87.3	28498	7.83	0.7	101	77.5	35845	7.58	0.4	28.47	100	71.6	20788	10.28	1.6	17.4
D	50	153	44.1	59071	3.28	0.1	159	48.8	35406	3.11	0.3	161	44.4	40181	2.96	0.2	20.44	145	47.4	22467	3.76	0.3	14.8
	100	156	49.0	37713	3.97	0.2	155	45.8	26664	4.00	0.3	160	54.8	30210	3.92	0.2	22.93	143	47.4	15504	5.28	0.8	12.6
	200	151	43.4	32567	5.26	0.2	154	40.3	21179	5.34	0.4	155	36.5	20829	5.31	0.3	23.39	136	63.5	9696	8.32	1.7	12.0
Γ	0.75	299	36.3	34915	2.41	0.1	298	34.6	22200	2.43	0.2	299	29.0	25081	2.24	0.1	19.41	286	45.6	11490	2.95	0.5	9.3
	0.90	161	62.7	51319	5.93	0.3	170	63.2	33299	5.87	0.5	177	72.9	35733	5.89	0.4	25.10	138	66.9	20288	8.63	1.4	16.9
T/A		460	45.2	43117	4.17	0.2	468	44.7	27750	4.15	0.3	476	45.0	30407	4.06	0.2	22.25	424	52.8	15889	5.79	0.9	13.1
Inst		Obj	Temps	Noeuds	Écart	LPT		Temps	Noeuds	Écart	LPT		Temps	Noeuds	Écart	LPT	LPG		Temps	Noeuds	Écart	LPT	LPG
R1		442,906	0.5	51	0.00	0.1		0.3	14	0.00	0.0		0.3	8	0.00	0.1	0.83		19.5	400	0.00	0.4	0.83
R2		59,153	5.8	0	0.00	1.1		11.7	0	0.00	12.7		4.8	0	0.00	1.2	0.00		241.1	0	0.00	91.4	0.00
R3		10,672	0.1	0	0.00	0.1		0.1	0	0.00	0.0		0.1	0	0.00	0.0	0.00		2.2	0	0.00	1.3	0.00

A la lumière de ces résutats, nous choisissons donc la formulation la plus performante, à savoir $Form3$, pour résoudre les instances IRG-B et IAP-B à l'aide du solveur MIP standard CPLEX 12.6.1 (en exploitant tous les coeurs du processeur et avec un temps limite d'une heure). Le résumé des résultats obtenus sur les instances IRG-B est reporté dans le Tableau 2.6. L'écart relatif moyen constaté entre la valeur de la solution entière obtenue en 1 heure et la valeur de la relaxation linéaire sur les instances IAP-B est supérieur à 30%. En particulier pour l'instance R5, l'écart est supérieur à 91%.

Sur la base de ces tests préliminaires, nous concluons que la performance médiocre de CPLEX est dûe, d'une part, à la faiblesse de la relaxation linéaire et, d'autre part, à la taille élévée des intances, sur lesquelles l'heuristique intégrée de CPLEX n'arrive pas à trouver de bonnes solutions de départ. Par conséquent, des algorithmes heuristiques efficaces doivent être développés. Ceux-ci sont présentés dans le chapitre suivant.

TABLE 2.6 – Résultats computationnels : CPLEX sur IAP-B

	Characteristics					LP		MIP					
Inst	T	R	N	D	Γ	Obj	Temps	Obj	Temps	Écart	LB	#Noeuds	BestLB
R5	25	30	46	668	91	2,935,797	79	35,511,200	3600	91.6	2,973,702	0	2,973,702
R6	25	30	36	425	74	1,277,107	18	1,456,011	3600	7.7	1,344,501	97	1,344,501
R7	20	31	80	1428	40	2,217,260	118	2,692,957	3601	16.7	2,244,422	29	2,244,422
R8	20	31	73	1404	41	2,081,921	83	2,597,838	3600	18.7	2,111,181	0	2,111,181
AVG						2,128,021	74	10,564,501	3600	33.7	2,168,451	32	2,168,451

Chapitre 3

Problème complexe de lot-sizing à capacité finie : Heuristiques

Dans le chapitre précédent, nous avons présenté le problème complexe de lot-sizing CLSC. Nous montrons qu'il est NP-difficile et ne peut être résolu efficacement par un solveur MIP standard d'après nos expériences de calcul. Dans ce chapitre, nous proposons donc des algorithmes heuristiques afin de résoudre CLSC.

Pour l'application de fabrication de vêtements, l'indicateur clé de performance concerne le niveau de satisfaction de la demande. En d'autres termes, les coûts dûs au retard ou à la non-satisfaction de la demande sont beaucoup plus importants que le coût de setup. Notre objectif étant de résoudre le problème de planification de la production issu de l'application, nous considérons par la suite uniquement le CLSC sans coût de setup.

Tous les algorithmes heuristiques que nous avons développés sont basés sur la formulation MIP du problème. Comme montré précédemment, la formulation $Form3$ donne les meilleures performances globales. Par conséquent, nous utilisons $Form3$ pour développer et tester l'algorithme à chaque fois que la formulation MIP du CLSC est requise.

Nous proposons trois algorithmes heuristiques pour construire des solutions réalisables au CLSC : l'algorithme Fix&Relax, qui est une adaptation d'un algorithme classique largement utilisé pour résoudre CLSP, *l'algorithme basé sur la décomposition des produits* (PD) qui explore la structure des instances réelles, et *l'algorithme heuristique avec solution initiale* basé sur la relaxation LP. L'algorithme Fix&Relax, basé sur la décomposition par périodes et machines, résout une série de modèles MIP. De la même façon, l'algorithme PD est basé sur la décomposition du produit, et résout également une série de modèles MIP à plus petite échelle que ceux de l'algorithme Fix&Relax. Enfin, l'algorithme heuristique avec solution initiale est basé sur la fixation des variables et résout une série de

modèles LP. Par conséquent, toutes les heuristiques constructives utilisent des formulations mathématiques, mais avec des modèles et des tailles différentes.

Nous développons ensuite un algorithme de recherche locale pour améliorer la qualité de la solution. Fix&Optimize (F&O) algorithm [108] est une autre méthode couramment utilisée pour résoudre LSP. Partant d'une solution initiale, chaque itération consiste à fixer une partie des variables, tandis que les variables restantes sont optimisées pour essayer d'améliorer la qualité de la solution. Après chaque itération, les variables de la fenêtre de décision sont mises à jour et le processus est répété jusqu'à ce que certains critères soient atteints. La solution finale ne peut pas être pire que la solution initiale. L'idée est de résoudre un problème MIP plus petit à chaque itération pour trouver une meilleure solution.

Tous les algorithmes heuristiques à l'exception de l'algorithme F&R sont testés à la fois sur des instances d'application réelles IAP-B et sur des instances pseudo-aléatoires IRG-B. Les résultats de calcul obtenus sur les instances IAP-B, sont donnés dans le Tableau 3.1.

TABLE 3.1 – Résultats expérimentaux : algorithmes heuristiques sur IAP-B

	CPLEX			FSH		PD		FSH+FO		PD+FO	
Inst	Obj	Temps	Écart	Temps	Écart	Temps	Écart	Temps	Écart	Temps	Écart
R5	35,511,200	3600	91.6	216	79.0	74	48.2	662	14.2	478	35.8
R6	1,456,011	3600	7.7	64	76.7	7	45.1	492	5.9	181	19.0
R7	2,692,957	3601	16.7	376	74.6	315	40.9	823	9.2	756	12.0
R8	2,597,838	3600	18.7	316	67.8	285	49.2	762	6.9	734	17.7
MOY	10,564,501	3600	33.7	243	74.5	170	45.9	685	9.0	537	21.1

Tout d'abord, à l'exception de CLPEX, le temps de calcul suit toujours l'ordre PD $<$ FSH $<$ PD $<$ PD + F&O $<$ FSH + F&O sur ces 4 instances. Par contre, l'écart suit toujours l'ordre FSH $>$ PD $>$ PD $>$ PD + F&O $>$ FSH + FO. Deuxièmement, entre deux algorithmes heuristiques constructifs, il semble qu'un effort de calcul plus important ne mène pas à une meilleure qualité de solution. En moyenne, le temps de calcul de FSH est de 243 secondes alors qu'il est de 170 secondes pour l'algorithme PD. Toutefois, l'écart moyen de l'algorithme FSH est de 74,5%, soit presque le double de celui de l'algorithme PD (45,9%). Cependant, l'effort est payant lorsque l'algorithme constructif est suivi de l'algorithme d'amélioration. Avec le même mécanisme de recherche locale amélioré que l'algorithme F&O, FSH + F&O donne de meilleurs résultats que PD + F&O sur toutes les instances. Cela implique qu'une meilleure solution de départ ne signifie pas une meilleure solution finale pour l'algorithme F&O. Troisièmement, par rapport à CPLEX, l'algorithme FSH + F&O et PD + F&O parvient à fournir de meilleures solutions en un temps de calcul plus court. L'écart moyen de FSH + algorithme F&O est de 9,0% alors que l'écart

moyen de PD + F&O est de 21,1%. Pour CPLEX, l'écart moyen est de 33,7%. Surtout pour l'exemple le plus difficile R5, l'écart est réduit de 91,6% (CPLEX) à 14,2% (FSH + F&O). Ceci démontre l'efficacité de nos algorithmes heuristiques développés par rapport à CPLEX. Pour l'application de fabrication de vêtements, une solution acceptable est fournie dans un délai raisonnable ($<$ 12 minutes) par notre algorithme heuristique. Enfin, nous observons que l'algorithme F&O améliore relativement la qualité de la solution, surtout celle obtenue avec l'algorithme FSH. Par conséquent, l'algorithme F&O reste efficace pour résoudre LSP ainsi que pour de nombreux cas dans la littérature malgré la simplicité de sa structure.

Pour les deux benchmark de référence, tous les algorithmes heuristiques ont un comportement constant sur la qualité de la solution. En résumé, l'algorithme PD ou PD + F&O a l'avantage de la vitesse, et peut être utilisé lorsque le temps de calcul est une ressource rare. L'algorithme FSH et l'algorithme FSH + F&O ont un temps de calcul non négligeable, surtout lorsque la taille du problème devient trop grande. Cependant, parmi tous les algorithmes développés, ce sont eux qui retournent la meilleure solution.

Chapitre 4

Problème de lot-sizing avec une séquence fixe de produits

Dans de nombreuses industries manufacturières, le transfert de la production d'un produit à un autre entraîne des opérations de setup. Le setup consomme une quantité limitée de capacité des machines et/ou engendre un coût de setup. Lorsque le setup dépend de la séquence de production, c'est-à-dire lorsque le setup pour produire le produit actuel dépend à la fois de lui-même et du produit précédent, on parle de setup dépendant de la séquence [46,63]. Dans ce cas, il est nécessaire de prendre une décision pour le lot-sizing et le séquençage. La difficulté de ce problème réside dans le nombre factoriel de séquences de setup possibles. Toutefois, dans certaines industries manufacturières, ce nombre peut être réduit en utilisant les connaissances des planificateurs. Dans ce chapitre, nous étudions un cas particulier de CLSP avec setup dépendant de la séquence, appelé problème de lot-sizing à capacité finie et séquence fixe de produits.

Definition 4.1. *Étant données deux séquence* $\omega = \langle \omega_1, \omega_2, \ldots, \omega_n \rangle$ *et* $\alpha = \langle \alpha_1, \alpha_2, \ldots, \alpha_m \rangle$ *(*$m \leq n+1$*), on dit que* α *suit l'ordre de* ω*, noté* $\alpha \preceq \omega$*, si*

1. $\alpha_i \in \omega$ *pour tout* $i \in \{1, 2, \ldots, m\}$.
2. $\alpha_i \neq \alpha_j$ *pour tout* $i \neq j \in \{1, 2, \ldots, m\}$ *et* $\{i, j\} \neq \{1, m\}$.
3. *Soient* i *un indice tel que* $\omega_i = \alpha_1$ *et une séquence*

$$\beta(i) = \langle \omega_i, \omega_{i+1}, \ldots, \omega_n, \omega_1, \omega_2, \ldots, \omega_{i-1}, \omega_i \rangle \tag{4.1}$$

Il existe un sous-ensemble $\Omega' = \{\omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_{nm}}\}$ *tel que* $\langle \alpha_{\mathbf{1}}, \omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_{n_1}}, \alpha_{\mathbf{2}}, \omega_{i_{n_1+1}}, \omega_{i_{n_1+2}}, \ldots, \omega_{i_{n_2}}, \alpha_{\mathbf{3}}, \ldots, \alpha_{\mathbf{m}}, \omega_{i_{n_{m-1}+1}}, \ldots, \omega_{i_{nm}} \rangle$ *est égal à* $\beta(i)$.

Les paramètres du CLSP avec une séquence fixe de produits sont donnés comme suit :

— $\mathcal{N} = \{1, 2, \ldots, N\}$ un ensemble de N produits.
— $\mathcal{T} = \{1, 2, \ldots, T\}$ un ensemble de T périodes.
— cap_t : capacité de la machine sur la période t.
— d_{it} : demande de chaque produit i sur la période t.
— pt_i : temps de production unitaire de chaque produit i.
— hc_{it} : coût d'inventaire unitaire de chaque produit i sur la période t.
— b_{it} : la quantité maximale de production i qui peut être produite sur t.
— st_{ij} : temps de setup du produit i au produit j.
— sc_{ij} : coût de setup du produit i au produit j.
— Une permutation de $\mathcal{N}$: $\omega = \langle \omega_1, \omega_2, \ldots, \omega_N \rangle$.

Le problème CLSP avec une séquence fixe de produits, noté CLSP-FS1, consiste à décider de la séquence de production et de la quantité de production de chaque produit dans chaque période, de façon à ce que toutes les demandes soient satisfaites avec un coût total minimum, tout en respectant les capacités des machines. De plus, la séquence de setup choisie pour chaque période doit suivre l'ordre donné par ω.

Theorem 4.1. *CLSP-FS1 est fortement NP-difficile.*

Démonstration. Cette affirmation se démontre par réduction au CLSP. □

Diverses formulations MIP pour CLSP-FS1 sont proposées. Il y a deux types de décisions à prendre : le lot-sizing et le séquençage. Pour le lot-sizing, nous avons la formulation classique agrégée et la reformulation basée sur le problème de l'emplacement d'installations en CLSP. Pour le séquençage, nous proposons une formulation compacte orientée produit et, une formulation orientée séquence avec un nombre exponentiel de variables.

En fonction d'une séquence de produit, si la production suit la séquence donnée, le setup est mineur. Cependant, lorsque nous avons besoin d'inverser les produits dans la séquence, le setup devient majeur. Ici nous étudions un cas extrême où le setup mineur vaut zéro et où le setup majeur est un nombre positif. Plus précisément, nous définissons

$$st_{\omega_i,\omega_j} = \begin{cases} 0 & i \leq j \\ \Delta_{st} & sinon \end{cases} \qquad sc_{\omega_i,\omega_j} = \begin{cases} 0 & i \leq j \\ \Delta_{sc} & sinon \end{cases}$$

où $\Delta_{st} > 0$ et $\Delta_{sc} > 0$. Sans perte de généralité, on peut supposer que la séquence fixe est $\langle 1, 2, \ldots, N \rangle$ par réindexation. Les matrices de setup $\{st_{ij}\}_{i,j\in\mathcal{N}}$, $\{sc_{ij}\}_{i,j\in\mathcal{N}}$ ont alors la structure suivante :

$$\begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ \Delta_{st} & 0 & 0 & \dots & 0 & 0 \\ \Delta_{st} & \Delta_{st} & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ \Delta_{st} & \Delta_{st} & \Delta_{st} & \ddots & 0 & 0 \\ \Delta_{st} & \Delta_{st} & \Delta_{st} & \dots & \Delta_{st} & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ \Delta_{sc} & 0 & 0 & \dots & 0 & 0 \\ \Delta_{sc} & \Delta_{sc} & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ \Delta_{sc} & \Delta_{sc} & \Delta_{sc} & \ddots & 0 & 0 \\ \Delta_{sc} & \Delta_{sc} & \Delta_{sc} & \dots & \Delta_{sc} & 0 \end{bmatrix}$$

Nous faisons référence à ce cas particulier de CLSP-FS1 par la notation CLSP-FS1-LT puisque ses matrices de setup forment des triangles inférieurs. Dans le théorème suivant, CLSP-FS1-LT est montré comme étant NP-difficile.

Theorem 4.2. *CLSP-FS1-LT est NP-difficile.*

Démonstration. La preuve est basée sur une réduction à CLSP avec un seul produit. □

En raison de la structure particulière, nous simplifions la formulation du MIP. Par rapport à la formulation orientée produit du problème général, le nombre de variables binaires est réduit de $O(N^3T)$ à $O(NT)$.

Dans la suite, nous effectuons une étude expérimentale. Nous comparons quatre formulations que nous avons développées : AG-SO, AG-PO, FL-SO, FL-PO. Les instances de référence sont 10 instances de [63], avec seulement les 10 premiers produits et 10 périodes. De plus, aucune vente perdue n'est considérée. Par conséquent, toutes les demandes doivent être satisfaites. Le résumé des résultats est donné dans le tableau 4.1. D'après les résultats moyens sur 10 cas pilotes, nous observons que la formulation FL-PO donne l'écart moyen le plus faible à 2,1%. D'autre part, la formulation AG-SO donne l'écart le plus grand avec 3,8%. De plus, la formulation FL-PO possède également le temps de calcul le plus court par rapport aux autres formulations. Pour la borne inférieure donnée par la relaxation linéaire, la formulation FL-SO donne la meilleure borne tandis que la formulation AG-PO donne la plus mauvaise borne inférieure. Il semble y avoir une relation de dominance entre les relaxations linéaires de ces différentes formulations.

TABLE 4.1 – Comparaison des formulations : résulats computationnels

	MIP								LP	
Inst	Obj	Temps	Écart	LB	Cols	Bin	Lignes	Noeuds	Obj	Temps
AG-SO	42921	600	3.8	37383	102501	102300	320	15448	11486	6
FL-SO	42200	601	2.2	40516	102851	102300	770	27462	39991	15
AG-PO	42277	581	2.4	40612	6712	6511	1312	159063	9038	0
FL-PO	42178	560	2.1	41188	7062	6511	1762	154882	37138	0

Chapitre 5

Conclusion générale et travaux futurs

Dans ce manuscrit, nous avons présenté nos recherches motivées par des applications réelles. Nous pouvons résumer les principaux résultats en deux parties.

Dans la première partie du manuscrit, nous avons étudié un problème de planification de la production qui nous a été soumis par un projet de fabrication de vêtements. Nous avons conçu un outil logiciel optimisé pour répondre efficacement à ce problème industriel. Un cadre de décomposition a été développé, qui résout successivement un modèle agrégé et un modèle détaillé.

Le problème agrégé, noté CLSC, s'avère être le goulot d'étranglement de l'approche et a été étudié sous différents angles. Le CLSC correspond à un problème complexe de lot-sizing à capacité finie, et il a été démontré comme étant NP-difficile, même sans les coûts de setup. Plusieurs formules de programmation linéaire mixtes (MIP) sont développées pour le CLSC. Afin d'évaluer expérimentalement les différentes formulations MIP, deux ensembles d'instances de référence ont été conçus. Le premier ensemble est constitué de données réelles, tandis que le deuxième ensemble est constitué d'instances pseudo-aléatoires avec des caractéristiques réalistes et des propriétés différentes. Grâce à ces évaluations expérimentales approfondies, nous avons pu constater qu'une formulation donne de meilleurs résultats que les autres.

Les instances de taille moyenne peuvent être résolues directement à l'aide de CPLEX, mais l'optimalité ne peut pas être prouvée pour les instances de grande taille en un temps de calcul court. Par conséquent, plusieurs algorithmes heuristiques efficaces sont développés à partir de phases constructives et améliorés par des phases de recherche locale.

Nous avons conçu un algorithme heuristique Fix & Relax (F&R) basé sur la relaxation de la programmation linéaire (LP) des formulations compactes. Cette heuristique fournit

des solutions de bonne qualité, mais elle nécessite des temps de calcul importants. Ensuite, afin d'obtenir des solutions de bonne qualité en un temps de calcul court, nous avons conçu un algorithme de décomposition des produits (PD) basé sur l'observation que 20% des familles de produits couvrent 80% des demandes (sur les instances réalistes). Nous avons fait l'expérience d'un compromis naturel entre la qualité des solutions et le temps de calcul en comparant les performances de F&R et PD. De plus, une heuristique constructive est développée, appelée First Solution Heuristic (FSH). L'algorithme FSH est basé sur la relaxation linéaire du modèle compact et la fixation des variables dans le but de construire des solutions réalisables de bonne qualité. Grâce à des expériences de calcul intensives, nous avons pu constater que l'algorithme PD est plus performant que l'algorithme FSH en terme de temps de calcul et de qualité de solution (pour les instances considérées). Cependant, nous avons observé que l'effet positif de la phase de recherche locale est plus fort pour l'algorithme FSH que pour l'algorithme PD.

Enfin, la combinaison FSH et F&O nous permet d'atteindre la meilleure performance globale. Dans la pratique, un écart d'optimalité maximum de 15% est observé entre les solutions réalisables et les valeurs optimales des relaxations linéaires. Ces résultats surpassent l'écart d'optimalité de CPLEX appliqué directement sur les formulations compactes, qui est supérieur à 90% en moyenne si l'on considère le même temps CPU d'une heure. En ce qui concerne les instances générées aléatoirement, si l'on compare la solution de CPLEX et celle de l'algorithme FSH + F&O, on constate une amélioration de leur qualité de 85%. En ce qui concerne les autres heuristiques, elles sont également plus performantes que CPLEX pour ce qui est du calcul des solutions réalisables en peu de temps. Tous les algorithmes heuristiques développés ont été intégrés dans l'outil de planification de production de DecisionBrain, améliorant ainsi l'efficacité du système d'optimisation.

Dans la deuxième partie du manuscrit, nous avons étudié une version restreinte du problème de lot-sizing à capacité finie et setup dépendant de la séquence, où les séquences de setup pour chaque période doivent suivre l'ordre d'une séquence donnée. Ce problème est appelé problème de lot-sizing à capacité finie et séquence de produit fixe (CLSP-FS1). Ce problème vient d'une application du monde réel. Par rapport au problème de lot-sizing à capacité finie et setup dépendant de la séquence, CLSP-FS1 réduit le nombre de séquences candidates de $O(n!)$ à $O(n2^n)$. Dans de nombreuses applications du monde réel, une séquence "idéale" est connue et seules les séquences suivant cet ordre peuvent être choisies. Il est démontré que ce problème est de type NP-difficile. Quatre modèles MIP sont développés à partir de formulations orientées séquence et produit (setup). Nous avons effectué des tests computationnels préliminaires pour comparer ces formulations à une reformulation classique. Nous avons observé qu'une nouvelle formulation proposée

garantit la meilleure performance globale pour les instances de référence testées. Pour la formulation orientée séquence, une heuristique simple de génération de colonne a été développée et testée. Même si la qualité de la relaxation linéaire de cette formulation est meilleure que celle de la formulation compacte, les solutions réalisables calculées par l'heuristique de génération de colonne sont moins bonnes que celles calculées par CPLEX en utilisant la formulation compacte. De plus, nous avons étudié un cas particulier de CLSP-FS1, dont les matrices de setup ont une forme triangulaire inférieure, noté CLSP-FS1-LT. Pour cette variante du problème, si le plan de production suit l'ordre de la séquence donnée, aucun setup n'est effectuée. Toutefois, si le plan de production inverse l'ordre des produits dans la séquence donnée, une coût de setup important doit être payée. Par conséquent, la décision cruciale concerne l'exécution du setup dominant (non nulle). De plus, cette variante de problème s'avère être NP-difficile et une formulation MIP sur mesure est développée. Comparativement à la formulation orientée produit avec des $O(N^3T)$ variables binaires, la formulation MIP sur mesure de CLSP-FS1-LT ne contient que $O(NT)$ variables binaires.

Pour résumer cette thèse, nous avons abordé des problèmes complexes de planification de la production et nous avons conçu des modèles mathématiques avancés et des algorithmes heuristiques efficaces. Ces outils nous permettent de calculer des solutions réalisables de bonne qualité, mais plusieurs pistes de recherche restent ouvertes.

En ce qui concerne le problème du CLSC, nous avons observé un saut d'intégrité important, de plus de 50% sur les instances difficiles IRG-B. Par conséquent, d'autres familles d'algorithmes heuristiques peuvent être développées à partir de formulations renforcées. Plus en détail, une ligne de recherche prometteuse peut être l'étude de la reformulation en réseau du CLSC, ou une formulation hybride MIP pourrait aussi être développée à partir de la littérature récemment proposée.

En ce qui concerne le problème CLSP-FS1, notre étude offre un premier regard sur ce problème et seules des expériences préliminaires ont été menées à ce jour. Par conséquent, nous aimerions tester les formulations développées sur des instances de plus grande échelle, en particulier sur les instances structurées où notre modèle s'applique, et elles devraient être comparées avec le modèle dépendant de la séquence classique.

Enfin, l'étude présentée au chapitre 2 a fait l'objet d'une publication en conférence [48]. Nous avons également lancé un autre projet concernant le problème de *Temporal Bin Pacing* (TBPP). Il s'agit d'une extension du problème de bin packing, où les produits consomment la capacité du bin pendant une période de temps seulement. Une formulation polynomiale et une formulation extensive sont étudiées. De plus, divers algorithmes heuristiques sont développés et comparés, dont l'heuristique de type gloutonne et une

heuristique basée sur la génération de colonnes. L'étude du TBPP a aussi fait l'objet d'une publication en conférence [49]. Comme TBPP ne concerne pas la planification de la production, nous ne l'avons pas incluse dans ce manuscrit.

Résumé

Dans cette thèse, nous étudions deux problèmes de planification de production motivés par des applications du monde réel. Tout d'abord, un problème de planification de production pour un projet de fabrication de vêtements est étudié et un outil d'optimisation est développé pour le résoudre. Deuxièmement, nous étudions un problème particulier de dimensionnement de lots de production avec contraintes de capacités et de paramétrages des machines dépendantes de la séquence produite. Diverses formulations mathématiques sont développées et une analyse de complexité est effectuée pour donner une première analyse du problème.

Abstract

In this thesis, we study two production planning problems motivated by challenging real-world applications. First, a production planning problem for an apparel manufacturing project is studied and an optimization tool is developed to tackle it. Second, a restricted version of the capacitated lot sizing problem with sequence dependent setups is explored. Various mathematical formulations are developed and complexity analysis is performed to offer a first glance to the problem.

Mots Clés

Planification de Production,
Problème de lot-sizing,
Programmation linéaire mixte
Heuristiques

Keywords

Production Planning,
Lot Sizing Problem,
Mixed Integer Programming,
Heuristics