October 2019

Three Essays on Data-Driven Optimization for Scheduling in Manufacturing and Healthcare

Ekin Koker

THREE ESSAYS ON DATA-DRIVEN OPTIMIZATION FOR SCHEDULING IN MANUFACTURING AND HEALTHCARE

A Dissertation Presented

by

EKIN KOKER

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

SEPTEMBER 2019

Industrial Engineering & Operations Research

THREE ESSAYS ON DATA-DRIVEN OPTIMIZATION FOR SCHEDULING IN MANUFACTURING AND HEALTHCARE

A Dissertation Presented

by

EKIN KOKER

Approved as to style and content by:

_____

Hari Balasubramanian, Co-Chair

_____

Ana Muriel, Co-Chair

_____

Anna Nagurney, Member

_____

Sundar Krishnamurty, Department Head

Mechanical and Industrial Engineering

DEDICATION

This dissertation is dedicated to my partner and soul mate, Mari Lentz.

ACKNOWLEDGMENTS

I would like to say thank you to everyone who has supported me through this journey.

My advisors, Prof. Ana Muriel and Hari Balasubramanian, who have been the most amazing guides, both in research and in life. They are the best advisors any PhD student could ever wish for.

Prof. Anna Nagurney, my dissertation committee member, who provided valuable feedback for this dissertation and was supportive of all my academic efforts.

Ted Acworth and Mike Trachtman at Artaic, the industry collaborators who provided funding for a research project that became the first essay of this dissertation through Center for e-Design at UMass. The views expressed in this dissertation are of the authors alone and not of Artaic or Center for e-Design.

National Science Foundation (NSF) who provided funding for another research project that became the second essay of this dissertation through Grant #1254519. The views expressed in this dissertation are of the authors alone and not of the National Science Foundation. I also would like to acknowledge Joanne Alvarez-Oh of Quinnipiac University, whose earlier work I have built upon in this essay.

Vivek Saxena and Josh Kuledge at Advisory Aerospace, the industry collaborators who provided the research project that became the third essay of this dissertation.

A shout-out to my fellow PhD friends, Deniz Besik, Pritha Dutta, Destenie Nock and Rodrigo Mercado Fernandez is well deserved. Without my peers this process would be much harder.

I would like to extend my sincere gratitude to my parents and my brother, Birgul, Cetin and Baris Koker, who supported me throughout my entire life. I would not be where I am without their help.

Finally, I would like to acknowledge my partner and soul mate, Mari Lentz and our cat, Artemis. You were my family when it counted, and you always will be. This dissertation would not have been possible without you. I love you both.

ABSTRACT

THREE ESSAYS ON DATA-DRIVEN OPTIMIZATION FOR SCHEDULING IN MANUFACTURING AND HEALTHCARE

SEPTEMBER 2019

EKIN KOKER, B.S., ISTANBUL TECHNICAL UNIVERSITY

M.S., SABANCI UNIVERSITY

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Ana Muriel and Hari Balasubramanian

This dissertation consists of three essays on data-driven optimization for scheduling in manufacturing and healthcare. In Chapter 1, we briefly introduce the optimization problems tackled in these essays. The first of these essays deals with machine scheduling problems. In Chapter 2, we compare the effectiveness of direct positional variables against relative positional variables computationally in a variety of machine scheduling problems and we present our results. The second essay deals with a scheduling problem in healthcare: the team primary care practice. In Chapter 3, we build upon the two-stage stochastic integer programming model introduced by Alvarez Oh (2015) to solve this challenging scheduling problem of determining patient appointment times to minimize a weighted combination of patient wait and provider idle times for the team practice. To overcome the computational complexity associated with solving the problem under the large set of scenarios required to accurately capture uncertainty in this setting, our approach relies on a lower bounding technique based on solving an exhaustive and mutually exclusive group of scenario subsets. Our computational results identify the structure of optimal schedules and quantify the impact of nurse flexibility, patient crossovers and no-shows. We conclude with practical scheduling guidelines for team primary care practices. The third essay deals with another scheduling problem observed

in a manufacturing setting similar to first essay, this time in aerospace industry. In Chapter 4, we propose mathematical models to optimize scheduling at a tactical and operational level in a job shop at an aerospace parts manufacturer and implement our methods using real-life data collected from this company. We generalize the Multi-Level Capacitated Lot-Sizing Problem (MLCLSP) from the literature and use novel computational techniques that depend on the data structure observed to reduce the size of the problem and solve realistically-sized instances in this chapter. We also provide a sensitivity analysis of different modeling techniques and objective functions using key performance indicators (KPIs) important for the manufacturer. Chapter 5 proposes extensions of models and techniques that are introduced in Chapters 2, 3 and 4 and outlines future research directions. Chapter 6 summarizes our findings and concludes the dissertation.

keywords: Machine scheduling, computational comparison, mathematical modelling, appointment scheduling, team primary care practice, two-stage stochastic integer programming, multi-level capacitated lot-sizing, hierarchical job shop scheduling, aerospace manufacturing

CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

This dissertation consists of three essays on data-driven optimization for scheduling in manufacturing and healthcare. Scheduling has been a focus of interest in operations research literature for decades. In this work, we built upon existing mathematical models and also develop new ones for a variety of problems both in deterministic and stochastic settings. In particular, these research problems rise from real-life scenarios that occur in manufacturing and healthcare environments. The need for optimal schedules in these settings allowed us to collaborate and collect data from real-life practitioners and come up with models, tools and guidelines that are driven by this data. The research problems themselves are novel and challenging and we hope that the research presented here will be useful for other practitioners in similar or different fields facing similar problems. The dissertation is organized as follows:

In Chapter 2, we present two different types of modeling techniques for two different problem families in machine scheduling: parallel machine and flexible flow shop. The modeling techniques consist of different types of decision variables: one of them, we refer to as direct positional variables while the other one we refer to as relative positional variables. Direct positional variables assign jobs to different positions in different machines while the relative positional variables determine whether one job comes before or after another. Hence, the positions of jobs are decided relative to one another. We compare these modeling techniques computationally using randomly generated instances and we present our findings. The models are based on a problem observed at a manufacturing company. We also collected data from this company and created random

instances based on this data. Finally, we implemented the most practically relevant model as initially an Excel tool that can be used by anyone and then in a cloud environment used by the company.

In Chapter 3, we build upon Alvarez Oh (2015)'s work on a healthcare scheduling problem: the team primary care practice. Team primary care practice scheduling problem consists of two nurses flexibly seeing patients before they are seen by their dedicated provider. Alvarez Oh (2015) proposes a stochastic integer programming model that minimizes a weighted combination of patients' wait time and providers' idle time. This system can be defined as a tandem queue. The problem is novel because the First-Come-First-Serve (FCFS) structure in the second stage (provider) creates a modeling complexity which is called patient crossover. FCFS allows the patients that are scheduled to be seen later during the session to be seen earlier if they complete before the patients that are scheduled earlier. Hence the name patient crossover, since a patient crosses over another one in schedule. This structure is not trivial to model and she overcomes this using a second-largest logic exploiting the special circumstances that are created by having two nurses and two providers. She then generates and solves random instances with 1000 scenarios and comes up with practical insights that can be used by primary care practices. We build upon her work by generalizing her insights using newly created problem instances with a different distribution, tightening the optimality gap and allowing one to solve for larger problem instances using a lower-bounding scheme, analyzing the effects of problem characteristics such as the value of stochastic solution, cost ratio, nurse flexibility, service time variability and patient crossover. This Chapter is based on our work in Alvarez Oh et al. (2018).

In Chapter 4, we propose mathematical models to optimize scheduling at a tactical and operational level in a job shop at an aerospace parts manufacturer and implement our methods using real-life data collected from this company. Aerospace parts manufacturing involves highly complex Bill-of-Materials (BOM) structures with many intermediate and end products. Another complication observed at a manufacturing setting is the limited amount of resources available at the job shop. Production and inventory levels of each item at different time periods must be determined and setup times must also be considered. Therefore, we generalize the Multi-Level Capacitated Lot-Sizing Problem (MLCLSP) from the literature by solving it at first in tactical level (planning stage) and then in operational level (execution stage). The problem thus becomes Hierarchical Job Shop Scheduling Problem (HJSP) introduced here for the first time. Planning and execution problems have different time horizons and different requirements, but they are also connected because planning problem's output becomes execution problem's input. In particular, we introduce large setups (that take longer than a day) for the first time and consider due dates, backlogs and lost sales. We also use novel computational techniques that depend on the data structure to reduce the size of the problem to solve realistically-sized instances in this chapter. Finally, we provide a sensitivity analysis of different modeling techniques and objective functions using key performance indicators (KPIs) important for the manufacturer.

In Chapter 5, we propose extensions to what we have accomplished in Chapters 2, 3 and 4 and outline future research directions. In particular, the modeling techniques that are described in Chapter 2 can be applied to more machine scheduling problem families and further computational studies can be done using more random problem instances.

Moreover, the problem described in Chapter 3 can be further generalized by allowing flexibility in the second stage (provider) and relaxing the homogeneity assumption on patients, thus allowing different patient types to be scheduled and sequenced using a single optimization model. Finally, the problem in Chapter 4 can be further generalized by considering machine availability/maintenance requirements, computational findings can be strengthened by doing a full computational study using benchmark instances or heuristics from the literature can be compared to our exact methods.

We summarize our contributions and conclude the dissertation in Chapter 6.

CHAPTER 2

COMPUTATIONAL COMPARISON OF DIRECT POSITIONAL VARIABLES AGAINST RELATIVE POSITIONAL VARIABLES IN MACHINE SCHEDULING PROBLEMS

My advisor Ana Muriel and our industry partners Mike Trachtman and Ted Acworth from Artaic have collaborated with me in this project and contributed to the work described in this essay.

2.1. Introduction

Machine scheduling problems are one of the most widely studied problem families in operations research. The roots of the problem go as far back as to late 19^{th} and early 20^{th} century when the industrial revolution began. With the introduction of factories and automation into our lives, the need for planning the shop floor arose. To be able to scale the businesses into sustainable levels and make a profit, it was crucial to be as efficient as possible with the usage of time. This also corresponded to the rise of operations research as a discipline, right after World War 2. It was a prosperous era and customers were demanding products faster and faster. Therefore, scheduling became the focus of operations research, and in particular, machine scheduling.

However, the problem turned out to be challenging and most of the literature on machine scheduling has focused on developing heuristics, testing meta-heuristics or exploring the effectiveness of dispatching rules. Even though the prominence of computers allows us to solve problems of larger scale today, the tradition of machine scheduling continued as it is, mostly avoiding exact methods.

Therefore, comparison of different modeling techniques in exact approaches such as mixed integer programming received surprisingly little attention in the machine scheduling literature. In this essay, we compare the effectiveness of direct positional variables against relative positional variables computationally in a variety of machine scheduling problems such as parallel machine and flow shop scheduling and we present our results.

The motivation behind our study is Artaic, a custom mosaic design studio and manufacturer in Boston, MA. They use robotic fabrication, which allows for fast, flexible and accurate assembly of unique tile work for their customers. They have two production stages. The first one is tile tubing, which groups different colored tiles into tubes that feed the tiles to the second stage, which is automated tile assembly. In this stage, the unique design is loaded up into the computer, which is translated into schematics for the robotic hand to build. The product is then packaged and shipped to the customer. The process can be summarized in Figure 2.1.1 below:

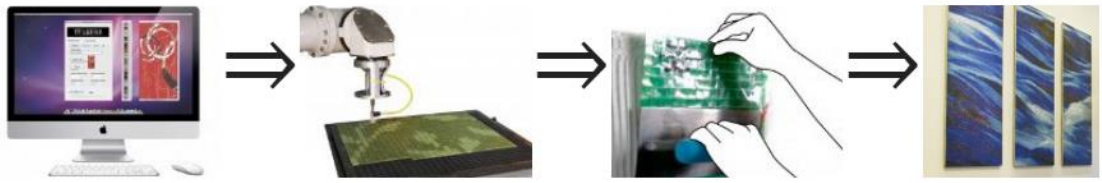


Figure 2.1.1 Production Process at Artaic[1]

The second stage is the bottleneck of their production and can be modeled as a parallel machine scheduling problem. Explicitly including the first stage makes the problem a Flexible Flow Shop. That is why our focus is on mathematically modelling these two problems and coming up with efficient exact methodologies to solve them. Because the jobs at Artaic have due dates, our objective will be to minimize a weighted combination of makespan (to reduce inventory) and total tardiness (to increase customer satisfaction).

The remainder of the essay is organized as follows: in Section 2.2, we briefly summarize the literature. In Section 2.3, we present the mathematical models. In Section 2.4, we present our computational study and the results. We conclude the essay in Section 2.5.

2.2. **Literature Review**

In our review, we will only focus on papers that compare exact approaches, due to the enormity of the machine scheduling literature.

Stafford, Tseng and Gupta (2005) model Permutation Flowshop Scheduling Problem using 2 mixed integer linear program families, which consist of 3 and 5 models, respectively. The families are called Wagner and Manne, named after the early

[1] Image from http://hizook.com/blog/2012/08/02/artaic-revolutionizing-tile-mosaics-through-robotic-assembly

researchers proposing the modeling approaches. In our terminology, the Wagner family of models corresponds to direct positional variables and the Manne family of models corresponds to relative positional variables. They only consider makespan as their single objective function. Their main purpose is to summarize the models found in the literature and to present new models, compare problem size complexity, solve a common set of problems to resolve conflicting results found in the literature and explore the limits of solvable problems using new technologies. Even though the permutation flowshop differs from our tubing model, which is flexible flowshop, their findings are in parallel with ours, which is that the direct positional variables (Wagner family of models) dominate relative positional variables (Manne family of models).

Keha, Kowala and Fowler (2008) model Single Machine Scheduling Problem using 4 different mixed integer programs, 2 of which use relative positional variables. One of the remaining 2 is a direct positional model and the other one is a time-indexed model. They consider weighted completion time, maximum lateness, number of tardy jobs and weighted tardiness as their objective functions. They also consider release dates as a factor in their computational study. For weighted completion time, they find that one of the two relative positional models called linear ordering performs the best. They find that the performance of various formulations depends on a lot of different factors, e.g. the range of processing time. Because the direct positional model seems to be promising, they improve upon their formulation by adding valid inequalities and present the results. They believe that an expert in integer programming may prefer this formulation to investigate further using a method such as branch and cut. They also find that even

though the time-indexed and linear ordering (relative positional) variables are used most widely in the literature, the direct positional variables seems to have potential.

Unlu and Mason (2010) model Parallel Machine Scheduling Problem using 4 different mixed integer programs, 2 of which use relative positional variables (called Network model and Linear Ordering Model in the paper). One of the remaining 2 is direct positional and the other one is time-indexed. Even though they propose models for non-identical and unrelated parallel machine scheduling problems and a variety of different objectives, they consider only makespan and weighted completion time as the objective functions and only identical parallel machine in their computational study. They also consider ready-times (release dates) in the computational study. They have a lengthy analysis for each objective function/number of parallel machines/existence of release dates combination, but to summarize, they believe the time-indexed and one of the relative positional models is superior to others. Therefore, according to their results, the direct positional model seems inferior which is contradicting our findings so far. However, our study differentiates from theirs since we consider unrelated parallel machines, setup times and due dates, the latter important since part of our objective function is based on due dates and they do not consider them at all.

Demir and Isleyen (2013) model the Flexible Job Shop Scheduling Problem (FJSP) using 5 different mixed integer programs, 3 of which use relative positional variables. One of the remaining 2 is a direct positional model and the other one is a time-indexed model, which they propose for the first time for FJSP with unrelated parallel machines. Even though they do a literature review of mathematical models for FJSP which include a variety of different objectives, they consider only makespan as the objective function in

their computational study. Similarly, even though they mention setup times in the literature review, they do not consider it in their models and their computational study. Out of 5 models they consider, only one of the relative positional models is linear and the remaining 4 models including direct positional model is nonlinear. The only linear model, which is a relative positional model, outperforms the other ones in solution time and optimality gap. Their newly-proposed, time-indexed model is outperformed by the other ones. FJSP is similar to our model, which is FFSP, that explicitly considers tubing stage.

Yu and Hung (2016) model the Parallel Machine Scheduling problem using 3 different mixed integer programs, all of which use relative positional variables. Their objective is to minimize total tardiness in the existence of ready dates. They find that one of these models that is enhanced with assignment variables (assigning jobs to machines), performs the best. Since they do not consider direct positional variables at all, their results are not directly comparable to ours.

Our contribution to the literature is thus four-fold: 1) above papers consider only one problem family (single machine, job shop, flowshop or parallel machine) while our essay encompasses multiple problem families, 2) we use makespan and tardiness as a combined objective, as opposed to the typical use of single objectives for the most of the literature, 3) Sequence-dependent setups are considered as a complicating factor, which is not considered in most of the comparison literature and 4) Our models are grounded and tested based on a real-life application, which is not the case in most of the literature.

2.3. **Mathematical Models**

2.3.1. **Parallel Machine Models**

2.3.1.1. **Direct Positional Model**

2.3.1.1.1. **Sets**

J = 1..n	Jobs
I = 1..m	Machines
K = 1..n	Positions on the machines

2.3.1.1.2. **Parameters**

n =	number of jobs
m =	number of machines
d_j =	Due date of job j
$a_{j,i}$ =	1 if job j can be done on machine i, 0 otherwise
$p_{j,i}$ =	Processing time of job j on machine i
c_j =	Penalty for tardiness of job j
s_j =	Earliest start time of job j
$M = Max_j(s_j) + \sum_{i,j} p_{j,i}$	A very large number

2.3.1.1.3. **Variables**

$C_{\max}$ =	Makespan - completion time of all jobs
$t_{k,i}$ =	Completion time of job at positions k on machine i, $k = 0..n$

D_j = Tardiness of job j

$x_{j,k,i}$ = 1 if job j is assigned to position k on machine i, 0 otherwise

2.3.1.1.4. **Model**

$$minimize\ C_{\max} + \sum_j c_j * D_j$$

$subject\ to$

$Cmax \geq t_{k,i}\ \forall\ k \in K, i \in I$ (1)

$t_{k,i} \geq t_{k-1,i} + \sum_j p_{j,i} * x_{j,k,i}\ \forall\ k \in K, i \in I$ (2)

$t_{k,i} \geq (s_j + p_{j,i}) * x_{j,k,i}\ \forall\ j \in J, k \in K, i \in I$ (3)

$t_{k,i} \leq d_j + M * (1 - x_{j,k,i}) + D_j\ \forall\ j \in J, k \in K, i \in I$ (4)

$\sum_{i,k} a_{j,i} * x_{j,k,i} = 1\ \forall\ j \in J$ (5)

$t_{0,i} = 0\ \forall\ i \in I$ (6)

$\sum_{i,k} x_{j,k,i} \leq 1\ \forall\ j \in J$ (7)

$\sum_j x_{j,k,i} \leq 1\ \forall\ i \in I, k \in K$ (8)

$\sum_j x_{j,k,i} \geq \sum_j x_{j,k+1,i}\ \forall\ i \in I, k \in 1..n-1$ (9)

Objective function is to minimize a weighted combination of makespan and tardiness. Constraint 1 ensures that the makespan is after all jobs on all machines are complete. Constraint 2 calculates completion time of the job j at position k on machine i as after completion time of previous position $k-1$ plus the processing time of job j on machine i. Constraint 3 ensures the completion time of job j at position k on machine i as after earliest start time of job j plus processing time of job j, if job j is assigned to position k on machine i. Constraint 4 ensures if job j is assigned to position k on machine i (notice the big M parameter becomes zero), then it must be completed at its due date or a tardiness is incurred. Constraint 5 ensures all jobs are assigned to a machine that can process that job and to a position on that machine. Constraint 6 initiates the position 0's completion time as 0. Constraint 7 ensures job j is assigned to at most one position and one machine. Constraint 8 ensures at most one job is assigned to position k on machine i.

These constraints also prevent more than one x_{jki} variable to become 1 when parameter a_{ji} is 0 for one of the x variables in constraint 5. Constraint 9 ensures smaller positions are filled first.

2.3.1.2. **Relative Positional Model**

2.3.1.2.1. **Parameters**

Same as above

2.3.1.2.2. **Sets**

Q_i = Jobs that can be processed on machine i ($j \mid a_{ji} > 0$)

$Q0_i = \{0\} \cup Q_i$ Jobs that can be processed on machine i, with artificial job 0

$QL_i = Q_i \cup \{n+1\}$ Jobs that can be processed on machine i, with artificial final job $n+1$

$Q0L_i = \{0\} \cup Q_i \cup \{n+1\}$ Jobs that can be processed on machine i, with both artificial jobs

R_j = Machines that job j requires ($i \mid a\_(j,i) > 0$)

2.3.1.2.3. **Variables**

$C_{\max}$ = Makespan - completion time of all jobs

$x_{i,j,k}$ = 1 if job j follows job k on machine i, 0 otherwise

($i \in I, j \in QL_i, k \in Q0_i$)

$y_{i,j}$ = 1 if job j is assigned to machine i, 0 otherwise ($i \in I, j \in QOL_i$)

t_{ji} = Completion time of job j on machine i

L_j = Tardiness of job j

2.3.1.2.4. **Model**

$minimize\ C_{\max} + \sum_j c_j * L_j$
$subject\ to$
$C_{max} \geq t_{j,i}\ \forall\ (j \in J, i \in I)$ (1)
$t_{j,i} \geq t_{k,i} + p_{j,i} - M * (1 - x_{i,j,k})\ \forall\ (i \in I, j \in Q_i, k \in Q_i: k \neq j)$ (2)
$t_{j,i} \geq (s_j + p_{j,i}) * (y_{i,j})\ \forall\ (j \in J, i \in R_j)$ (3)
$t_{j,i} \leq d_j + L_j + M * (1 - y_{i,j})\ \forall\ (j \in J, i \in R_j)$ (4)
$\sum_{j \in QL_i: j \neq k} x_{i,j,k} = y_{i,k}\ \forall\ (i \in I, k \in QO_i)$ (5)
$\sum_{k \in QO_i: k \neq j} x_{i,j,k} = y_{i,j}\ \forall\ (i \in I, j \in QL_i)$ (6)
$y_{i,0} = 1\ \forall\ (i \in I)$ (7)
$y_{i,n+1} = 1\ \forall\ (i \in I)$ (8)
$\sum_{i \in R_j} y_{i,j} = 1\ \forall\ (j \in J)$ (9)

The objective function is to minimize the makespan and tardiness. Constraint 1 makes sure that the makespan is after all jobs on all machines are complete. Constraint 2 allows if job j follows job k, completion time of job j is after completion time of job k plus processing time of job j on machine i. Constraint 3 ensures that the completion time of job j on machine i after earliest start time of job j plus processing time of job j, if job j is assigned to machine i. Constraint 4 ensures that completion time of job j on machine i must either be less than or equal to due date or tardiness occurs if job j is assigned to machine i. Constraint 5 assigns a successor for each job that is assigned to machine i, first job succeeds artificial job 0. Constraint 6 assigns a predecessor for each job that is assigned to machine i, last job precedes artificial final job. Constraint 7 ensures artificial

job 0 occurs on all machines. Constraint 8 ensures artificial final job occurs on all machines. Constraint 9 assigns all jobs to machines that can process them.

2.3.2. **Parallel Machine with Sequence-Dependent Setup Time Models**

2.3.2.1. **Direct Positional Model**

2.3.2.1.1. **Sets**

J = 1..n	Jobs
I = 1..m	Machines
K = 1..n	Positions on the machines
U = 1..o	Geometries

2.3.2.1.2. **Parameters**

n =	number of jobs
m =	number of machines
o =	number of geometries
d_j =	Due date of job j
$a_{j,i}$ =	1 if job j can be done on machine i, 0 otherwise
$p_{j,i}$ =	Processing time of job j on machine i
c_j =	Penalty for tardiness of job j
s_j =	Earliest start time of job j

$f_{u,v,i}$ = Setup time from geometry u to v on machine i

$g_{u,i}$ = Initial setup time of geometry u on machine i

$q_{j,u}$ = 1 if job j is geometry u, 0 otherwise

$M = Max_j(s_j) + \sum_{i,j} p_{j,i} + \sum_{u,v,i} f_{u,v,i} + \sum_{u,i} g_{u,i}$ A very large number

2.3.2.1.3. **Variables**

$C_{\max}$ = Makespan - completion time of all jobs

$t_{0..n,i}$ = Completion time of jobs at positions 0 through n on machine i

D_j = Tardiness of job j

$x_{j,k,i}$ = 1 if job j is assigned to position k on machine i, 0 otherwise

$y_{u,v,i,k}$ = 1 if there is a setup from geometry u to v on machine i at position k; 0 otherwise.

2.3.2.1.4. **Model**

$minimize\ C_{\max} + \sum_j c_j * D_j$

$subject\ to$

$Cmax \geq t_{k,i}\ \forall\ k \in K, i \in I$ (1)

$t_{k,i} \geq t_{k-1,i} + \sum_j p_{j,i} * x_{j,k,i} + \sum_{u,v} f_{u,v,i} * y_{u,v,i,k}\ \forall\ k \in K, i \in I$ (2)

$t_{k,i} \geq (s_j + p_{j,i}) * x_{j,k,i}\ \forall\ j \in J, k \in K, i \in I$ (3)

$t_{k,i} \leq d_j + M * (1 - x_{j,k,i}) + D_j\ \forall\ j \in J, k \in K, i \in I$ (4)

$\sum_{i,k} a_{j,i} * x_{j,k,i} = 1\ \forall\ j \in J$ (5)

$t_{0,i} = \sum_{u,j} q_{j,u} * g_{u,i} * x_{j,1,i}\ \forall\ i \in I$ (6)

$\sum_{i,k} x_{j,k,i} \leq 1\ \forall\ j \in J$ (7)

$\sum_j x_{j,k,i} \leq 1\ \forall\ i \in I, k \in K$ (8)

$\sum_j x_{j,k,i} \geq \sum_j x_{j,k+1,i}\ \forall\ i \in I, k \in 1..n-1$ (9)

$\sum_j q_{j,u} * x_{j,k-1,i} + \sum_l q_{l,v} * x_{l,k,i} \leq y_{u,v,i,k} + 1\ \forall\ (i \in I, k \in 2..n, u \in U, v \in U)$ (10)

Constraint 1 makes sure that the makespan is after all jobs on all machines are complete. Constraint 2 calculates completion time of the job j at position k on machine i as after completion time of previous position $k-1$ plus the processing time of job j on machine i plus the setup time from geometry u to geometry v on machine i, if there is a setup at that position. Constraint 3 ensures the completion time of job j at position k on machine i as after earliest start time of job j plus processing time of job j, if job j is assigned to position k on machine i. Constraint 4 ensures if job j is assigned to position k on machine i (notice the big M parameter becomes zero), then it must be completed at its due date or a tardiness is incurred. Constraint 5 ensures all jobs are assigned to a machine that can process that job and to a position on that machine. Constraint 6 initiates the position 0's completion time as the initial setup time of job j at position 1, job j being geometry u. Constraint 7 ensures job j is assigned to at most one position and one machine. Constraint 8 ensures at most one job is assigned to position k on machine i. Constraint 9 ensures smaller positions are filled first. Constraint 10 ensure that if job j, which is geometry u, and job l, which is geometry v, is assigned to positions $k-1$ and k, then there is a setup from geometry u to v.

2.3.2.2. **Relative Positional Model**

2.3.2.2.1. **Parameters**

Same as above

2.3.2.2.2. **Sets**

Q_i = Jobs that can be processed on machine i $(j \mid a_{ji} > 0)$

$Q0_i = \{0\} \cup Q_i$ Jobs that can be processed on machine i, with artificial job 0

$QL_i = Q_i \cup \{n+1\}$ Jobs that can be processed on machine i, with artificial final job $n+1$

$Q0L_i = \{0\} \cup Q_i \cup \{n+1\}$ Jobs that can be processed on machine i, with both artificial jobs

$R_j =$ Machines that job j requires $(i \mid a\_(j,i) > 0)$

2.3.2.2.3. **Variables**

$C_{\max}$ = Makespan - completion time of all jobs

$x_{i,j,k}$ = 1 if job j follows job k on machine i, 0 otherwise $(i \in I, j \in QL_i, k \in Q0_i)$

$y_{i,j}$ = 1 if job j is assigned to machine i, 0 otherwise $(i \in I, j \in Q0L_i)$

$t_{i,j}$ = Completion time of job j on machine i, j=0,1…..,n.

L_j = Tardiness of job j

2.3.2.2.4. **Model**

$minimize\ C_{\max} + \sum_j c_j * L_j$
$subject\ to$

$C_{max} \geq t_{i,j}\ \forall (j \in J, i \in I)$ (1)

$t_{i,j} \geq t_{i,k} + p_{j,i} + \sum_{u,v} q_{j,u} * q_{k,v} * f_{u,v,i} - M * (1 - x_{i,j,k})\ \forall (i \in I, j \in Q_i, k \in Q_i : k \neq j)$ (2)

$t_{i,j} \geq (s_j + p_{j,i}) * (y_{i,j})\ \forall (j \in J, i \in R_j)$ (3)

$t_{i,j} \leq d_j + L_j + M * (1 - y_{i,j})\ \forall (j \in J, i \in R_j)$ (4)

$\sum_{j \in QL_i : j \neq k} x_{i,j,k} = y_{i,k}\ \forall (i \in I, k \in Q0_i)$ (5)

$\sum_{k \in Q0_i : k \neq j} x_{i,j,k} = y_{i,j}\ \forall (i \in I, j \in QL_i)$ (6)

$t_{i,0} = \sum_{u \in U, j \in Q_i} q_{j,u} * g_{u,i} * x_{i,j,0}\ \forall i \in I$ (7)

$y_{i,0} = 1\ \forall (i \in I)$ (8)

$y_{i,n+1} = 1\ \forall (i \in I)$ (9)

$\sum_{i \in R_j} y_{i,j} = 1\ \forall (j \in J)$ (10)

$$t_{i,j} \geq t_{i,0} + p_{j,i} - M * (1 - x_{i,j,0}) \forall (i \in I, j \in Q_i) \quad (11)$$

The objective function is to minimize the makespan and tardiness. Constraint 1 makes sure that the makespan occurs after all jobs on all machines are complete. Constraint 2 allows if job j follows job k, completion time of job j is after completion time of job k plus processing time of job j on machine i and setup time from geometry u to geometry v on machine i (given job j is geometry u and job k is geometry v). Constraint 3 ensures that the completion time of job j on machine i after earliest start time of job j plus processing time of job j, if job j is assigned to machine i. Constraint 4 ensures that completion time of job j on machine i must either be less than or equal to due date or tardiness occurs if job j is assigned to machine i. Constraint 5 assigns a successor for each job that is assigned to machine i, first job succeeds artificial job 0. Constraint 6 assigns a predecessor for each job that is assigned to machine i, last job precedes artificial final job. Constraint 7 initiates the job 0's completion time as the initial setup time of job j at position 1, job j being geometry u. Constraint 8 ensures artificial job 0 occurs on all machines. Constraint 9 ensures artificial final job occurs on all machines. Constraint 10 assigns all jobs to machines that can process them. Constraint 11 allows if job j follows job 0, completion time of job j is after initial setup time (completion time of job 0) plus processing time of job j on machine i.

2.3.3. **Flexible Flow Shop Models (Tubing without Setups)**

2.3.3.1. **Direct Positional Model**

2.3.3.1.1. **Sets**

J = 1..n Jobs

I = 1..m	Machines
$I_t \in I$	Machines that require tubing
$I_{nt} \in I$	Machines that do not require tubing
K = 1..n	Positions on the machines
U = 1..o	Geometries
H = 1..e	Tubing Stations

2.3.3.1.2. **Parameters**

n =	number of jobs
m =	number of machines
o =	number of geometries
e =	number of tubing stations
d_j =	Due date of job j
$a_{u,i}$ =	1 if geometry u can be done on machine i, 0 otherwise
$p_{u,i}$ =	Unit processing time of geometry u on machine i
c_j =	Penalty for tardiness of job j
s_j =	Earliest start time of job j
$q_{j,u}$ =	1 if job j is geometry u, 0 otherwise

$r_{u,h}$ = Unit tubing time of geometry u on tubing station h

$w_{u,h}$ = 1 if geometry u can be processed on station h, 0 otherwise

α_j = Output of job j

$M = Max_j(s_j) + \sum_{i \in I, j \in J, u \in U} \alpha_j * q_{j,u} * p_{u,i}$

$+ \sum_{h \in H, j \in J, u \in U} \alpha_j * q_{j,u} * r_{u,h}$ A very large number

2.3.3.1.3. **Variables**

$C_{\max}$ = Makespan - completion time of all jobs

$t_{0..n,i}$ = Completion time of jobs at positions 0 through n on machine i

L_j = Tardiness of job j

$x_{j,k,i}$ = 1 if job j is assigned to position k on machine i, 0 otherwise

$\tau_{0..n,h}$ = Completion time of jobs at positions 0 through n on tubing station h

$z_{j,k,h}$ = 1 if job j is assigned to position k on tubing station h, 0 otherwise

$\rho_{j,i}$ = Possible start time of job j on machine i, after tubing

2.3.3.1.4. **Model**

$minimize\ C_{\max} + \sum_j c_j * L_j$

$subject\ to$

$Cmax \geq t_{k,i}\ \forall\ k \in K, i \in I$ (1)

$t_{k,i} \geq t_{k-1,i} + \sum_{j,u} q_{j,u} * p_{u,i} * \alpha_j * x_{j,k,i}\ \forall\ k \in K, i \in I$ (2)

$t_{k,i} \geq \left(\rho_{j.i} + \sum_u q_{j,u} * p_{u,i} * \alpha_j\right) - M * (1 - x_{j,k,i})\ \forall\ j \in J, k \in K, i \in I$ (3)

$\rho_{j,i_{nt}} = s_j\ \forall\ j \in J, i_{nt} \in I_{nt}$ (4)

$t_{k,i} \leq d_j + M * (1 - x_{j,k,i}) + L_j\ \forall\ j \in J, k \in K, i \in I$ (5)

$\sum_{i,k} a_{u,i} * q_{j,u} * x_{j,k,i} = 1\ \forall\ j \in J$ (6)

$t_{0,i} = 0\ \forall\ i \in I$ (7)

$\sum_{i,k} x_{j,k,i} \leq 1\ \forall\ j \in J$ (8)

$\sum_j x_{j,k,i} \leq 1\ \forall\ i \in I, k \in K$ (9)

$\sum_j x_{j,k,i} \geq \sum_j x_{j,k+1,i}\ \forall\ i \in I, k \in 1..n-1$ (10)

$\tau_{k,h} \geq \tau_{k-1,h} + \sum_{j,u} q_{j,u} * r_{u,h} * \alpha_j * z_{j,k,h}\ \ \forall\ (k \in K, h \in H)$ (11)

$\tau_{k,h} \geq \left(s_j + \sum_u q_{j,u} * r_{u,h} * \alpha_j\right) * z_{j,k,h}\ \forall\ (j \in J, k \in K, h \in H)$ (12)

$\sum_{h \in H, k \in K, u \in U} w_{u,h} * q_{j,u} * z_{j,k,h} = 1 - \sum_{k \in K} x_{j,k,i_{nt}}\ \forall\ (j \in J,\ i_{nt} \in I_{nt})$ (13)

$\tau_{0,h} = 0\ \forall\ (h \in H)$ (14)

$\sum_{h,k} z_{j,k,h} \leq 1 - \sum_k x_{j,k,i_{nt}}\ \forall\ (j \in J, i_{nt} \in I_{nt})$ (15)

$\sum_j z_{j,k,h} \leq 1\ \forall\ (h \in H, k \in K)$ (16)

$\sum_j z_{j,k,h} \geq \sum_j z_{j,k+1,h}\ \forall\ (h \in H, k \in 1..n-1)$ (17)

$\rho_{j,i_t} + M * \left(1 - z_{j,k,h}\right) \geq \tau_{k,h}\ \forall\ (j \in J, k \in K, h \in H, i_t \in I_t)$ (18)

Constraint 1 makes sure that the makespan is after all jobs on all machines are complete. Constraint 2 calculates completion time of the job j at position k on machine i as after completion time of previous position $k-1$ plus the processing time of job j on machine i. Constraint 3 ensures the completion time of job j at position k on machine i as after possible start time of job j after tubing station, plus processing time of job j, if job j is assigned to position k on machine i. For the machines that there is no tubing, the earliest start time and possible start time after tubing are equal, using constraint 4. Constraint 5 ensures if job j is assigned to position k on machine i (notice the big M parameter becomes zero), then it must be completed at its due date or a tardiness is incurred. Constraint 6 ensures all jobs are assigned to a machine that can process that job and to a position on that machine. Constraint 7 initiates the position 0's completion time as 0. Constraint 8 ensures job j is assigned to at most one position and one machine. Constraint

9 ensures at most one job is assigned to position k on machine i. Constraint 10 ensures smaller positions are filled first. Constraint 11 calculates completion time of the job j at position k on tubing station h as after completion time of previous position $k-1$ plus the tubing time of job j on tubing station h. Constraint 12 ensures the completion time of job j at position k on tubing station h as after earliest start time of job j plus tubing time of job j, if job j is assigned to position k on tubing station h. Constraint 13 ensures all jobs are assigned to a tubing station that can process that job and to a position on that station, if the job is not assigned to a machine that doesn't require tubing. Start time for tubing at position 0 is zero using constrain 14. Constraint 15 ensures job j is assigned to at most one position and one tubing station, if it is not assigned to a machine that doesn't require tubing. Constraint 16 ensures at most one job is assigned to position k on tubing station h. Constraint 17 ensures smaller positions are filled first on tubing stations also. Constraint 18 links tubing completion time of a job to possible start time on machines that require tubing.

2.3.3.2. **Relative Positional Model**

2.3.3.2.1. **Parameters**

Same as above

2.3.3.2.2. **Sets**

Q_i = Jobs that can be processed on machine i $(j \mid \sum_u q_{ju} * a_{ui} > 0)$

$Q0_i = \{0\} \cup Q_i$ Jobs that can be processed on machine i, with artificial job 0

$QL_i = Q0_i \cup \{n+1\}$ Jobs that can be processed on machine i, with artificial final job $n+1$

$Q0L_i = \{0\} \cup Q_i \cup \{n+1\}$ Jobs that can be processed on machine i, with both artificial jobs

$R_j =$ Machines that job j requires $(i \mid \sum_u q_{ju} * a_{ui} > 0)$

$W_h =$ Jobs that can be processed on tubing station h $(j \mid \sum_u q_{ju} * w_{uh} > 0)$

$W0_h = \{0\} \cup W_h$ Jobs that can be processed on station h, with artificial job 0

$WL_h = W_h \cup \{n+1\}$ Jobs that can be processed on station h, with artificial final job $n+1$

$W0L_h = \{0\} \cup W_h \cup \{n+1\}$ Jobs that can be processed on station h, with both artificial jobs

$V_j =$ Stations that job j requires $(h \mid \sum_u q_{ju} * w_{uh} > 0)$

2.3.3.2.3. **Variables**

$C_{\max} =$ Makespan - completion time of all jobs

$x_{i,j,k} =$ 1 if job j follows job k on machine i, 0 otherwise $(i \in I, j \in QL_i, k \in Q0_i)$

$y_{i,j} =$ 1 if job j is assigned to machine i, 0 otherwise $(i \in I, j \in Q0L_i)$

$t_{i,0..n} =$ Completion time of job 0..n on machine i

$L_j =$ Tardiness of job j

$\tau_{j,h}$ = Completion time of job j on tubing station h

$z_{j,k,h}$ = 1 if job j is follows job k on tubing station h, 0 otherwise ($h \in H, j \in WL_h, k \in W0_h$)

$\rho_{j,i}$ = Possible start time of job j on machine i, after tubing

$\zeta_{j,h}$ = 1 if job j is assigned to station h, 0 otherwise ($h \in H, j \in W0L_h$)

2.3.3.2.4. **Model**

$minimize\ C_{\max} + \sum_j c_j * L_j$
$subject\ to$

$C_{max} \geq t_{i,j}\ \forall (j \in J, i \in I)$ (1)
$t_{i,j} \geq t_{i,k} + \sum_u q_{j,u} * p_{u,i} * \alpha_j - M * (1 - x_{i,j,k})\ \forall (i \in I, j \in Q_i, k \in Q_i : k \neq j)$ (2)
$t_{i,j} \geq \rho_{j,i} + \sum_u q_{j,u} * p_{u,i} * \alpha_j - M * (1 - y_{i,j})\ \forall (i \in I, j \in Q_i)$ (3)
$\rho_{j,i_{nt}} = s_j\ \forall (j \in J, i_{nt} \in I_{nt})$ (4)
$t_{i,j} \leq d_j + M * (1 - y_{i,j}) + L_j\ \forall (i \in I, j \in Q_i)$ (5)
$\sum_{j \in QL_i : j \neq k} x_{i,j,k} = y_{i,k}\ \forall (i \in I, k \in Q0_i)$ (6)
$\sum_{k \in Q0_i : k \neq j} x_{i,j,k} = y_{i,j}\ \forall (i \in I, j \in QL_i)$ (7)
$t_{i,0} = 0\ \forall (i \in I)$ (8)
$y_{i,0} = 1\ \forall (i \in I)$ (9)
$y_{i,n+1} = 1\ \forall (i \in I)$ (10)
$\sum_{i \in R_j} y_{i,j} = 1\ \forall (j \in J)$ (11)
$t_{i,j} \geq t_{i,0} + \sum_u q_{j,u} * p_{u,i} * \alpha_j - M * (1 - x_{i,j,0})\ \forall (i \in I, j \in Q_i)$ (12)
$\tau_{j,h} \geq \tau_{h,k} + \sum_u q_{j,u} * r_{u,h} * \alpha_j - M * (1 - z_{h,j,k})\ \forall (h \in H, j \in W_h, k \in W_h : k \neq j)$ (13)
$\tau_{j,h} \geq (s_j + \sum_u q_{j,u} * r_{u,h} * \alpha_j) * \zeta_{j,h}\ \forall (h \in H, j \in W_h)$ (14)
$\sum_{j \in WL_h : j \neq k} z_{h,j,k} = \zeta_{k,h}\ \forall (h \in H, k \in W0_h)$ (15)
$\sum_{k \in W0_h : k \neq j} z_{h,j,k} = \zeta_{j,h}\ \forall (h \in H, j \in WL_h)$ (16)
$\zeta_{0,h} = 1\ \forall (h \in H)$ (17)
$\zeta_{n+1,h} = 1\ \forall (h \in H)$ (18)
$\sum_{h \in V_j} \zeta_{j,h} = 1 - y_{i_{nt},j}\ \forall (j \in J, i \in I_{nt})$ (19)
$\rho_{j,i} + M * (1 - \zeta_{j,h}) \geq \tau_{j,h}\ \forall (j \in J, h \in V_j, i \in I_t)$ (20)

Constraint 1 makes sure that the makespan is after all jobs on all machines are complete. Constraint 2 allows that if job j follows job k, completion time of job j is after completion time of job k plus processing time of job j on machine i.Constraint 3 ensures the completion time of job j on machine i is after possible start time of job j after tubing

plus processing time of job j, if job j is assigned to machine i. Constraint 4 ensures that for machines that do not require tubing, the earliest start time and possible start time after tubing are equal. Constraint 5 ensures if job j is assigned to machine i (notice the big M parameter becomes zero), then it must be completed at its due date or a tardiness is incurred. Constraint 6 assigns a successor for each job that is assigned to machine i, first job succeeds artificial job 0. Constraint 7 assigns a predecessor for each job that is assigned to machine i, last job precedes artificial final job. Constraint 8 initiates the job 0's completion time as 0. Constraint 9 allows artificial job 0 to occur on all machines. Constraint 10 allows artificial final job to occur on all machines. Constraint 11 assigns jobs to machines that can process them. Constraint 12 allows if job j follows job 0, completion time of job j is after completion time of job 0 plus processing time of job j on machine i. Constraint 13 ensures that the completion time of job j on tubing station h is after earliest start time of job j plus tubing time of job j, if job j is assigned to tubing station h. Constraint 14 calculates completion time of the job j on tubing station h as after completion time of previous job k plus the tubing time of job j on tubing station h. Constraint 15 assigns a successor for each job that is assigned to tubing station h, first job succeeds artificial job 0. Constraint 16 assigns a predecessor for each job that is assigned to tubing station h, last job precedes artificial final job. Constraint 17 allows artificial job 0 to occur on all tubing stations. Constraint 18 allows artificial final job to occur on all tubing stations. Constraint 19 assigns jobs to tubing stations that can process them, if the job is not assigned to a machine that doesn’t require tubing. Constraint 20 links tubing completion time of a job to possible start time on machines that require tubing.

2.3.4. **Flexible Flow Shop with Sequence Dependent Setups Models (Tubing with Setups)**

2.3.4.1. **Direct Positional Model**

2.3.4.1.1. **Sets**

J = 1..n	Jobs
I = 1..m	Machines
$I_{nt} \in I$	Machines that do not require tubing beforehand
$I_t \in I$	Machines that require tubing beforehand
K = 1..n	Positions on the machines
U = 1..o	Geometries
H = 1..e	Tubing Stations

2.3.4.1.2. **Parameters**

n =	number of jobs
m =	number of machines
o =	number of geometries
e =	number of tubing stations
d_j =	Due date of job j
$a_{u,i}$ =	1 if geometry u can be done on machine i, 0 otherwise

$p_{u,i}$ = Unit processing time of geometry u on machine i

c_j = Penalty for tardiness of job j

s_j = Earliest start time of job j

$f_{u,v,i}$ = Setup time from geometry u to v on machine i

$g_{u,i}$ = Initial setup time of geometry u on machine i

$q_{j,u}$ = 1 if job j is geometry u, 0 otherwise

$r_{u,h}$ = Unit tubing time of geometry u on tubing station h

$w_{u,h}$ = 1 if geometry u can be processed on station h, 0 otherwise

α_j = Size of job j

$M = Max_j(s_j) + \sum_{i\in I, j\in J, u\in U} \alpha_j * q_{j,u} * p_{u,i} + \sum_{h\in H, j\in J, u\in U} \alpha_j * q_{j,u} * r_{u,h} + \sum_{u\in U, v\in U, i\in I} f_{u,v,i} + \sum_{u\in U, i\in I} g_{u,i}$ A very large number

2.3.4.1.3. **Variables**

$C_{\max}$ = Makespan - completion time of all jobs

$t_{K,i}$ = Completion time of jobs at position k, k= 0…n, on machine i

L_j = Tardiness of job j

$x_{j,k,i}$ = 1 if job j is assigned to position k on machine i, 0 otherwise

$y_{u,v,i,k}$ = 1 if there is a setup from geometry u to v on machine i at position k

$\tau_{k,h}$ = Completion time of job at position k, k= 0…n, on tubing station h

$z_{j,k,h}$ = 1 if job j is assigned to position k on tubing station h, 0 otherwise

$\rho_{j,i}$ = Possible start time of job j on machine i, after tubing

2.3.4.1.4. **Model**

$$minimize\ C_{\max} + \sum_j c_j * L_j$$

$subject\ to$

$$Cmax \geq t_{k,i}\ \forall\ k \in K, i \in I \quad (1)$$
$$t_{k,i} \geq t_{k-1,i} + \sum_{j,u} q_{j,u} * p_{u,i} * \alpha_j * x_{j,k,i} + \sum_{u,v} f_{u,v,i} * y_{u,v,i,k}\ \forall\ k \in K, i \in I \quad (2)$$
$$t_{k,i} \geq \left(\rho_{j.i} + \sum_u q_{j,u} * p_{u,i} * \alpha_j\right) - M * (1 - x_{j,k,i})\ \forall\ j \in J, k \in K, i \in I \quad (3)$$
$$\rho_{j,i_{nt}} = s_j\ \forall\ j \in J, i \in I_{nt} \quad (4)$$
$$t_{k,i} \leq d_j + M * (1 - x_{j,k,i}) + L_j\ \forall\ j \in J, k \in K, i \in I \quad (5)$$
$$\sum_{i,k} a_{u,i} * q_{j,u} * x_{j,k,i} = 1\ \forall\ j \in J \quad (6)$$
$$t_{0,i} = \sum_{u,j} q_{j,u} * g_{u,i} * x_{j,1,i}\ \forall\ i \in I \quad (7)$$
$$\sum_{i,k} x_{j,k,i} \leq 1\ \forall\ j \in J \quad (8)$$
$$\sum_j x_{j,k,i} \leq 1\ \forall\ i \in I, k \in K \quad (9)$$
$$\sum_j x_{j,k,i} \geq \sum_j x_{j,k+1,i}\ \forall\ i \in I, k \in 1..n-1 \quad (10)$$
$$\sum_j q_{j,u} * x_{j,k-1,i} + \sum_l q_{l,v} * x_{l,k,i} \leq y_{u,v,i,k} + 1\ \forall\ (i \in I, k \in 2..n, u \in U, v \in U) \quad (11)$$
$$\tau_{k,h} \geq \tau_{k-1,h} + \sum_{j,u} q_{j,u} * r_{u,h} * \alpha_j * z_{j,k,h}\ \forall\ (k \in K, h \in H) \quad (12)$$
$$\tau_{k,h} \geq \left(s_j + \sum_u q_{j,u} * r_{u,h} * \alpha_j\right) * z_{j,k,h}\ \forall\ (j \in J, k \in K, h \in H) \quad (13)$$
$$\sum_{h \in H, k \in K, u \in U} w_{u,h} * q_{j,u} * z_{j,k,h} = 1 - \sum_{k \in K} x_{j,k,i_{nt}}\ \forall\ (j \in J, i \in I_{nt}) \quad (14)$$
$$\tau_{0,h} = 0\ \forall\ (h \in H) \quad (15)$$
$$\sum_{h,k} z_{j,k,h} \leq 1 - \sum_k x_{j,k,i_{nt}}\ \forall\ (j \in J, i \in I_{nt}) \quad (16)$$
$$\sum_j z_{j,k,h} \leq 1\ \forall\ (h \in H, k \in K) \quad (17)$$
$$\sum_j z_{j,k,h} \geq \sum_j z_{j,k+1,h}\ \forall\ (h \in H, k \in 1..n-1) \quad (18)$$
$$\rho_{j,i} + M * \left(1 - z_{j,k,h}\right) \geq \tau_{k,h}\ \forall\ (j \in J, k \in K, h \in H, i \in I_t) \quad (19)$$

Constraint 1 makes sure that the makespan is after all jobs on all machines are complete. Constraint 2 calculates completion time of the job j at position k on machine i as after completion time of previous position $k-1$ plus the processing time of job j on machine i plus the setup time from geometry u to geometry v on machine i, if there is a setup at that position. Constraint 3 ensures the completion time of job j at position k on machine i as after possible start time of job j after tubing station, plus processing time of job j, if job j is assigned to position k on machine i. For the set of jobs that there is no tubing, the

earliest start time and possible start time after tubing are equal, using constraint 4. Constraint 5 ensures if job j is assigned to position k on machine i (notice the big M parameter becomes zero), then it must be completed at its due date or a tardiness is incurred. Constraint 6 ensures all jobs are assigned to a machine that can process that job and to a position on that machine. Constraint 7 initiates the position 0's completion time as the initial setup time of job j at position 1, job j being geometry u. Constraint 8 ensures job j is assigned to at most one position and one machine. Constraint 9 ensures at most one job is assigned to position k on machine i. Constraint 10 ensures smaller positions are filled first. Constraint 11 ensure that if job j, which is geometry u, and job l, which is geometry v, is assigned to positions $k-1$ and k, then there is a setup from geometry u to v. Constraint 12 calculates completion time of the job j at position k on tubing station h as after completion time of previous position $k-1$ plus the tubing time of job j on tubing station h. Constraint 13 ensures the completion time of job j at position k on tubing station h as after earliest start time of job j plus tubing time of job j, if job j is assigned to position k on tubing station h. Constraint 14 ensures all jobs are assigned to a tubing station that can process that job and to a position on that station, if the job is not assigned to WAT. Start time for tubing at position 0 is zero using constrain 15. Constraint 16 ensures job j is assigned to at most one position and one tubing station, if it is not assigned to a machine that requires tubing. Constraint 17 ensures at most one job is assigned to position k on tubing station h. Constraint 18 ensures smaller positions are filled first on tubing stations also. Constraint 19 links tubing completion time of a job to possible start time on machines that require tubing.

2.3.4.2. **Relative Positional Model**

2.3.4.2.1. **Parameters**

Same as above

2.3.4.2.2. **Sets**

Q_i = Jobs that can be processed on machine i ($j \mid \sum_u q_{ju} * a_{ui} > 0$)

$Q0_i = \{0\} \cup Q_i$ Jobs that can be processed on machine i, with artificial job 0

$QL_i = Q0_i \cup \{n+1\}$ Jobs that can be processed on machine i, with artificial final job $n+1$

$Q0L_i = \{0\} \cup Q_i \cup \{n+1\}$ Jobs that can be processed on machine i, with both artificial jobs

R_j = Machines that job j requires ($i \mid \sum_u q_{ju} * a_{ui} > 0$)

W_h = Jobs that can be processed on tubing station h ($j \mid \sum_u q_{ju} * w_{uh} > 0$)

$W0_h = \{0\} \cup W_h$ Jobs that can be processed on station h, with artificial job 0

$WL_h = W_h \cup \{n+1\}$ Jobs that can be processed on station h, with artificial final job $n+1$

$W0L_h = \{0\} \cup W_h \cup \{n+1\}$ Jobs that can be processed on station h, with both artificial jobs

V_j = Stations that job j requires ($h \mid \sum_u q_{ju} * w_{uh} > 0$)

2.3.4.2.3. **Variables**

$C_{\max}$ = Makespan - completion time of all jobs

$x_{i,j,k}$ = 1 if job j follows job k on machine i, 0 otherwise ($i \in I, j \in QL_i, k \in Q0_i$)

$y_{i,j}$ = 1 if job j is assigned to machine i, 0 otherwise ($i \in I, j \in Q0L_i$)

$t_{i,0..n}$ = Completion time of job 0..n on machine i

L_j = Tardiness of job j

$\tau_{j,h}$ = Completion time of job j on tubing station h

$z_{j,k,h}$ = 1 if job j is follows job k on tubing station h, 0 otherwise($h \in H, j \in WL_h, k \in W0_h$)

$\rho_{j,i}$ = Possible start time of job j on machine i, after tubing

$\zeta_{j,h}$ = 1 if job j is assigned to station h, 0 otherwise ($h \in H, j \in W0L_h$)

2.3.4.2.4. **Model**

$minimize\ C_{\max} + \sum_j c_j * L_j$
$subject\ to$

$C_{max} \geq t_{i,j}\ \forall (j \in J, i \in I)$ (1)

$t_{i,j} \geq t_{i,k} + \sum_u q_{j,u} * p_{u,i} * \alpha_j + \sum_{u,v} q_{j,u} * q_{k,v} * f_{u,v,i} - M * (1 - x_{i,j,k})$
$\forall (i \in I, j \in Q_i, k \in Q_i : k \neq j)$ (2)

$t_{i,j} \geq \rho_{j,i} + \sum_u q_{j,u} * p_{u,i} * \alpha_j - M * (1 - y_{i,j})\ \forall (i \in I, j \in Q_i)$ (3)

$\rho_{j,i_{nt}} = s_j\ \forall (j \in J, i \in I_{nt})$ (4)

$t_{i,j} \leq d_j + M * (1 - y_{i,j}) + L_j\ \forall (i \in I, j \in Q_i)$ (5)

$\sum_{j \in QL_i : j \neq k} x_{i,j,k} = y_{i,k}\ \forall (i \in I, k \in Q0_i)$ (6)

$\sum_{k \in Q0_i : k \neq j} x_{i,j,k} = y_{i,j}\ \forall (i \in I, j \in QL_i)$ (7)

$t_{i,0} = \sum_{u \in U, j \in Q_i} q_{j,u} * g_{u,i} * x_{i,j,0}\ \forall (i \in I)$ (8)

$y_{i,0} = 1\ \forall (i \in I)$ (9)

$y_{i,n+1} = 1\ \forall (i \in I)$ (10)

$\sum_{i \in R_j} y_{i,j} = 1\ \forall (j \in J)$ (11)

$t_{i,j} \geq t_{i,0} + \sum_u q_{j,u} * p_{u,i} * \alpha_j - M * (1 - x_{i,j,0})\ \forall (i \in I, j \in Q_i)$ (12)

$\tau_{j,h} \geq \tau_{h,k} + \sum_u q_{j,u} * r_{u,h} * \alpha_j - M * (1 - z_{h,j,k})\ \forall (h \in H, j \in W_h, k \in W_h : k \neq j)$ (13)

$\tau_{j,h} \geq (s_j + \sum_u q_{j,u} * r_{u,h} * \alpha_j) * \zeta_{j,h}\ \forall (h \in H, j \in W_h)$ (14)

$\sum_{j \in WL_h : j \neq k} z_{h,j,k} = \zeta_{k,h}\ \forall (h \in H, k \in W0_h)$ (15)

$\sum_{k\in W0_h:k\neq j} z_{h,j,k} = \zeta_{j,h} \forall (h \in H, j \in WL_h)$ (16)

$\zeta_{0,h} = 1 \forall (h \in H)$ (17)

$\zeta_{n+1,h} = 1 \forall (h \in H)$ (18)

$\sum_{h\in V_j} \zeta_{j,h} = 1 - y_{i_{nt},j} \forall (j \in J, i_{nt} \in I_{nt})$ (19)

$\rho_{j,i_t} + M * (1 - \zeta_{j,h}) \geq \tau_{j,h} \forall (j \in J, h \in V_j, i_t \in I_t)$ (20)

Constraint 1 makes sure that the makespan is after all jobs on all machines are complete. Constraint 2 allows that if job j follows job k, completion time of job j is after completion time of job k plus processing time of job j on machine i and setup time from geometry u to geometry v on machine i (given job j is geometry u and job k is geometry v).Constraint 3 ensures the completion time of job j on machine i is after possible start time of job j after tubing plus processing time of job j, if job j is assigned to machine i. Constraint 4 ensures that for the machine that don't require tubing, the earliest start time and possible start time after tubing are equal because there is no tubing. Constraint 5 ensures if job j is assigned to machine i (notice the big M parameter becomes zero), then it must be completed at its due date or a tardiness is incurred. Constraint 6 assigns a successor for each job that is assigned to machine i, first job succeeds artificial job 0. Constraint 7 assigns a predecessor for each job that is assigned to machine i, last job precedes artificial final job. Constraint 8 initiates the job 0's completion time as the initial setup time of job j following job 0, job j being geometry u. Constraint 9 allows artificial job 0 to occur on all machines. Constraint 10 allows artificial final job to occur on all machines. Constraint 11 assigns jobs to machines that can process them. Constraint 12 allows if job j follows job 0, completion time of job j is after initial setup time (completion time of job 0) plus processing time of job j on machine i. Constraint 13 ensures that the completion time of job j on tubing station h is after earliest start time of job j plus tubing time of job j, if job j is assigned to tubing station h. Constraint 14

calculates completion time of the job j on tubing station h as after completion time of previous job k plus the tubing time of job j on tubing station h. Constraint 15 assigns a successor for each job that is assigned to tubing station h, first job succeeds artificial job 0. Constraint 16 assigns a predecessor for each job that is assigned to tubing station h, last job precedes artificial final job. Constraint 17 allows artificial job 0 to occur on all tubing stations. Constraint 18 allows artificial final job to occur on all tubing stations. Constraint 19 assigns jobs to tubing stations that can process them, if the job is not assigned to machines that don't require tubing. Constraint 20 links tubing completion time of a job to possible start time on machines that require tubing.

2.4. **Computational Study**

2.4.1. **Experimental Setup**

In our computational study, we tested the aforementioned 4 problems, which we also list below:

- Parallel Machines (3 Machines)
- Parallel Machines with Setups (3 Machines)
- Flexible Flow Shop (2 Station 1 and 3 Station 2)
- Flexible Flow Shop with Setups (2 Station 1 and 3 Station 2)

We generated 5 random instances for each of our 4 problems, with 5 different instance sizes, thus a total of 100 instances:

- 10 jobs
- 15 jobs

- 21 jobs
- 31 jobs
- 41 jobs

An early data set we have received from Artaic contained 21 jobs, so we generated our medium-sized instances to match that data set. We added 10 and 20 jobs to those instances to create larger ones, and we used more typical, rounded numbers (10 and 15) for our instances smaller than the medium size. We solved these instances using IBM ILOG CPLEX Studio using CPLEX 12.7 on a computer with Windows 10 Home and 64 bit with Intel(R) Core™ i7-6700 CPU @ 3.40 GHz, and 32GB RAM. We limited the runtime of instances to 1 hour.

2.4.2. **Results**

We present the results for our computational experiments Table 2.4.1 below, comparing objective function values and optimality gaps for each instance under direct positional models and relative positional models:

PMP	Direct Positional Model		Relative Positional Model	
Size	Objective	Gap	Objective	Gap
10	756.47	0.00%	756.47	0.00%
15	985.44	0.01%	986.05	11.83%
21	1327.31	2.82%	1328.91	71.61%
31	1636.25	10.14%	1653.22	78.80%
41	2392.34	14.27%	2410.38	84.45%
PMP with Setups	Direct Positional Model		Relative Positional Model	
Size	Objective	Gap	Objective	Gap
10	758.06	0.00%	758.06	0.00%
15	989.13	0.01%	989.56	12.94%
21	1330.81	3.48%	1334.17	71.77%
31	1649.91	11.20%	1655.62	78.71%
41	2419.10	15.43%	2423.60	84.54%

FFS	Direct Positional Model		Relative Positional Model	
Size	Objective	Gap	Objective	Gap
10	839.00	0.00%	839.00	0.00%
15	1091.05	5.03%	1100.67	21.43%
21	1380.10	7.73%	1413.41	62.56%
31	1765.13	16.71%	1863.92	69.78%
41	2555.02	20.02%	2734.14	78.16%
FFS with Setups	Direct Positional Model		Relative Positional Model	
Size	Objective	Gap	Objective	Gap
10	839.45	0.00%	839.45	0.00%
15	1093.80	7.95%	1095.00	20.51%
21	1387.39	8.04%	1395.67	61.72%
31	1786.99	18.25%	1873.35	69.94%
41	2564.49	20.54%	2699.56	77.91%

Table 2.4.1 Objective Function Value and Optimality Gap for Direct Positional and Relative Positional Models for PMP and FFS with and without Setups

We compared the means of objective function values and optimality gaps for direct positional model and relative positional model using a t-test and found out that the difference between two means are statistically significant. The results of our t-tests are given below in Table 2.4.2 and Table 2.4.3:

	Variable 1	Variable 2
Mean	0.080821496	0.478345112
Variance	0.006882863	0.114977324
Observations	100	100
Pearson Correlation	0.666174073	
Hypothesized Mean Difference	0	
df	99	
t Stat	-13.68499511	
P(T<=t) one-tail	7.32414E-25	
t Critical one-tail	1.660391157	
P(T<=t) two-tail	1.46483E-24	
t Critical two-tail	1.9842169	

Table 2.4.2 T-Test for Optimality Gaps

	Variable 1	Variable 2
Mean	1477.363438	1507.510903

Variance	398392.6521	443447.4348
Observations	100	100
Pearson Correlation	0.996508467	
Hypothesized Mean Difference	0	
df	99	
t Stat	-4.684527996	
P(T<=t) one-tail	4.47577E-06	
t Critical one-tail	1.660391157	
P(T<=t) two-tail	8.95155E-06	
t Critical two-tail	1.9842169	

Table 2.4.3 T-Test for Objective Function Values

2.5. **Conclusions**

2.5.1. **Findings**

Based on the results of our computational experiments and statistical tests, we conclude that the direct positional formulation dominates relative positional formulation in Parallel Machine and Flexible Flow Shop Scheduling problems. This modeling approach allows for the exact solution of relatively complex machine scheduling problems of moderate size that include:

- Parallel Machines
- Flexible Flow Shops
- Sequence-dependent setups
- Start dates
- Due dates and tardiness
- Real-life inspired problems

The motivation behind our study was Artaic, a manufacturer of unique tile products that use automation. We were able to present our results and receive feedback on usability of

our models. The most practical model that can be solved the fastest, the parallel machine model without setups, is implemented and being used integrated in their cloud computing environment to respond to customer demand and schedule their orders in a timely fashion. The implementation will be explained further in the next section.

2.5.2. **Implementation**

We delivered two implementations of our models to Artaic: the earlier version was an Excel tool developed for optimization of schedules of the jobs that Artaic is committed to. The later version was a Node.js application that can be used by Artaic to solve the problem in their cloud computing environment. Both versions used the most practical model that can be solved the fastest, the parallel machine model without setups. The Excel tool can also visualize the optimal schedule in a Gantt chart and can simulate different schedules given by the decision maker. The Node.js application can list all the jobs, the machines that they are assigned to, and their start and completion times, which can then be put in a Google Gantt chart for visualization purposes. We will explain the use of each implementation briefly below:

2.5.2.1. **Excel tool**

The Excel tool consists of 4 tabs: TimeData tab consists of the unit manufacturing time of each different type of job (different types of tiles). Project tab uses the information from TimeData tab and the order information received from the customer (which is inputted to this tab by the decision maker) to calculate time needed for each project. The output from the optimization model is also projected back to this tab. Optimizer tab reflects the data from Projects tab into matrices that can be used by the optimizer. This tab also has the

optimization model coded in a free Excel add-in called Solver Studio.[2] We coded the model in PuLP, an open-source Python-based COIN-OR modelling language developed by Stu Mitchell, which is included with SolverStudio. Once the user inputs the project data into Project tab, all they must do is specify the time limit on Optimizer tab and hit Solve Model. The output is projected into Optimizer tab and reflected back to Projects tab using formulas. The user can then copy-paste the optimal assignment of jobs into columns that will assign the jobs to different machines in different sequences. Finally, the Gantt tab uses the input from the Projects tab to display the schedule in a Gantt chart. This is how the Excel tool can also serve as a simulation tool. The assignment that is copy-pasted from optimal columns to assignment columns then can be changed to see the effect of different schedules, which allows the Gantt chart to act as a simulator of different schedules. The tool is also color coded for ease of display. The screenshots from the Excel tool can be found below:

[2] Solverstudio.org

	A	B	C	D
1	Job Type	Machine1-Make-time	Machine2-Make-time	(mins)
2				
3				
4				
5				
6				
7				
8				
9				
10				
11	Machine1			
12	Machine2			

Figure 2.5.1: TimeData Tab

	A	B	C	D	E	F	G	H	I	J
1	Project	Job Type	Output (sq.ft)	Machine1-Make-time (hrs)	Machine2-Make-time (hrs)	Machine1-Make-time (days)	Machine2-Make-time (days)	Percent Complete	Rem. Machine1-Make-time (hrs)	Rem. Machine2-Make-time (hrs)
2	1			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
3	2			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
4	3			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
5	4			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
6	5			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
7	6			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
8	7			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
9	8			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
10	9			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
11	10			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
12	11			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
13	12			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
14	13			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
15	14			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
16	15			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
17	16			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
18	17			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
19	18			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
20	19			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A
21	20			#N/A	#N/A	#N/A	#N/A	0.00	#N/A	#N/A

Figure 2.5.2: Projects Tab – Left Side

	A	B	K	L	M	N	O	P	Q	R	S	T	U	V
1	Project	Job Type	Rem. Machine1-Make-time (days)	Rem. Machine2-Make-time (days)	Due Date	Due Date (Days)	Due Date (hrs)	Ready Date	Ready-time (days)	Ready-time (hours)	To-machine	Position	Optimal-Machine	Optimal-Posit
2	1		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
3	2		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
4	3		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
5	4		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
6	5		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
7	6		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
8	7		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
9	8		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
10	9		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
11	10		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
12	11		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
13	12		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
14	13		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
15	14		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
16	15		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
17	16		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
18	17		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
19	18		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
20	19		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	
21	20		#N/A	#N/A		-30837.00	-246696.00		-30838.00	-246696.00			0	

Figure 2.5.3: Projects Tab – Right side

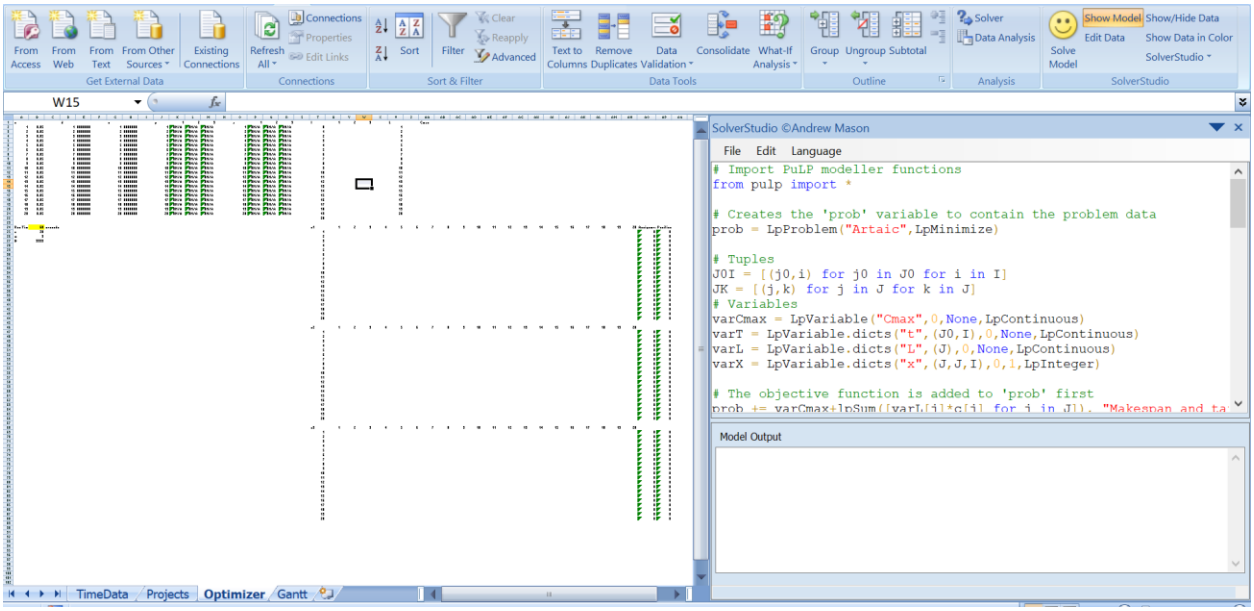


Figure 2.5.4: Optimizer Tab

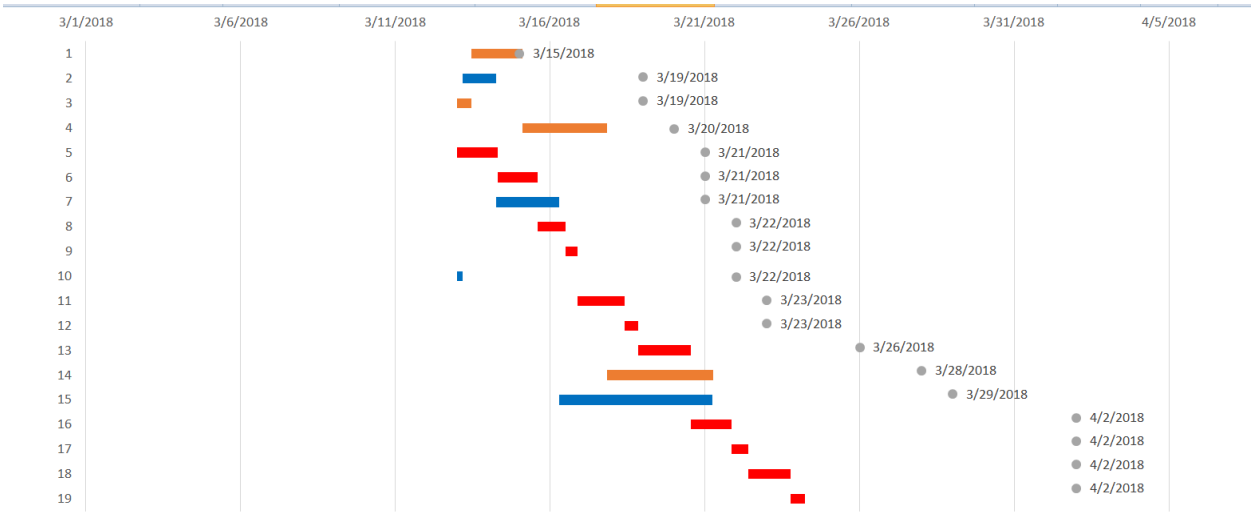


Figure 2.5.5: Gantt Tab

2.5.2.2. **Node.js Implementation**

Node.js is an open-source, cross-platform JavaScript run-time environment for executing JavaScript code server-side.[3] Due to Artaic having a cloud server that can work with Node.js, we also coded our model using the Node.js package GLPK[4]. Our implementation takes in the high-level model file that we wrote on GMPL[5] and the problem data that will be given by the decision maker as an input, solves it using GLPK and prints out all the jobs, the machines that they are assigned to, and their start and completion times in a table format to a text file, which can then be easily put into a Google Gantt chart[6] for visualization purposes. This was the final implementation delivered to Artaic, which is currently being used as far as we know.

---

[3] https://nodejs.org/en/ - https://en.wikipedia.org/wiki/Node.js
[4] https://www.gnu.org/software/glpk/ - https://github.com/hgourvest/node-glpk
[5] https://en.wikibooks.org/wiki/GLPK/GMPL_(MathProg)
[6] https://developers.google.com/chart/interactive/docs/gallery/ganttchart

CHAPTER 3

STOCHASTIC APPOINTMENT SCHEDULING IN A TEAM PRIMARY CARE PRACTICE WITH TWO FLEXIBLE NURSES AND TWO DEDICATED PROVIDERS

My advisors Ana Muriel and Hari Balasubramanian and their former PhD student Joanne Alvarez-Oh from Quinnipiac University have collaborated with me in this project and contributed to the work described in this essay.

3.1. Introduction

According to Alvarez Oh (2015), effective scheduling in primary care practices plays an important role in smoothing patient flow. Many papers have studied the scheduling problem in the outpatient setting, but commonly assume a single step in the patient flow process: the provider step. However, most practices also involve a nurse step prior to the provider step. According to the empirical data analysis in Oh et al. (2013), nurse service time durations are comparable for many appointments to provider service time durations. For example, for routine physicals and well child exams – two common appointment types in primary care – nurses spend as much time with the patients as providers do. In addition, Oh et al. (2013) reports that there is a significant difference in the performance as well as the structure of the optimal schedule in a single provider practice when the nurse step is explicitly considered in the scheduling formulation compared to when it is not.

Another common assumption is a single resource at each step: for example, a solo provider working at the practice. However, the majority of practices (68%) have two or more providers (Bodenheimer and Pham, 2010). In her consultations with practices, she

has noticed that nurses work as a team in prepping patients for provider appointments. Nurses flexibly see patients scheduled on providers' calendars whenever they are available, while providers stay dedicated to their appointment schedules. She calls this a *team primary care practice*.

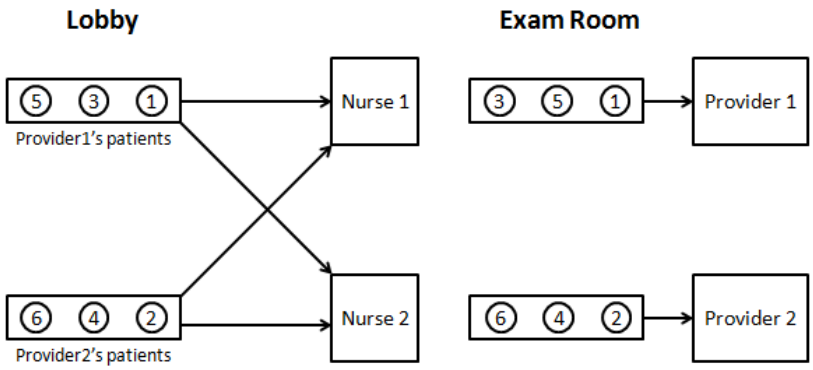


Figure 3.1.1: Patient flow example in a team practice seeing 6 patients

This multi-step patient flow process with multiple human resources at each step coupled with uncertain service times makes the problem difficult from an optimal scheduling viewpoint. Figure 3.1.1 shows an example of a team primary care practice. Patient waiting can occur in the lobby and the exam room – i.e. there are two queueing steps in the patient flow process. Each provider has a set of patients whose appointment times determine the sequence in which they are seen by a nurse. But since the nurses can see any of the two providers' patients, a *crossover* can happen in the schedule when the nurse's service time for a particular patient is long. In Figure 3.1.1, a longer than expected nurse service time for Patient 3 results in Patient 5 seeing the other nurse, and potentially completing the nurse step and being seen by the provider earlier.

Choosing to keep the original sequence versus following the new crossover sequence has implications for both patients' waiting time as well as the idle time of the provider. From

a queuing viewpoint, this is a choice of queue discipline at the second step of a tandem queue. The queue discipline could be either to see patients in the original appointment sequence (no crossovers allowed), or to see patients on a first come first serve basis (allow crossovers). While she has observed that situations that lead to crossover are common in practice, it is not clear what their operational impact is or what a practice's strategy should be.

Alvarez Oh (2015) contributes a new 2-stage stochastic integer programming formulation of the team primary care practice that allows for nurse flexibility and patient crossovers while minimizing a weighted combination of provider idle time and patient wait time. To the best of our knowledge, appointment scheduling in team primary care practices has not been tackled from a mathematical programming perspective before her work. Modeling nurse flexibility and patient crossovers is non-trivial when we consider that the appointment times need to be optimally determined in the first stage, and the resulting patient flows through the practice need to be identified. Computationally the problem becomes more challenging given a set of probabilistic nurse and provider service time scenarios that get realized in the second stage. For a feasible first stage solution of appointment times, sequence changes due to crossover can happen in some scenarios while in other scenarios the original sequence will be retained. Thus, tracking patient flow in each scenario within the framework of a stochastic program poses a modeling challenge which is tackled in this paper.

In this essay, we build upon Alvarez Oh (2015)'s work by helping to answer the following practically relevant questions:

- What does the literature on outpatient scheduling that consider multiple resources and steps look like and how does this work compare to the rest?
- Do optimal team practice schedules consistently exhibit a certain structure that translate to generalizable guidelines?
- How does the (flexible) team practice perform in comparison to a practice in which each nurse is dedicated to a single provider?
- What if we impose that, despite the possibility of crossover, the provider sees patients in the same sequence as their original appointment times? How would the wait and idle times of such a solution compare with a solution that did allow crossovers?
- How do patient no-shows and greater variability in service times affect optimal schedules?

From the computational tractability viewpoint, we demonstrate the use of tightening constraints and a lower bounding procedure to solve realistic instances with a large number of scenarios.

The rest of the essay is structured as follows. In Section 3.2, we provide a brief literature review, particularly focusing on studies considering multiple resources and steps, and exact solution approaches. In Section 3.3, we revisit the team practice, and introduce the mathematical model and solution method by Alvarez Oh (2015). The computational study is divided into two parts: in Section 3.4, we revisit practical guidelines relevant for practices by Alvarez Oh (2015) and built upon them by generalizing her findings with a new set of experiments using lognormal distribution and analyzing the effects of nurse flexibility, patient crossovers, and service time variability. In Section 3.5 we revisit

computational feasibility results from Alvarez Oh (2015) and present our findings regarding the effect of our new lower-bounding technique. In Section 3.6, we revisit Alvarez Oh (2015)'s incorporation of no-shows. In Section 3.7, we summarize our conclusions. This essay is based on our work in Alvarez-Oh et al. (2018).

3.2. **Literature Review**

Outpatient appointment scheduling is a widely studied topic. It refers to a broad class of problems ranging from primary care to specialty clinics and outpatient procedure centers (nuclear medicine and chemotherapy infusion centers) to surgical scheduling. The characteristics of each setting have led to a unique set of assumptions and modeling approaches in the literature. For a comprehensive review of recent literature related to optimization in appointment scheduling, we refer the reader to Ahmadi-Javid, Jalali and Klassen (2016).

Papers in appointment scheduling can be broadly split into three categories based on the time horizon modeled: on the more operational side, papers that focus on a single day, optimizing direct wait time of the patients at the clinic; on the more tactical side, papers that focus on multiple days/weeks, optimizing indirect wait time between an appointment request and an actual appointment day; and papers that focus on a combination of both. In the latter two categories, rather than provide a comprehensive review, we will instead provide a few representative examples. We will explore in more detail papers that focus on a single day, which are the most closely related to our study. Furthermore, we restrict ourselves to papers that consider multiple resources and multiple steps in the patient flow process and use an exact solution approach.

Zacharias and Armony (2017) combine tactical and operational level scheduling by jointly optimizing panel size (the number of patients a primary care physician or a practice cares for in the long term) and the number of offered appointments per day. Their objective is to minimize clinical delay (direct wait time) and appointment delay (indirect wait time) and by using a single-server, two-stage queueing model (an appointment book and the clinic itself) they demonstrate that an "Open Access" policy, where the clinic tries to offer a same-day appointment with high probability, is optimal. Wang & Gupta (2011) consider assigning dynamically arriving phone requests for appointments with patient preferences to multiple primary care providers (i.e. a single step with multiple servers). The problem is motivated as an MDP and solved with heuristics. Other papers that consider multiple days or weeks in their model and multiple stages include Perez et al. (2011) and Perez et al. (2013) (nuclear medicine clinics); Castro & Petrovic (2012) and Conforti et al. (2010) (radiotherapy treatment).

We now discuss papers that schedule a single day. El-Sharo et al. (2015) focus on the overbooking aspect of single-stage, multi-server outpatient scheduling in the presence of no-shows, cancellations and walk-ins. Wang & Fung (2014a), Wang et al. (2015) and Wang & Fung (2014b) study outpatient scheduling with patient preferences using different approaches (MDP, DP and IP, respectively). All of their work involves a single-stage queue with multiple servers (doctors). Tsai and Teng (2014) approach the online scheduling problem for physical therapy in a rehabilitation service which involve a single stage but multiple servers such as therapy equipment. Balasubramanian et al. (2014) consider assigning dynamically arriving same-day requests to multiple primary care providers (i.e. a single step with multiple servers) to maximize the number of same-day

patients seen in the day. They use a stochastic dynamic programming approach. Hahn-Goldberg et al. (2014) consider a multi-stage, multi-server chemotherapy scheduling problem and use the deterministic version to come up with templates that are dynamically adjusted, also considering future requests. Lin (2015) consider an appointment scheduling problem in an eye clinic where the patients go through different pathways (stages) and use different servers (doctors, nurses, optometrists) and propose an adaptive scheduling heuristic with memory (previous distinct schedules are recorded). Liang and Turkcan (2016) focus on a single-stage, multi-server queue in an oncology clinic and propose mixed integer programming models for nurse assignment and patient scheduling. Their unique contribution is they consider patient acuity levels that estimate nurse requirements more accurately.

Surgical scheduling research differs substantially from the rest of the outpatient appointment research since it can be done both in an inpatient and an outpatient setting. However, the outpatient surgical scheduling closely relates to our study. Three examples for outpatient surgical scheduling with multiple steps and resources are Saremi et al. (2013), Bai, Storer and Tonkay (2016) and Neyshabouri and Berg (2016). Saremi et al. (2013) consider 3 stages of the operating room: pre-operation, surgery and recovery in PACU (post-anesthesia care unit). Bai, Storer and Tonkay (2016) tackle the multiple-OR and PACU surgery scheduling problem by using a sample-gradient based algorithm. Neyshabouri and Berg (2016) consider 2 stages of the operating room: surgery and SICU (surgical intensive care unit). They decompose the 2 stages into separate mixed integer linear programs and use a column & constraint generation algorithm. SICU length-of-stay (LOS) is in a different time-scale from PACU stay and surgery duration since SICU LOS

might take a few days while PACU stay and surgery duration usually take only hours. These two time-scale features make their problem different from Saremi et al. (2013) and Bai, Storer and Tonkay (2016). For a more extensive review of recent research on outpatient surgical scheduling with multiple steps and resources, we point to the literature review section in Neyshabouri and Berg (2016). Our essay differs from surgical scheduling because the shared (flexible) resources are upstream in primary care (nurses), as opposed to downstream in surgery (PACU or SICU beds). Patient crossovers (FCFS in second stage) are a direct consequence of this fact, and make the problem more challenging.

The two papers, in addition to Oh et al. (2013), that relate most closely to our work are Castaing et al. (2016) and Kuiper and Mandjes (2015). Castaing et al. (2016) formulate the outpatient scheduling problem in chemotherapy infusion centers as a two-stage stochastic program. One interesting aspect of the problem is that the infusion chairs are considered as the main resource and the nurses are considered as an external resource; nurses can care for multiple patients at the same time. To reduce waste of expensive chemotherapy drugs, a common practice is to delay the preparation of a dose until the patient is ready. The objective is to minimize a weighted combination of wait time and total length of operations, which directly correlates with staff overtime. Because of the weakness in their formulation due to big-M type of constraints, they face high run times for the solution procedure and propose decomposition heuristics that perform well.

Kuiper and Mandjes (2015) approach the outpatient appointment scheduling problem as a tandem-type queuing model with two stages and single server at each stage. The objective is to minimize a weighted combination of patient wait time and provider idle

time. They approximate the service times with their phase-type counterparts, which fit the distribution using first two moments. They propose a recursive method to evaluate the sojourn-time distribution of patients and computationally determine optimal schedules in a transient as well as steady-state environment. They also consider extensions such as heterogeneous service-time distributions and blocking, where the buffer between two stages are finite (clients cannot move between two stages because the servers are busy). They observe the familiar dome-shape pattern in their optimal schedules.

Finally, in our previous work in Oh et al. (2013), we formulate a two-stage stochastic integer program for the single-provider primary care practice composed of a single nurse and provider. The model is optimally scheduled and sequenced patient appointments using stochastic service time of two service steps, nurse and provider, and new patient classifications with the objective of minimization of patient wait time and provider idle time. We suggest the scheduling guidelines obtained by the optimal schedules as well as heuristic schedules capable of accommodating patient time-of-day preferences.

All of the papers above either consider 1) a single-stage with multiple servers, 2) multiple stages with a single server, 3) multiple stages and servers but with a single server at each stage, or 4) multiple stages and servers but deal with the problem deterministically. None of them deal with crossovers. This is understandable because problems involving multiple steps with multiple resources at each step, stochastic service durations and crossovers are intractable using exact approaches. We tackle this issue by exploiting the structural properties of the problem with two servers and by developing tightening constraints and lower bounding techniques.

In contrast to Oh et al. (2013) and the rest of the literature, this essay contributes to current research on outpatient appointment scheduling by formulating a 2-stage stochastic programming model (an exact approach) for a two-stage problem (nurse and provider stages) *with two servers at each stage* (two flexible nurses and two dedicated providers) in a primary care practice. We also consider a homogeneous mix of patients in this study in contrast to the heterogeneous (different patient classifications) in our previous work in Oh et al. (2013). To the best of our knowledge, the problem of appointment scheduling in primary care practices with multiple stages and servers under stochasticity *with flexible resources and patient crossovers (FCFS in second stage)* has not been tackled before using an exact approach.

3.3. **Modeling Approach**

3.3.1. **Description of Team Primary Care Practice**

Alvarez Oh (2015) considers appointment scheduling for a team primary care practice in which two *flexible* nurses share patients while each provider oversees appointments only from his/her own panel (the configuration shown in Figure 3.1.1). At the practice that motivated this study, there are morning and afternoon sessions distinguished by a lunch break. Each session consists of appointment slots, and the length of each slot is 15 minutes. The patient visit consists of the following steps: after check-in, a patient waits in the lobby until a nurse calls (wait time in the lobby); the first available nurse calls the patient into the exam room and examines the patient (nurse service time); after the nurse step, the patient waits in the exam room until her/his primary provider is available (wait time in the exam room); and once the provider finishes with the previous patient, she/he will examine the patient (service time with provider). A provider takes care of the earliest available patient from his own panel after the nurse step. In other words, the provider sees patients in the order of their finish times at the nurse step (first come first serve), instead of the order of appointment times. *Patient crossover* will thus occur when a patient with an earlier appointment may have a long nurse service and end up seeing the provider after a patient with a later appointment.

For service time distributions, Alvarez Oh (2015) focuses on empirical distribution and we extend her work and generalize her results by also considering lognormal distribution. Figure 3.3.1 shows empirical distributions of service time with nurse and provider for *high complexity* (HC) patient visits. This data was collected at the collaborating practice as part of the time study in Oh et al (2013). Type HC involves physicals and complex

conditions, which require long service time with nurses and providers. Community health centers based in low-income and medically underserved neighborhoods often schedule a full day of HC appointments for primary care providers.

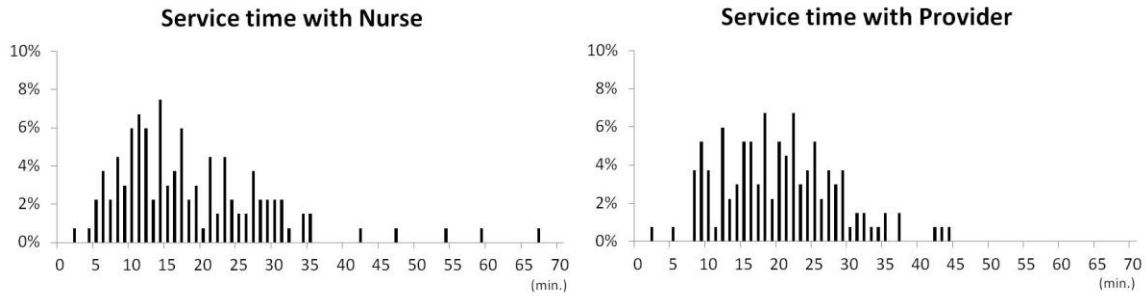


Figure 3.3.1:Distribution of service time with nurse and provider

According to Alvarez Oh (2015), as shown in Figure 3.3.1, service times with both nurse and provider are highly variable. On average, a type HC appointment takes 17.8 min with nurse (standard deviation: 10.7 min.; CV=0.6), and 19.5 min. with provider (standard deviation: 8.2; CV=0.42). Although provider service times tend to be longer, the service time distribution for the nurse step is skewed to the right, leading to nurse visits that are significantly longer than the provider visits. It is apparent that the nurse and provider steps should be effectively coordinated in order to avoid long patient waits or low provider utilization.

In addition to random samples of empirical data on HC appointments, we also use samples from lognormal distributions for nurse and provider service time distributions. The lognormal distribution allows us to control for the mean while increasing the variance of service times, thereby allowing us to test a wider range of cases and establish the generalizability of our findings.

3.3.2. **Integer Programming Formulation**

Alvarez Oh (2015) formulates a mixed integer program to schedule patients into appointment slots. Key features of the model are to accommodate two sequential steps - nurse and provider, multiple human resources at each step, stochastic service times, flexible nurses, providers dedicated to own panels, and patient crossover. She uses a fixed, predetermined appointment length of 15-minutes and consider homogeneous patients. The objective of the model is to minimize a weighted measure of provider idle time and patient wait time across all scenarios. She assumes that the patients punctually arrive at the appointed time since 89% of patients come early or on time based on the data analysis.

She uses the following notation to formulate the problem.

3.3.2.1. **Sets**

I	Set of patients to be scheduled in the session, indexed by $i = 1,\ldots, I$
J_k	Set of patients to be scheduled with provider k, indexed by $j = 1,\ldots, J_k$
S	Set of scenarios, indexed by $s = 1,\ldots, S$
K	Set of providers, indexed by $k = 1,2$

3.3.2.2. **Parameters**

α	Weight for idle time
β	Weight for wait time
$\tau_{i,s}^{N}$	Service time of patient i with nurse under scenario s
$\tau_{j,s}^{P_k}$	Service time of the jth patient to see provider k under scenario s
$f[j,k]$	Patient index (in the overall set of patients in the practice) of the jth patient of provider k

3.3.2.3. **Variables**

$y_{i,s}^{start}$	Start time of patient i with nurse under scenario s
$y_{i,s}^{finish}$	Finish time of patient i with nurse under scenario s
$t_{j,s}^{k}$	Finish time with nurse of the j[th] patient in provider k's panel under scenario s
$z_{j,s}^{k,start}$	Start time of the j[th] patient to visit with provider k under scenario s
$z_{j,s}^{k,finish}$	Finish time of the j[th] patient to visit with provider k under scenario s
$N_{i,s}^{max}$	Maximum of the finish times of patients 1,…, i-1 with nurses under scenario s
$P_{j,s}^{k,max}$	Maximum of the finish times of patients 1,…, j of provider k's panel under scenario s
X_i	Appointment slot assigned to patient i, an integer variable in {0,1,2,...}.

The team practice problem, which we refer to as *Problem TP*, is modeled as the following integer program.

$$Min. \quad \frac{1}{S}\left(\alpha\left[\sum_{k}\sum_{s}\left(\left(z_{J_k,s}^{k,finish}-\sum_{k=1}^{J_k}\tau_{j,s}^{P_k}\right)\right)\right]+\beta\left[\sum_{s}\sum_{i=1}^{n}\left(y_{i,s}^{start}-15X_i\right)+\sum_{s}\left(\sum_{k}\sum_{j=1}^{J_k}\left(z_{j,s}^{k,start}-t_{j,s}^{k}\right)\right)\right]\right) \quad (1)$$

Subject to.

$$y_{i,s}^{start}=0 \quad \forall s\in S, i=1,2 \quad (2)$$

$$z_{0,s}^{k,finish}=0 \quad \forall k\in K, s\in S \quad (3)$$

$$X_i=0 \quad i=1,2 \quad (4)$$

$$y_{3,s}^{start}\geq \min(y_{1,s}^{finish}, y_{2,s}^{finish}) \quad \forall s\in S \quad (5)$$

$$N_{3,s}^{max}\geq \max(y_{1,s}^{finish}, y_{2,s}^{finish}) \quad \forall s\in S \quad (6)$$

$$N_{i,s}^{max}\geq \max(N_{i-1,s}^{max}, y_{i-1,s}^{finish}) \quad \forall i\in 4..I, s\in S \quad (7)$$

$$y_{i,s}^{start}\geq \min(N_{i-1,s}^{max}, y_{i-1,s}^{finish}) \quad \forall i\in 4..I, s\in S \quad (8)$$

$$y_{i,s}^{finish}=y_{i,s}^{start}+\tau_{i,s}^{N} \quad \forall i\in I, s\in S \quad (9)$$

$$y_{i,s}^{start}\geq 15X_i \quad \forall i\in I, s\in S \quad (10)$$

$$t_{j,s}^{k} = y_{f[j,k],s}^{finish} \quad \forall k \in K, j \in J_k, s \in S \tag{11}$$

$$z_{1,s}^{k,start} \geq \min(t_{1,s}^{k}, t_{2,s}^{k}) \quad \forall k \in K, s \in S \tag{12}$$

$$P_{2,s}^{k,max} \geq \max(t_{1,s}^{k}, t_{2,s}^{k}) \quad \forall k \in K, s \in S \tag{13}$$

$$P_{j,s}^{k,max} \geq \max\left(P_{j-1,s}^{k,max}, t_{j,s}^{k}\right) \quad \forall k \in K, j \in \{3..J_k\}, s \in S \tag{14}$$

$$z_{j,s}^{k,start} \geq \min\left(P_{j,s}^{k,max}, t_{j+1,s}^{k}\right) \ \forall k \in K, j \in \{2..J_k - 1\}, s \in S \tag{15}$$

$$z_{J,s}^{k,start} \geq P_{J_k,s}^{k,max} \quad \forall k \in K, s \in S \tag{16}$$

$$z_{j,s}^{k,finish} = z_{j,s}^{k,start} + \tau_{j,s}^{P_k} \quad \forall k \in K, j \in J_k, s \in S \tag{17}$$

$$z_{j,s}^{k,start} \geq z_{j-1,s}^{k,finish} \quad \forall k \in K, j \in J_k, s \in S \tag{18}$$

$$X \geq 0, INT;\ y^{start}, y^{finish}, z^{start}, z^{finish} \geq 0$$

The objective function (1) minimizes a weighted average measure of provider idle time and patient wait time across all scenarios. Note that provider idle time is calculated as the finish time of the last patient minus the sum of the service times of all patients with provider *k* under each scenario. The wait time in the lobby is the difference between the patient's start time with nurse and the appointment time. The wait time in the exam room is calculated as the sum of the differences of the patients' start times with provider and finish times at the nurse step. Constraints (2-4) initialize the start time with nurses for the first two patients, and set the 0th patient finish time with provider *k* to be zero in every scenario. Constraint (5) makes sure that patient 3 is seen by the earliest available nurse, by comparing the finish times of the first two patients with nurses. Constraint (6) calculates the maximum finish time of the first two patients with nurses. Similarly, Constraint (7) keeps track of maximum finish time with nurse for patients 1 to patient *i*-1. The max value for patients 1 through *i*-2 is compared with the finish time of patient *i*-1 with nurses in constraint (8) to find the earliest time a nurse is available to take care of the subsequent patient *i*. Constraint (9) calculates the finish time of patient *i* with nurse, as the start time plus the service time with nurse. Constraint (10) ensures that a nurse can only see a patient after the patients' appointment time (recall that patients arrive

punctually; they are not available any earlier or later than their appointment time). Constraint (11) assigns the nurse finish time of patient i in the full schedule to the finish time with nurse of the corresponding patient *j* in provider *k*'s panel for each scenario *s*; that is, it transfers information from the full ordered set of patients in the practice, to the ordered set of patients in doctor k's panel. Constraints (13-14) track the maximum of the nurse finish times of the first *j*-1 patients scheduled from provider *k*'s panel, and this max value is recursively updated in constraint (15). Constraints (12 and 15) ensure that each provider *k* serves the patient *j* who finishes the nurse step earlier; this is done by comparing the nurse finish times of the first *j*+1 patients in provider *k*'s panel, to account for possible crossover. Constraint (16) ensures the start time of the last patient seen by provider *k* is no sooner than the finish time with nurse for all the patients in the panel. Constraint (17) calculates the finish time of patient *j* as the start time plus service time with provider *k*. Constraint (18) ensures that provider *k* starts to examine the j^{th} patient after seeing the $j\text{-}1^{th}$ patient.

The current model with min and max constraints can be reformulated into a linear program as follows. The max constraints can be easily broken into two inequalities, one for each term in the maximum. For min constraints (5, 8, 12, and 15), she applies a big M method and introduce two sets of binary variables.

$n_{i,s}$ 1 if the earliest nurse available to see patient *i* is the one that serves patient *i*-1, that is, there is some earlier patient that is still seeing the other nurse; 0, otherwise

$p_{j,s}^{k}$ 1 if crossover occurs, that is, the j^{th} patient to see provider *k* is the $j + 1$

patient in his appointment schedule; 0, otherwise

Each of the min constraints is reformulated into two constraints. Let M^1 and M^2 be sufficiently large constants. Constraint (8) is rewritten as constraints (8-1 and 8-2):

$$y_{i,s}^{start} \geq N_{i-1,s}^{max} - M^1 n_{i,s} \quad \forall i \in 4..I, s \in S \tag{8-1}$$

$$y_{i,s}^{start} \geq y_{i-1,s}^{finish} - M^1(1 - n_{i,s}) \quad \forall i \in 4..I, s \in S \tag{8-2}$$

Similarly, constraint (15) becomes constraints (15-1 and 15-2):

$$z_{j,s}^{k,start} \geq P_{j-1,s}^{k,max} - M^2 p_{j,s}^{k} \quad \forall k \in K, j \in \{2..J_k - 1\}, s \in S \tag{15-1}$$

$$z_{j,s}^{k,start} \geq t_{j+1,s}^{k} - M^2(1 - p_{j,s}^{k}) \forall k \in K, j \in \{2..J_k - 1\}, s \in S \tag{15-2}$$

Constraints (5) and (12) follow the same structure.

3.3.3. **Tightening of the Formulation**

The proposed integer programming model is computationally challenging because the number of scenarios needs to be sufficiently high to ensure robustness of the solution. We use 1000 scenarios in our experiments as it provides a good balance between robustness and computational complexity.

Alvarez Oh (2015) states that for instances with more than 5 patients per provider, the general model fails to find a guaranteed optimal solution within 4-hours of computation time. She thus seeks strategies to tighten the formulation. Specifically, she derives tight lower bounds on the big M parameters, stage-based bounds, and additional constraints to eliminate unnecessary processing and strengthen the formulation. In addition, we also propose a lower bound on the optimal cost based on solving the problem for exhaustive and mutually exclusive subsets of scenarios. As we shall see in the computational section, this significantly helps reduce the computational time.

Alvarez Oh (2015) tightens the big M constraints, constraints (5 and 8) with M^1 and constraints (12 and 15) with M^2. M^1 is a bound on the difference of nurse finish times of patient *i*+1 and the maximum of patients 1 through *i*; and M^2 is a bound on the difference of nurse finish times between provider k's patient *j* and *j*+1. The following theorems provide closed form expressions for tight values of M^1 and M^2, respectively. The proofs are provided in Appendix.

Theorem 1. The value of M^1 for each patient under each scenario can be given by

$$M_{i,s}^1 = Max\left\{\tau_{i-1,s}^N + Max\{0,30 - \tau_{i-2,s}^N\}, \underset{r=1,\dots,i-2}{Max}\left\{\tau_{r,s}^N - \sum_{u=r+1}^{i-1}\tau_{u,s}^N\right\}\right\}$$

Theorem 2. The value of M^2 for patient j of provider *k* under scenario s can be provided by

$$M_{j,s}^{2,k} =$$

$$Max\left\{\tau_{i+2,s}^N + \tau_{i+1,s}^N + Max\{0, -\tau_{i,s}^N + 30\}, \underset{r=1,\dots,j,r\ in\ provider's\ k\ panel}{Max}\left\{\tau_{r,s}^N - \sum_{u=r+1}^{i+1}\tau_{u,s}^N\right\}\right\}$$

where *i = f[j,k]*; that is, *i* is the patient number in the overall practice schedule corresponding to the j^{th} patient in provider *k*'s schedule.

Next, we derive a tight lower bound on the optimal solution as follows. Let $\mathcal{S}$ be the full set of scenarios, that is $\mathcal{S}$ ={1,2,…S}.

- Step 1: Divide the set of scenarios $\mathcal{S}$ into a number of exhaustive and mutually exclusive subsets $\{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_n\}$ such that $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \dots \cup \mathcal{S}_n$.

- Step 2: Solve Problem TP under each scenario subset $\mathcal{S}_i$,. Let $C^*_{S_i}$ denote the optimal cost, and let S_i be the size of set $\mathcal{S}_i$, , i=1,2,…,n.
- Step 3: Calculate the lower bound on Problem TP under the full set of scenarios $\mathcal{S}$ as $C^{LB}_{S} = \frac{1}{S}\sum_{i=1}^{n} S_i C^*_{S_i}$.

In addition, Alvarez Oh (2015) proposes stage-based lower bounds (see Santos et al. 1995) for both the nurse and provider steps. At the nurse step she derives lower bounds for the start time and finish time with nurses for each patient under each scenario *s*. At the provider step, she determines bounds on the finish time of the last patient with each provider *k*, which is essentially session completion time of each provider. Her lower bounds are derived using constraints (5-9) to calculate the start time and finish time with nurses without consideration of the appointment times introduced in constraint (10). In other words, the earliest time a patient visit starts can be calculated recursively as the second largest value of the finish times up to patient *i*-1 at the nurse step. This provides tight lower bounds for the nurse start and finish times of patient *i* in the nurse step and the completion time with provider *k* in the provider step under scenario *s*.

She also introduces additional constraints to further tighten the formulation and reduce unnecessary processing. First, the appointment times can be required to be in ascending order, w.l.o.g.; that is, the appointment time of patient *i*+1 must be greater than or equal to that of patient *i*.

$$X_i \leq X_{i+1} \leq \cdots \leq X_I, \quad \forall i \in I \qquad (19)$$

Second, when appropriate, she restricts the appointment schedule to have at most one open slot (slack) between consecutive patients, both within the overall set of patients in the practice [constraint (20)] and within the patients in a provider *k*'s panel [constraint (21), with *i* and *i*+2 as consecutive patients in provider *k*'s panel]. Observe that constraint (21) does not allow for double booking within provider *k*'s panel. This is appropriate for the complex appointments, Type HC, under consideration, as they require long service times; double-booking would highly increase patient wait time. Note also that she is assuming that all patients show up at their appointment time. When no-shows are prevalent, this constraint will be relaxed to allow for double-booking.

$$X_{i+1} - X_i \leq 2, \quad \forall i \in I \quad (20)$$

$$1 \leq X_{i+2} - X_i \leq 2, \quad \forall i \in I \quad (21)$$

She notes that the initial model, without these additional constraints, is fully general and yields solutions that follow these properties. These assumptions are the result of observing the properties of the optimal schedules for small instances. The associated constraints are then added to solve the larger instances. They are also backed up by her analysis of the time data obtained from the observed practice. According to the data, 12% of the patients take over 30 minutes (2 slots) and only 3% over 45 min. with nurse. 10% of the patients spend time with provider over 30 min. and the maximum service time is 44 minutes for the provider.

3.4. **Scheduling Guidelines**

In this section, we use the optimal schedules to derive guidelines that practices can follow to better balance patient wait and provider idle time. In particular, we identify when to add slack, the quality of schedules that ignore the stochastic nature of nurse and provider visit times, and the impact on performance of sharing nurses across providers and allowing for crossovers.

In our experiments, we use both empirical and lognormal visit time distributions, and study small, medium and large instances: 5 patients, 8 patients and 10 patients per provider, respectively. These values are chosen based on observed practice, average service times of 20 minutes and the typical 4-hr length of morning and afternoon sessions. Considering the initial time with the nurse, the slacks in between patients and the variability of service times, the large instances are reasonably sized. The small instances are chosen as half of the size of the large instances and medium instances are chosen as a value in between. Please note that although the observed practice is unlikely to have only 5 patients per provider per session, due to computational challenges, the small instances are analyzed because it allows us to find the exact optimal solutions. We also include results for medium instances to show the progression of the shape of optimal policies as the number of patients increases. Numerical results, however, are only presented for either small instances because the results are optimal, or practical large instances that show how the effect of different factors becomes more pronounced.

In the objective function, we use coefficients of 0.8 for idle time and 0.2 for wait time (Cost Ratio = 4) since we find these weights align best with the desired performance of

the practice in our study. In Section 3.4.6 we explore the sensitivity of the results to this cost ratio.

3.4.1. **Scheduling Guidelines for HC Appointments**

Figure 3.4.1 below shows the optimal schedules for small (5 patients per provider), medium (8 patients per provider) and large (10 patients per provider) instances with HC appointments following the empirical nurse and provider visit times observed in practice. The small instances were solved by Alvarez Oh (2015) while we solved the medium and large instances to optimality. The first important structural property of these schedules is that they are staggered, i.e. the slack for each of the two providers is not scheduled for the same appointment interval.

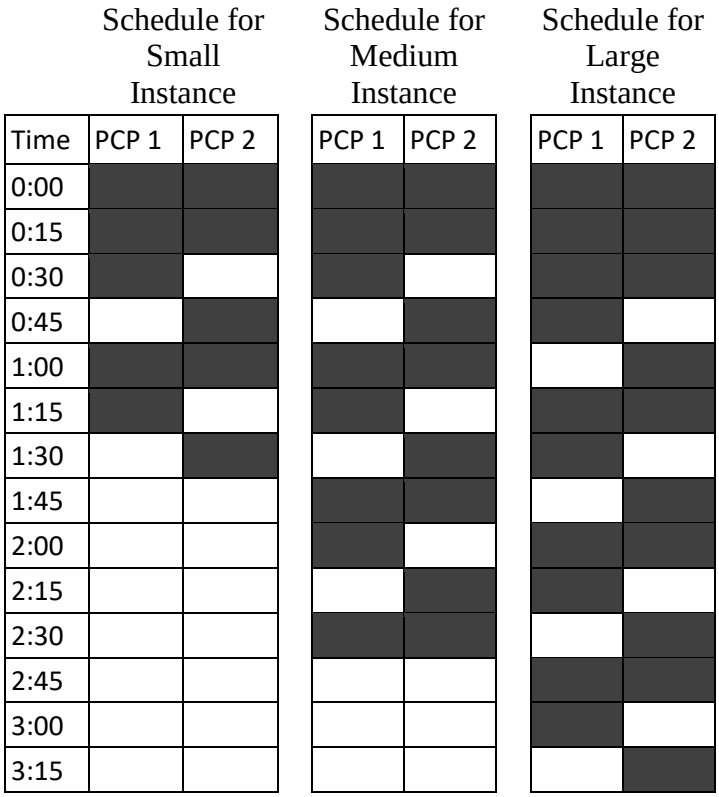


Figure 3.4.1: Schedules for small, medium and large instances

Alvarez Oh (2015) notes that the times given above indicate when the patients are asked to arrive to the practice relative to the practice's working hours. For example, if the practice opens up at 9 AM, the patient scheduled at time 0:00 will arrive at 9 AM. In the practice we have observed, providers are busy with paperwork and other necessary tasks before their first and after their last patients. It is only the idle time between patients that causes inefficiency.

In her effort to derive scheduling guidelines, she compares three schedules: practice policy schedule, identical schedule, and optimal staggered schedule, according to her observation above. The *practice policy schedule* follows the scheduling rules of the practice that inspired our study. Their policy is to book a HC appointment in two 15-min slots, as they regard HC appointments as 30-min appointments – in other words, a 15-min

slack is placed after every HC appointment. The *identical schedule* is determined by the solution of her model with an additional constraint (22) which makes sure that both providers have identical schedules.

$$X_i = X_{i+1}, \quad \forall i \in 1,3,5 \quad (22)$$

Figure 3.4.2 displays the schedules of practice, identical, and staggered policies for small instances.

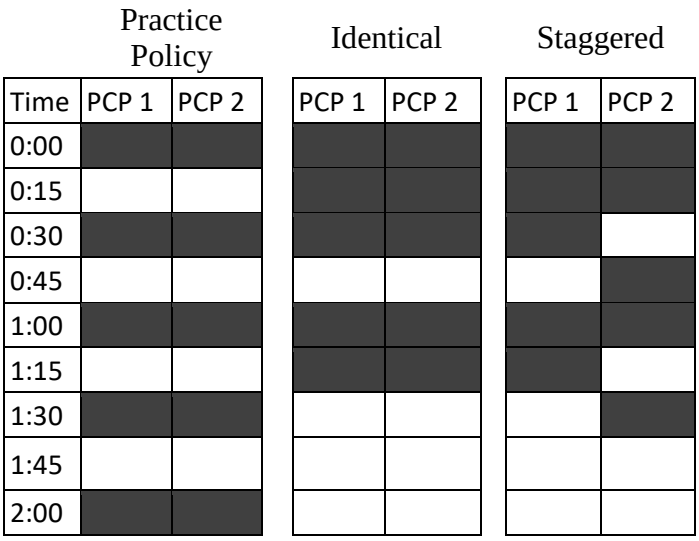


Figure 3.4.2: Schedules of practice, identical, and staggered policies for small instances

As shown in Figure 3.4.2, the identical schedule consists of three appointments followed by slack and two appointments. The first three appointments are consecutively scheduled since the wait time has not accumulated yet. In the staggered schedule, the schedule of provider 1 follows the identical schedule while the schedule of provider 2 assigns slack after two appointments; staggering in this fashion allows a steadier flow into the flexible nurse step. For the team practice under study, her model suggests to schedule two HC appointments followed by slack, a similar scheduling structure proposed in Oh et al. (2013) for the single-provider practice. In addition, since her patient indexing makes the

jth patient of provider 1 be patient i=2j-1, and that of provider 2 be patient i=2j, in the overall practice sequence, and thus gives nurse priority to the patients of provider 1 over provider 2 given the same appointment times, the schedule for provider 1 is more packed (i.e. has fewer and later empty slots).

Next, we compare the wait time, idle time and completion time performance of the three schedules: practice policy, identical and staggered. The values specified below are given for small instances, but medium and large instances also follow similar results. The identical schedule provides about 25% better objective value and 45% lower idle time compared to the practice policy, on average over the 1000 scenarios. In the practice policy, however, the wait time performance is significantly better (3.5 minutes per patient) while the average idle time is more than one hour with only 5 patients per provider. The practice schedule introduces more than enough slack, which causes very low wait times but unsustainably high idle times. The objective difference between the identical and staggered schedules, however, is only 2%. This is because although the staggered schedule improves 17% on wait time, the idle time increases 5% compared to the identical schedule. The numerical values for each policy are given below in Table 3.4.1:

	Practice Policy	Identical	Staggered
Wait Time (per patient)	3.45	13.03	10.79
Idle Time (per provider)	62.38	34.09	35.76
Objective Function Value	106.72	80.59	78.81

Table 3.4.1: Comparison of results for practice, identical and staggered policies – small instance (5 patients per provider) and empirical service times

To provide a better perspective, Alvarez Oh (2015) displays the performance of the practice as the session unfolds, for each of the ten patients in the sequence. Figures 3.4.3

and 3.4.4 show the wait time per patient and idle time between patients for all three schedules. In the figures, we omit the first patient for each provider, patients p1 and p2 in the practice, because they are always scheduled at the beginning of the session and independent of the appointment policy used.

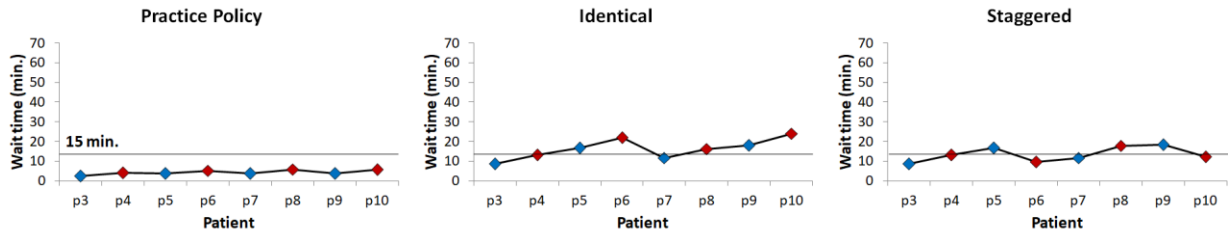


Figure 3.4.3: Average wait time per patient

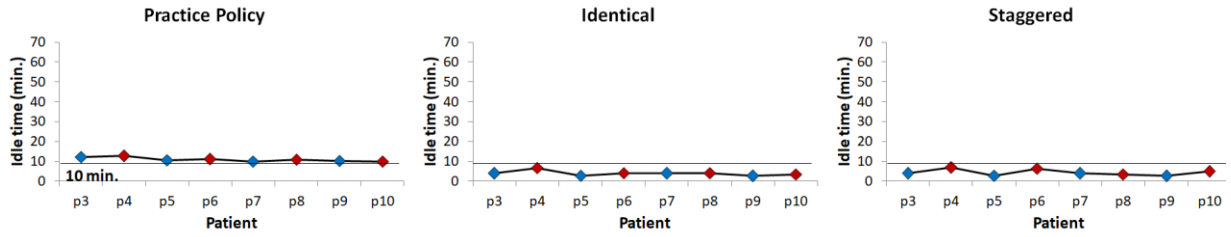


Figure 3.4.4: Average provider idle time between patients

Figure 3.4.3 shows that the wait time per patient followed by the practice policy is way below the 15-min. line but providers go idle more than 10 minutes before seeing each patient, on average. It is because unneeded slack is scheduled, which results in inefficient use of resources. In the identical and staggered schedules, the wait time accumulates and then drops down where slack has been added. The patient wait time of the staggered schedule stays consistently around the 15-min line. Thus, patients in the staggered schedule experience less wait time than those in the identical schedule: three patients wait slightly more than 15 min. in the staggered while five patients wait more than 15 min in the identical.

The idle time of both the identical and the staggered schedules (Figure 3.4.4) is in a similar range and much less than 10-min. per patient after the very first two patients. Next, we study the 90th percentile of wait time per patient and idle time between patients for the three schedules, to see how they perform in the "worst case".

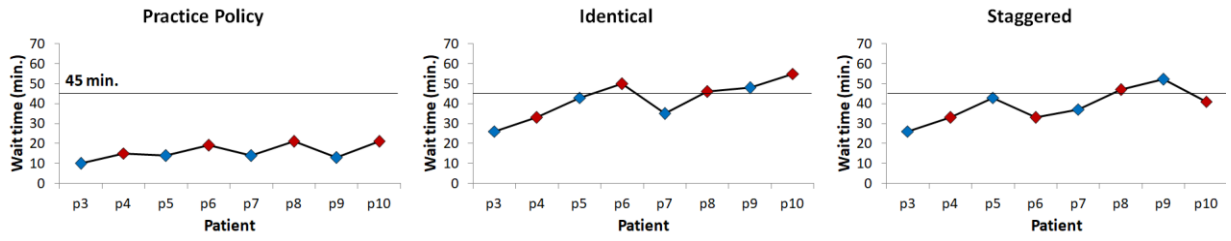


Figure 3.4.5: 90th percentile of wait time per patient

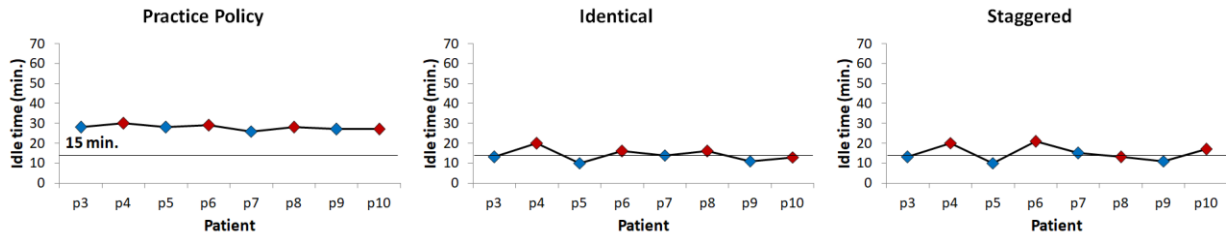


Figure 3.4.6: 90th percentile provider idle time between patients

Figures 3.4.5 and 3.4.6 show that each patient's wait time in the practice policy's worst case is approximately 30 minutes below the wait times associated with the identical and staggered policies. On the other hand, the provider idle time in the practice policy's worst case is almost twice that of the identical and staggered. Comparing wait time between the identical and the staggered policies, only two patients in the staggered schedule wait more than 45 min. while four patients spend more than 45 min. to wait in the identical schedule. Therefore, the staggered schedule performs fairly well.

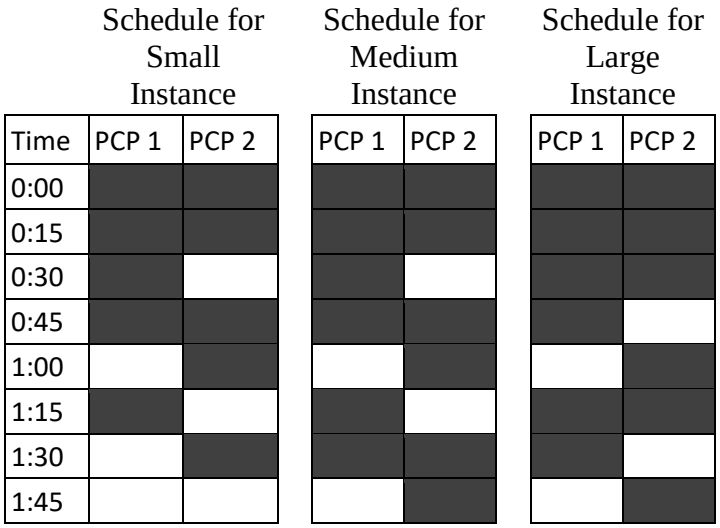
In summary, she derives the following guidelines for the scheduling of HC appointments in the practice under study: 1) team practices are better off staggering slack slots rather

than locating them identically in both providers' schedules; 2) the patients of the provider with nurse priority will have a more packed schedule (less slack); 3) two HC appointments should be followed by a slack slot, except perhaps in the first sequence of the session; 4) no double-booking for a provider in the absence of no-shows, since HC appointments have long and highly variable service times.

3.4.2. **Effect of Service Time Distribution and Variance**

To generalize Alvarez Oh (2015) and our analyses and insights, we test instances with lognormally-distributed nurse and provider service times. This allows us to assess the effect of the distributional shape, while keeping the service time mean and variance constant, as well as the effect of increasing the service time variance while keeping the mean the same.

Figure 3.4.7 below shows the optimal schedules for small (5 patients per provider), medium (8 patients per provider) and large (10 patients per provider) instances with lognormally-distributed service times, with the same mean and variance as in the empirical distribution.



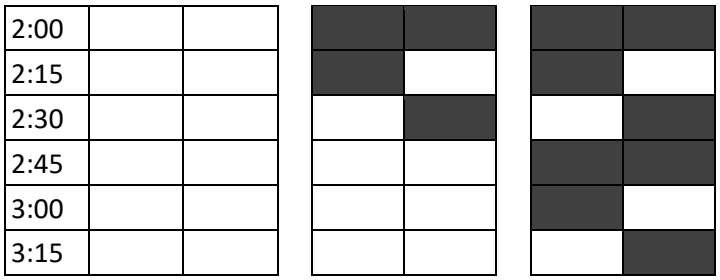


Figure 3.4.7: Schedules for small, medium and large instances with lognormally-distributed service times

As can be seen above, the structures of optimal schedules in instances with lognormally-distributed service times are very similar to those with empirically-distributed service times, and the guidelines we developed for HC appointments still hold.

In order to test the effect of service time variance on the structure of optimal schedules, we created two more sets of instances with lognormally-distributed nurse and provider service times with doubled variance (CV=0.59 for provider and 0.85 for nurse) and quadrupled variance (CV=0.84 for provider and 1.2 for nurse), respectively. The mean is kept constant. As the service time variance increases, the optimal schedule gets more packed. i.e. less slack is introduced; see Figure 3.4.8. Intuitively, this makes sense as the practice is placing a higher weight (0.8) on idle time, relative to wait time (0.2).

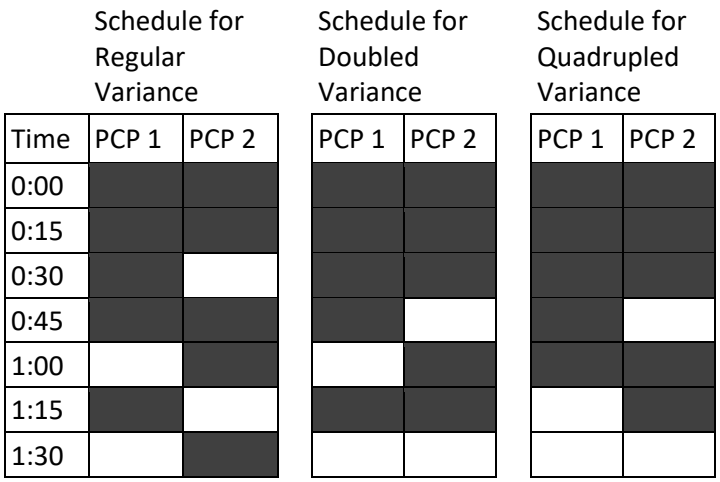


Figure 3.4.8: Schedules for small instances with lognormally-distributed service times with increasing service time variances

3.4.3. **Value of Stochastic Solution (VSS)**

The stochastic nature of nurse and provider visit times results in a computationally challenging, large-scale problem including a wide set of scenarios. What would be the loss in performance associated with the schedule generated by a deterministic model that simply considers average nurse and provider visit times? VSS is calculated as the difference between the expected cost (over all scenarios) of the schedule generated by the deterministic, or single scenario, version of the problem with expected nurse and provider times, and the cost of the optimal schedule suggested by the stochastic problem. VSS is under 2% of the objective function value of the stochastic solution for all cases, which shows that the deterministic solution is actually a very good heuristic for this problem. Interestingly, the schedules generated by the deterministic model are also staggered, which further demonstrates the superiority of staggered schedules and explains the high quality of the deterministic solution. The deterministic model thus provides a very effective and efficient heuristic for the case of interest to our practice, a cost ratio of provider idle time to patient wait time of 4, where packed schedules are attractive. The

VSS, however, increases as more weight is placed on patient wait, being around 10% when the ratio is 1 (0.5:0.5).

3.4.4. **Effect of Nurse Flexibility**

Next, we compare the joint performance in wait time and idle time between two independent single nurse-provider practices and a 2-flexible-nurse, 2-provider team practice, seeing 5 patients per provider. Table 3.4.2 displays the objective function values, mean and 90^{th} percentile of wait time per patient and idle time per provider for the two practices under empirical and lognormal service time distributions; that is, wait time is averaged across all patients and idle time is averaged across the providers. Then, the mean and 90^{th} percentiles are found across 1000 scenarios. The results given from this point on will focus on staggered policy as the optimal solution. We must note that we compared the effect of nurse flexibility statistically using a paired t-test for the means of the objective function over 1000 scenarios and found out that the difference is significant (p-value ~= 0) under all settings.

	Empirical Distribution		Lognormal Distribution	
	Dedicated Nurses	Flexible Nurses	Dedicated Nurses	Flexible Nurses
Mean Wait Time	11.93	10.79	11.75	10.35
Mean Idle Time	37.68	35.76	37.12	34.92
90th Percentile of Wait Time	27.41	25.21	22.01	23.52
90th Percentile of Idle Time	63	56	60	53.55
Objective Function Value	84.15	78.81	82.89	76.57

Table 3.4.2: Comparison of results for dedicated vs. flexible nurses – small instances (5 patients per provider) for empirical and lognormal service times

Table 3.4.2 shows that while the wait time and idle time performance of the team practice (flexible nurses) dominates that of the single provider practices (a.k.a. dedicated nurses), the impact is rather low from an operational viewpoint. The objective function improvements are 6.5% and 7.5% for empirical and lognormal service time distributions, respectively. However, the benefits of nurse flexibility increase with the increase in the number of patients and the variance in service time, as illustrated by Tables 3.4.3 and 3.4.4. Table 3.4.3 shows the 5 patients per provider and 10 patient per provider cases under lognormally distributed service times. Although the objective function improvements are similar in percentage, 7.5% and 6.5% for 5 and 10 patients per provider, respectively, the 90th percentile improvements in wait time and idle time are more pronounced, with a reduction of 5.5 minutes in wait time and 6.5 minutes in idle time for the case with 10 patients per provider while it is 6.5 minutes reduction in idle time and 1.5 minutes increase in wait time for the case with 5 patients per provider.

	5 patients per provider		10 patients per provider	
	Dedicated Nurses	Flexible Nurses	Dedicated Nurses	Flexible Nurses
Mean Wait Time	11.75	10.35	15.57	14.08
Mean Idle Time	37.12	34.92	56.83	54.52

90th Percentile of Wait Time	22.01	23.52	34.62	26.96
90th Percentile of Idle Time	60.00	53.55	87.00	81.50
Objective Function Value	82.89	76.57	153.23	143.57

Table 3.4.3: Comparison of results for dedicated vs. flexible nurses – small instance (5 patients per provider) and large instance (10 patients per provider) for lognormal service times

Finally, Table 3.4.4 shows the regular and quadrupled service time variance cases for lognormal distribution, for 10 patients per provider. The objective function improvement increases from 6.5% for the regular service time variance to 18% for the quadrupled service time variance. Wait time and idle time improvements are also much more pronounced, reaching 8 and 10 minutes for the mean and 21 and 22.5 minutes for the 90th percentile.

	Regular Service Time Variance		Quadrupled Service Time Variance	
	Dedicated Nurses	Flexible Nurses	Dedicated Nurses	Flexible Nurses
Mean Wait Time	15.57	14.08	30.22	22.00
Mean Idle Time	56.83	54.52	90.05	79.61
90th Percentile of Wait Time	34.62	26.96	72.73	51.78
90th Percentile of Idle Time	87.00	81.50	155.55	133.05
Objective Function Value	153.23	143.57	264.95	215.39

Table 3.4.4: Comparison of results for dedicated vs. flexible nurses – large instance (10 patients per provider) and lognormal service times with regular and quadrupled variance

3.4.5. **Effect of Crossovers**

Crossovers naturally happen in the case of flexible nurses as a later patient in the schedule of a provider may complete the nurse step before an earlier patient with a longer nurse visit time. In our model, we assume that the provider will see next the patient that

first becomes available, thus allowing for patient crossover, to minimize uncomfortable patient wait time in the exam room. What would be the loss of performance if crossover is disallowed and providers see patients in the same sequence as they arrive at the practice and are seen by the nurses? Observe that the no-crossover model is easier but not trivial to formulate and solve because tracking patient flow at the flexible nurse step still requires the introduction of binary variables and M constraints; see online companion for the detailed formulation. The provider step, however, follows the original patient sequence and is straightforward. As a result, the second set of binary variables and M constraints (in the original team care problem with crossovers) are no longer necessary and solution speed is improved. In what follows, we compare the joint performance in objective function value, wait time and idle time. We must note that we compared the effect of crossovers statistically using a paired t-test for the means of the objective function over 1000 scenarios and found out that the difference is significant (p-value ~= 0) under all settings.

Table 3.4.5 displays the objective function value, mean and 90^{th} percentile of wait time and idle time for a team practice with crossover vs. a team practice without crossover for 5 patients per provider for empirically and lognormally distributed service times.

	Empirical Distribution		Lognormal Distribution	
	Crossover	No Crossover	Crossover	No Crossover
Mean Wait Time	10.79	11.81	10.35	10.56
Mean Idle Time	35.76	36.98	34.92	37.28
90^{th} Percentile of Wait Time	25.21	27.40	23.52	24.21
90^{th} Percentile of Idle Time	56.00	59.50	53.55	58.00
Objective Function Value	78.81	82.79	76.57	80.77

Table 3.4.5: Comparison of results for models with vs. without crossovers – small instance (5 patients per provider) for empirical and lognormal service times

Table 3.4.5 shows that while the wait time and idle time performance of the practice with crossovers dominates that of the practice without crossovers, the impact is rather low from an operational viewpoint. The benefits of patient crossover, however, increase with the number of patients and service time variance, as shown in Tables 3.4.6 and 3.4.7. Table 3.4.6 shows the cases with 5 patients per provider and 10 patients per provider under lognormally distributed service times. Although the objective function improvements are similar in percentage, around 5% for both 5 and 10 patients per provider, the 90^{th} percentile improvement in idle time is significantly more pronounced for the larger-size problem, with savings of almost 10 minutes, while the wait time is slightly worse, increasing by about 1 minute relative to the no-crossover solution.

	5 patients per provider		10 patients per provider	
	Crossover	No Crossover	Crossover	No Crossover
Mean Wait Time	10.35	10.56	15.38	14.14
Mean Idle Time	34.92	37.28	48.38	56.07
90th Percentile of Wait Time	23.52	24.21	30.41	28.77
90th Percentile of Idle Time	53.55	58.00	74.55	84.05
Objective Function Value	76.57	80.77	138.93	146.27

Table 3.4.6: Comparison of results for models with vs. without crossovers – small instance (5 patients per provider) and large instance (10 patients per provider) for lognormal service times

Finally, Table 3.4.7 shows the performance comparison for the cases of regular vs quadrupled service time variance with lognormal distribution and 10 patients per provider. The objective function improvement of the schedule with vs without crossovers increases from 5% for the regular service time variance to 14.5% for the quadrupled service time variance. The value of the model with crossovers to reduce both wait time and idle time is significantly higher for quadrupled service time variance. The wait time is reduced by 3 minutes for mean and 4 minutes for 90th percentile while the idle time is reduced by 17 minutes for mean and 24 minutes for 90th percentile.

	Regular Service Time Variance		Quadrupled Service Time Variance	
	Crossover	No Crossover	Crossover	No Crossover
Mean Wait Time	15.38	14.14	24.78	27.28
Mean Idle Time	48.38	56.07	72.62	89.41
90th Percentile of Wait Time	30.41	28.77	57.68	61.27
90th Percentile of Idle Time	74.55	84.05	124.10	148.05
Objective Function Value	138.93	146.27	215.32	252.19

Table 3.4.7: Comparison of results for models with vs. without crossovers – large instance (10 patients per provider) for lognormal service times with regular and quadrupled variance

While the impact of crossovers is more significant on idle time under the cost ratio of 4 under consideration, further tests using a cost ratio of 1 show even greater improvements

on wait time. In the most extreme case, with quadrupled service time variance and 90th percentile, the improvement is 38%, from 33 minutes to 20 minutes. These results can be found on the appendix of our essay.

In general, the probability of occurrence of patient crossover increases when the number of patients per provider increases and when the variance in the service time increases. In the small instance with empirical distribution, there was only a 5% chance that a particular patient will experience crossover. Therefore, it is not a surprise that a schedule that considered crossover did not have a significant impact. On the other hand, in the large instance with lognormal service time distribution and quadrupled service time variance, there is a 15% chance that a particular patient in the schedule will experience crossover. A patient flow model that accounts for crossovers is therefore more beneficial in the latter situation.

3.4.6. **Sensitivity to Cost Ratio and Granularity of Appointment Slots**

The appendix contains further sensitivity analyses with respect to the idle to wait time cost ratio and the appointment slot length. Increasing the weight placed on wait time, by varying the idle to wait time cost ratios from the original (0.8:0.2) to (0.66:0.34) and (0.5:0.5) has the expected results. The number of open slots increases. The original practice schedule, leaving one open slot after every patient, becomes then attractive. Increasing the granularity of the appointment slot lengths, by reducing the slot duration from the original 15 minutes to 5 minutes, and even further allowing unrestricted appointment times, results in just modest improvements in the optimal patient wait and provider idle time. This suggests that it may not be worth the added complexity it entails for patients.

3.5. **Computational Performance**

3.5.1. **Effectiveness of Tightened Formulation**

Alvarez Oh (2015) evaluates the computational performance of the model, with and without tightening constraints for the case of five patients per provider. In the model without tightening constraints, she applies the big M method with a sufficiently large M value of 200.

In evaluating the computational performance of various approaches, she reports the *optimality gap*, which can be defined as the relative gap between the objective of the best integer solution and the objective of the best node remaining generated by CPLEX. Her model is implemented with IBM ILOG Optimization Programming Language using CPLEX 12.6 and run on a Windows 8.1 pro and 64 bit with Intel(R) Core™ i7-4770 CPU @ 3.40 GHz, 3401 Mhz, and 32GB RAM. The solution and its performance (speed and quality) may depend on the particular sample of scenarios selected, which is denoted as a replication. This is especially true if the sample is small. For that reason, she generates two replications of 1000 scenarios by randomly sampling from the empirical service time distribution. The model contains 118,002 constraints, 15,000 binary variables, and 10 integer variables. Tables 3.5.1 and 3.5.2 present the optimality gap for the various models with and without tightening constraints after 1 hour and 4 hour run times, respectively.

Gap	Model with large M	Model with tight M	Model with large M & bounds	Model with tight M & bounds
1st replication	46.93%	14.02%	1.46%	1.05%
2nd replication	59.80%	15.59%	1.54%	1.34%

Table 3.5.1 Computational performance for models with and without tightening constraints with allowance of 1 hour

Gap	Model with large M	Model with tight M	Model with large M & bounds	Model with tight M & bounds
1st replication	28.52%	10.54%	1.27%	0.91%
2nd replication	32.16%	10.74%	1.32%	1.13%

Table 3.5.2 Computational performance for models with and without tightening constraints with allowance of 4 hours

As shown in Tables 3.5.1 and 3.5.2, the gap significantly decreases when incorporating tight M values and bounds, with the bounds narrowing the optimality gap far more quickly. It is interesting to note that when running the model for 4 hours, all models produce the same objectives and schedules. However, she cannot confirm the quality of the solution produced by the formulation without any tightening bounds or tight M. Due to the time limit, the search process has not been completed to guarantee optimality; however, the best integer solution has not improved after a certain time. The significant computational effort shows that "one of the incumbents found in the first minutes of the branch and bound process was indeed the best solution that was to be found (Topaloglu 2006)." The objectives and schedules obtained by the model satisfy the goal of the study from the practical viewpoint.

Next, she investigates the computational performance of the tightened formulation. As shown in Figure 3.5.1, at the end of node 0, the gap reaches close to 5.24% in 62 seconds

in the 1st replication and 4.81% in 70 seconds in the 2nd replication. Within 10 mins, the gap is 1.2% in the 1st replication and 1.7% in the 2nd replication. The objective after 10 min. is only 0.03% and 0.2% higher than that after 4hours, respectively. Therefore, the tightened formulation leads to a near optimal solution very quickly.

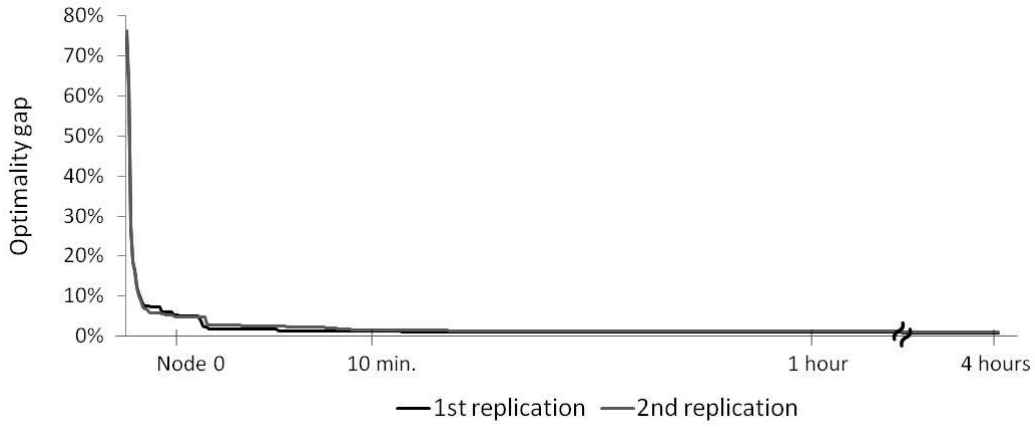


Figure 3.5.1: Computational performance of tightened formulation

The computational experiments for medium and large HC cases and lognormally distributed instances were performed in the same computer as Alvarez Oh (2015)'s and show similar results regarding tightening of the formulation.

3.5.2. **Impact of Lower Bound Based on Solving Mutually Exclusive Scenario Subsets**

For the medium and large cases (8 and 10 patients per provider respectively), we apply the lower bounding technique described in Section 3.3. For the medium case, we split the 1000 scenarios of the medium practice into 50 mutually exclusive groups of 20. It takes 3 hours 45 minutes to solve the 50 groups to create the lower bound, but because that lower bound is very tight, the optimality gap for the original full 1000-scenario problem decreases down to 8% within 10 minutes, and 2% within 4 hours.

Due to the high computation times, for the large case we use the lower bounding technique with even smaller subsets of scenarios: 100 mutually exclusive subsets with 10 scenarios in each subset. It takes 1 hour and 10 minutes to solve the corresponding 100 sub-problems to create a lower bound, but because that lower bound is tight, the optimality gap decreases down to 6.4% within 4 hours (5 hours and 10 minutes total) while solving the global problem.

Thus, we conclude that the lower bounding technique is extremely useful in cutting down the optimality gap for large instances. Without this lower bound, optimality gaps can be as high as 22% for large instances at the end of 4 hours.

When solving instances drawn from various lognormal service time distributions, we find that as the service time variance increases relative to a fixed mean, problem TP gets computationally easier. Again, detailed results can be found in the appendix to our essay.

3.6. **Extension To Incorporate No-Shows**

In this section, we re-iterate the results from Alvarez Oh (2015) regarding no shows:

Until now, we have assumed that scheduled patients always show up to their appointments, since the practice that inspired our study has only a 3% patient no-show rate. In this section, she studies the performance of our models and suggest scheduling guidelines for various no-show rates. She considers no-show rates ranging from 5 to 30 percent (Cayirli and Veral, 2003), in increments of 5 percent. The method to model the different no-show rates within her stochastic programming formulation is to randomly place zero-length visit durations with nurse and provider in the data used to generate the scenarios. This approach captures provider idle time exactly, but results in an

approximation of patient wait times. While the wait time of all the patients seen by the provider is calculated correctly, the objective function also includes the wait time the patient who did not show up would have experienced. As in previous sections, she optimizes appointment times by solving the model with the tightened formulation, but she makes sure to allow for double-booking of appointment slots.

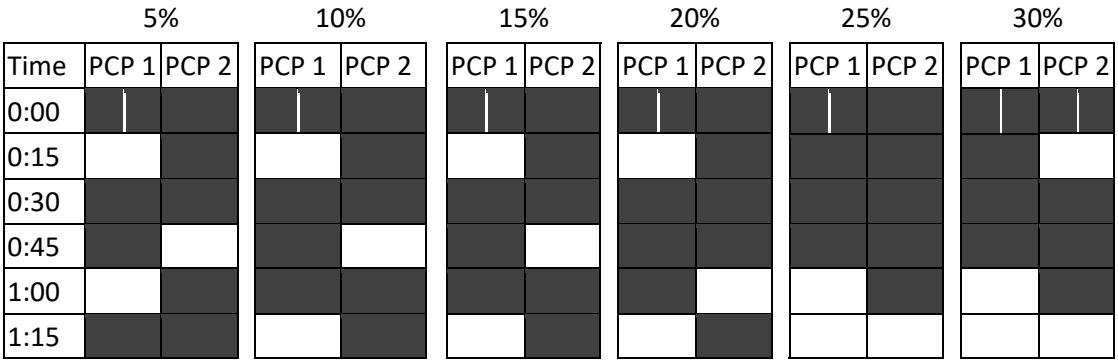


Figure 3.6.1: Schedules under different no-show rates

Figure 3.6.1 displays the schedule under different no-show rates. As expected, the schedule gets packed when no-show rates increase. With a 25 percent no-show rate, slack is no longer needed in the schedule, even when double-booking the first two patients of one of the providers. The optimal schedule under 30% patient no-shows includes one open slot (slack) since both providers are double-booked at the beginning of the session. Double booking the first two patients of provider 1 is a robust scheduling guideline in the range of 5-30% no-shows. The double-booking is followed by an open slot for no-show rates in the range of 5-20%; no slack is necessary under 30% no-shows. It is also interesting to note that the slack position is pushed down, to later in the schedule, as the no-show rates increase. Because of no-shows, wait time and idle time accumulates at a slower pace. In addition, although her formulation allows double-booking any two

consecutive patients, the optimal solutions generated only suggest double-booking the very first two appointments.

3.7. **Conclusion**

In this essay, we build upon a challenging outpatient scheduling problem presented by Alvarez Oh (2015): the team primary care practice. The system can be modeled as a tandem queue: the patients are first seen by a team of nurses in a flexible manner and then seen by their dedicated provider. We restrict our study to the case of two nurses and two providers, which is highly relevant because larger practices often operate in smaller independent teams such as the one studied.

While a first-come-first-serve (FCFS) queueing policy at the second step (provider) is attractive in practice, it results in a significant modelling challenge as patients will crossover and see the provider in a sequence different from that suggested by their appointment times. Alvarez Oh (2015) developed a stochastic mixed integer program to solve this unique problem and we generalize her insights based on the structure of the optimal schedules.

In particular, we draw the following conclusions:

1. The optimal schedule is staggered, introducing slacks for the two providers in different time slots. The later slack is assigned to the provider whose patients are given priority at the nurse step, resulting in a more packed schedule with potentially fewer open appointment slots for that provider.
2. A deterministic model based on average nurse and provider visit times provides a fast, high quality heuristic for the stochastic team care problem, producing schedules

within 2% of optimality for all instances tested under the cost ratio of 4 (provider idle time is weighted 4 times higher than patient wait time) suggested by the practice we collaborated with. When a heavier weight is placed on patient wait time the quality of the deterministic solution deteriorates. For a cost ratio of 1, the optimality gap is around 10% for all levels of service time variance tested.

3. As the variance of the service times increases, the benefits of nurse flexibility and accounting for crossovers on wait time and idle time grow. An optimized schedule allowing for nurse flexibility and patient crossovers leads to significantly lower wait time and idle time when the service time variance is high.
4. As the relative value placed on idle vs. wait time is decreased, the optimal schedules approach the current practice policy where a slack is introduced after every patient.
5. The advantages in reduced wait and idle time of increasing the granularity of appointment slots are rather small and do not outweigh the operational disadvantage in difficult-to-remember appointment times for patients.
6. As the no-show rate increases, the optimal schedules get more packed (introduce less slack), the slacks are scheduled later, towards the end of the time horizon, and double-booking of the first two patients becomes attractive.

Because of the computationally challenging nature of the problem, we developed methods to improve the solution time and optimality gap. Alvarez Oh (2015) had tightened the big-M parameters using the problem structure and generated additional cuts and constraints to close the optimality gap. In addition, we generate a strong lower bound by solving the stochastic mixed integer program for exhaustive and mutually exclusive subset of the full set of scenarios and close the optimality gap even further and solve

larger problem sets. This allowed us to solve realistically-sized problems within a reasonable time frame and optimality gap.

CHAPTER 4

HIERARCHICAL PLANNING AND EXECUTION MODELS FOR JOB SHOP SCHEDULE OPTIMIZATION

My advisor Ana Muriel, her undergraduate research assistant Isabelle Levi and our industry partners Josh Kuledge and Vivek Saxena from Advisory Aerospace have collaborated with me in this project and contributed to the work described in this essay.

4.1. Introduction

Hierarchical job shop schedule planning and execution is a challenging tactical and operational problem (HJSP) observed in aerospace industry. In this problem, the amount of production and inventory on a job shop must be determined on a tactical (usually weekly) and then on an operational (usually daily) level. Tactical level corresponds to planning phase whereas operational level corresponds to execution phase of the manufacturing system. The problem that we are tackling here is inspired by a real-life industrial application observed at an aerospace parts manufacturer.

Aerospace parts manufacturing involves highly complex Bill-of-Materials (BOM) structures with many intermediate and end products that can be represented by a network or a tree graph, a simple example of which can be seen below. The connections (arcs) in the network or tree graph represent how these parts are manufactured or assembled to each other. The nodes in the network or tree graph represent the parts/items themselves. The node at the higher level is called a parent whereas the node at the lower level is called a child. The nodes at the highest level/end items without a parent (top assembly) represent the final parts/assembly that are usually demanded by a customer, although it is not uncommon to have a demand for sub-assemblies/children items as spare parts. Figure

4.1.1 shows a simple BOM tree with one end item. The aerospace manufacturer produces many end items with BOM their trees overlapping. One final level that can be considered on top of the end items is the order level. Customers may demand many end items in a single order, creating a parent level composed of orders.

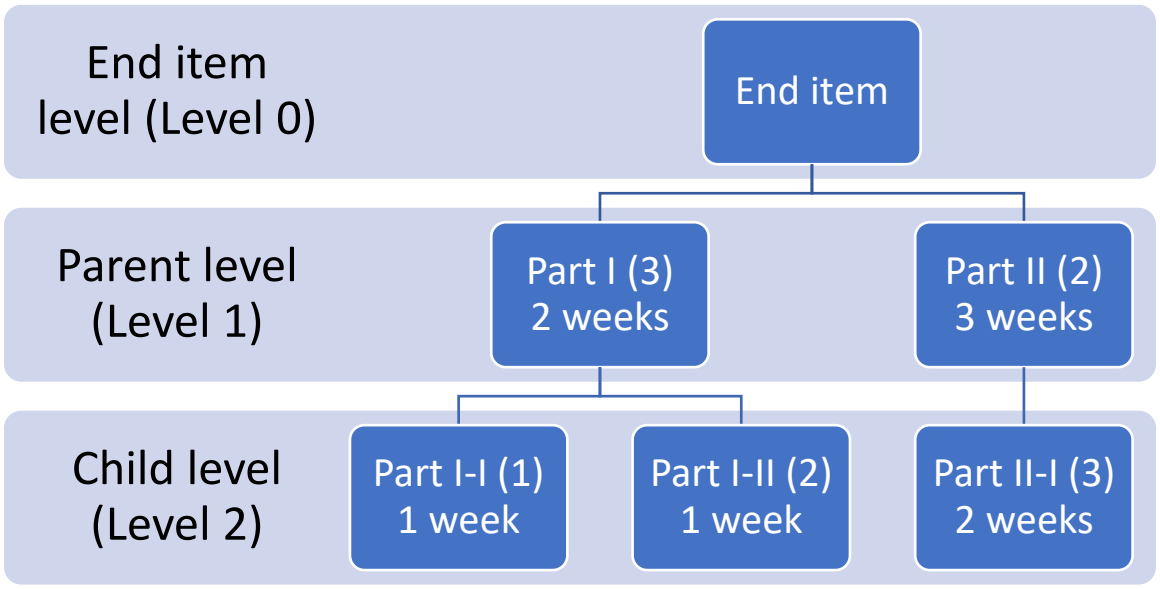


Figure 4.1.1 Bill of materials for a simple end item. Number in parenthesis represents the number of units of that part required to build one unit of the parent part (units per parent, UPP).

A common complication observed in manufacturing settings is the limited amount of resources available. These limited resources might be raw materials, machines or personnel. There are only so many machining hours that can be dedicated to making of the products demanded by the customers. Even if we ignore down-time or maintenance and the job shop is operated 24/7, there are only 24 hours that a machine can be run in a given day. If the limited resources are an issue, the job shop scheduling problem is called "capacitated".

In the literature, the type of job shop scheduling problems described above has a common name: Multi-Level Capacitated Lot-Sizing Problem (MLCLSP). Lot-Sizing here refers to determining how big the production lot should be on a limited resource/machine before switching over to a different product. Lot-Sizing generally implies setups/setup costs that must be completed/paid before a production batch/lot can be started. Therefore, a balance must be achieved between producing large amounts after a setup versus keeping a smaller inventory. Multi-Level refers to the BOM structure described above. Finally, capacitated refers to the limited resources.

HJSP differs from the problems seen in the literature because two separate MLCLSPs must be solved hierarchically at first a tactical level and then at an operational level. The tactical problem's output becomes the operational problem's input. In our case, this corresponds to solving a planning problem that determines the broad production and inventory levels on a BOM structure composed of higher-level items and assemblies over multiple weeks (approximately 12 or 13 weeks corresponding to a quarter). Then, once these production levels are determined, execution of this production plan must be operationalized with a time horizon/production window of 2 weeks (10 or 12 days, depending on whether a 5-day or a 6-day working week is implemented). This is necessary when planning the shop's schedule in the aerospace industry because the production lead time for these products can be up to 12 weeks. The longer-term picture is necessary in order to prioritize what needs to be completed in the short term. It is important to note that the BOM structure needs to be adapted to the timescale modeled in order to appropriately account for the resources needed each period. In the planning problem a part requiring several processing steps on limited resources (as shown in the

BOO) is divided into a sequence of part-steps where several consecutive processing steps are grouped only if they can be completed within a week, so that the necessary capacity used that week is correctly allocated. In the execution problem, the part will be broken down into smaller part-steps, where processing steps are grouped only if they can be completed within one day. Each part-step will be a node in the adapted BOM, whose parent is simply the next part-step in the processing of that part. The operational BOM thus differs from the tactical BOM. In the operational BOM, the items on the tactical BOM are broken down into further sub-assemblies and sub-items that correspond to day-to-day manufacturing operations.

MLCLSP is a so-called "big-bucket" problem where multiple items can be produced over the course of a period (in the case of the planning problem described above, weeks) but the sequencing of items going from one period to another is not considered. A "small-bucket" version of the problem is where only a single item can be produced in a period using a limited resource, due to shortness of the time horizon. The reader is referred to Sahling et al. (2009) for a discussion of the differences between these two problems. In HJSP, both planning and execution problems are technically "big-bucket" problems. This is due to the size of the production in the aerospace industry: even at a daily level, multiple items must be produced on a single machine. However, the execution problem also has characteristics of a "small-bucket" problem: the time horizon is much smaller and the sequencing of items going from one period to another must be considered. This is because considering the sequencing makes a big difference. If a particular item is setup in a given period and that item is the last item produced on the machine in that period (or its setup is completed precisely at the end of the period), the setup state carries over: the

machine does not have to be setup again, if that item is continued to be produced as the first item on the machine in the next period. This is called a setup carry-over. MLCLSP with setup carry-over is a generalized version of the MLCLSP called MLCLSP-L (multi-level capacitated lot-sizing problem with linked lot sizes) in the literature.

One more differentiating/complicating factor in HJSP is the existence of long setups that take multiple days to complete. Although a small percentage of setups in the execution problem exhibit this behavior. It needs to be considered to be able to model the manufacturing setting accurately. This subset of items and machines that need many days of setup will be captured by a set, Γ^L in our modeling approach. These long setups complicate the problem even further – they need special sets and constraints that were not considered in the previous MLCLSP literature. Finally, we also explore the option of backlogs and lost-sales.

MLCLSP has been proven to be NP-hard. Therefore, HJSP as a generalization/extension on MLCLSP can also be considered to be NP-hard. The complexity of the MLCLSP has led researchers to develop many heuristics/meta-heuristics to provide high-quality solutions for industrial sized problems. Therefore, exact methods have not yet been explored to their full potential. The development of more powerful computers and availability of better commercial optimization software in the recent years, together with the use of tight formulations, has led us to consider an exact solution methodology. In this essay, we propose a full mathematical model for the challenging problem of HJSP and offer computational methods to decrease the size of the problem to allow a commercial solver to find a solution in a reasonable amount of computation time. We test our approach on an industrial-size problem provided by the aerospace parts manufacturer

mentioned above. We were able to find solutions for this problem in a reasonable amount of time that renders it a useful tool in practice.

The remainder of the essay is organized as follows. In Section 4.2, we briefly review the relevant literature. In Section 4.3, we present our mathematical modelling approach. In Section 4.4, we define the test application provided by the aerospace parts manufacturer, describe our computational method to reduce the size of the problem and present our results. In Section 4.5 we will conclude the essay and future research directions will be given in Section 5, along with the remainder of the chapters.

4.2. **Literature Review**

The importance of effective production lot sizes in the presence of fixed costs was first recognized at the turn of the 20^{th} century, with the introduction of the classic Economic Ordering Quantity (EOQ) problem by Harris (1913). A rich literature ensued to tackle the many related problems that arise as assumptions of this basic problem are relaxed. In particular, for the case of dynamic demand over T periods, which is known as the Lot Sizing Problem (LSP), Wagner and Whitin (1958) present an exact dynamic programming algorithm that runs in time $O(T^2)$. Linear run time algorithms are now available for this problem; e.g. Wagelmans, van Hoesel and Kolen (1992).

The capacitated version of LSP (CLSP) was introduced by Manne (1958), to account for the existence of resources with limited capacities. Since many excellent reviews of the literature exist, we will not go deep into reviewing these papers. For a relatively recent review, we refer the reader to Karimi, Ghomi and Wilson (2003).

The multi-Level version of CLSP (MLCLSP) was introduced by Billington et al. (1983). This problem considers a BOM structure where components are assembled into sub-assemblies and final products, as discussed above. Similar to CLSP, there are many reviews of MLCLSP. For a recent review, we refer the reader to Buschkühl et al. (2010).

In this section, we will review the papers that are most closely related to our essay and will highlight how our work differs from them.

Chen and Chu (2003) model a supply chain planning problem as an MLCLSP. They develop a heuristic approach to solve this problem based on Lagrangian Relaxation (LR) and local search. Their approach only relaxes the binary setup constraints and forces them to take the value of 1 if the corresponding continuous variable is non-zero. They solve the relaxed linear problem (LP) using the simplex method and update the Lagrange multipliers using a surrogate subgradient method. A feasible solution is obtained at each step and improved by a local search by changing two setup variables at a time. They take advantage of a special structure of the LSP and improve upon the local search, reducing computation time. They use numerical experiments to show the effectiveness of their approach by comparing their solutions to those obtained by a commercial solver. They solve small sized problems (10 items, 6 periods) to 1% optimality gap within a second and their algorithm finds solutions to medium sized problems (60 items, 12 periods) within 360 seconds. The objective function value obtained by the algorithm is 10.5% better than the solution found by the commercial solver when it terminates after 10000 seconds.

Stadtler (2003) proposes a time-oriented decomposition heuristic to solve the MLCLSP with general product structures, multiple constrained resources and setup times. The

heuristic depends on an internally rolling lot-sizing window and the lot-sizing decisions are made sequentially. This approach decomposes the problem into submodels, which are represented by the "Simple Plant Location" model formulation. They test their approach using a computational study and find that their approach provides better solutions than another heuristic by Tempelmeier and Derstroff (1996), as well as the ability to solve larger problem sizes.

Robinson and Lawrence (2004) solve the coordinated version of MLCLSP using an LR heuristic algorithm. The coordinated version of the problem arises when a family of items share the setup costs on a common resource and coordination of these items becomes economically attractive. Specifically, if the items from the same family use the same resource at the same time, their costs are lower. This is similar to sequence-dependent setups with the exception that coordination is considered at the objective function and not at the constraints. Computational experiments show the algorithm to yield a 22.5% cost reduction compared to a current scheduling practice at a manufacturing firm, using an industrial-size problem obtained from the manufacturer.

Sahling et al. (2009) extends the MLCLSP by allowing setup carry overs and considers the version of the problem called MLCLSP-L introduced in the previous section. In this version, partial sequencing of the items (mainly, the first and the last item produced in a period) is important because setup states can be carried over from one period to another. They solve this problem using a fix-and-optimize heuristic which fixes binary setup variables from previous periods and optimizes only a small subset of them. They do a computational study to show that their algorithm provides high-quality solutions with a moderate computational effort. This paper is the most relevant one in the literature to our

study since our execution model also considers partial sequencing of the items similar to this paper. The differences between this paper and ours will be discussed below, along with the rest of the literature.

Mohammadi et al. (2010) discuss the MLCLSP with sequence-dependent setups. They provide an exact model which they determine is impractical to solve for non-small instances. They use rolling-horizon fix-and-relax heuristics to solve them and also analyze trade-offs between quality of the solutions and computation time. Sequence-dependent setups are a generalized version of setup carry overs since they also consider within-period sequencing of items and not just the first and last items in a period. Setups of items depend on which item they are scheduled after. Setup carry over is a special case of sequence dependent setups where if the same items are scheduled one after another at the end of the period and the beginning of the next period, setup becomes zero.

Helber and Sahling (2010) use the fix-and-optimize heuristic introduced by Sahling et al. (2009) to solve MLCLSP with positive lead times. They compare their algorithm to previous algorithms by Tempelmeier and Derstroff (1996) and Stadtler (2003) and conclude that it outperforms them. This paper is also relevant for our purposes because HJSP deals with positive lead times as well.

Ramezanian, Saidi-Mehrabad and Fattahi (2013) solve the MLCLSP with availability constraints. In a manufacturing environment, machines might not be available due to breakdown or maintenance, so this paper includes maintenance planning into MLCLSP using availability constraints. They also consider sequence-dependent setups. They propose a MIP formulation and develop three heuristics based on their formulation and rolling-horizon framework. Computational experiments show that their heuristics solve

small instances (3 items, 2 machines, 2 maintenance tasks, 3 periods) to 5.99% optimality gap within 12.3 seconds and large instances (15 items, 15 machines, 2 maintenance tasks, 15 periods) to 28.02% optimality gap within 1807.5 seconds.

Here is where our essay falls within and differentiates from the rest of the literature mentioned above. HJSP solves two MLCLSPs hierarchically, first at a tactical and then at an operational level. This is necessary in the aerospace application considered because manufacturing lead times span over many weeks (up to 12 weeks) requiring an overall order coordination tool at the larger timescale, as well as an operational tool at the microscale to schedule daily production in the shop. A different BOM structure exists at each timescale. The tactical BOM is broken down into further items and operations to provide the level of detail necessary for the operational BOM. We will refer to the tactical level problem as the planning problem and to operational level problem as the execution problem for the remainder of the paper. The planning problem is a classic MLCLSP similar to the rest of the literature whereas the execution problem is an MLCLSP-L, similar to Sahling et al. (2009) which is a special case of sequence-dependent setups described in Mohammadi et al. (2010). We also consider positive lead times like Helber and Sahling (2010). We solve an actual industrial test problem like Robinson and Lawrence (2004). To the extent of our knowledge, Robinson and Lawrence (2004) and ours are the only papers in the recent literature that solve a real-life problem instead of randomly generated test instances. In working on a real application, some new challenges need to be addressed, such as capturing the current status of the shop in the mathematical model based on the data available.

The main contribution of our essay is thus three-fold: 1) we introduce a hierarchical planning-execution approach to address the long lead times experienced in the aerospace industry. 2) We generalize the MLCLSP literature to consider due dates and corresponding delay penalties, as well as to consider setups that take more than one period at the execution timescale. 3) We apply the model to a real industry setting and address the non-trivial data processing requirements necessary for the model to accurately represent current shop conditions and processing steps at the different timescales. Moreover, setups that take longer than a period are being introduced in our essay for the first time. Finally, while the literature has focused on heuristics, we solve our problem using an exact approach (mathematical model implemented using a commercial solver) within reasonable times for application in practice. As can be seen above, most of the recent MLCLSP literature focuses on LR and time-decomposition/rolling-horizon approaches to solve realistically-sized instances. However, recent developments in computer hardware and optimization software plus a novel computational method that we use to reduce the size of the problem allowed us to solve our problem optimally, in contrast to recent literature which focus heavily on heuristic methods.

We will describe the problem and our solution method (MIP) in the next section.

4.3. **Mathematical Models**

We first present the planning model where the plan is over a few months to allow for sufficient time to complete full orders. We then use the rough production schedule in the early periods output of that model as input to the execution model, where a detailed schedule will be generated. Therefore, we will be considering two different time scales

for two different but related problems. Weeks and days are the case for our particular application, but other timescales may be appropriate for other cases. We will use the example on Figure 4.1.1 to explain how BOM changes on different time scales and how they relate to each other:

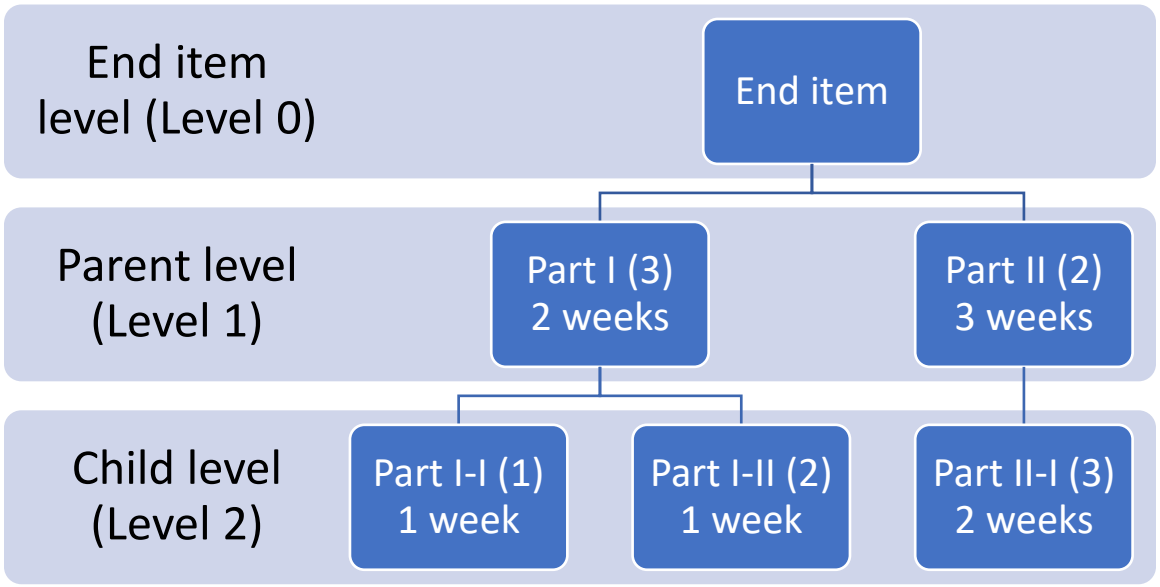


Figure 4.3.1 BOM in weeks (same as Figure 4.1.1)

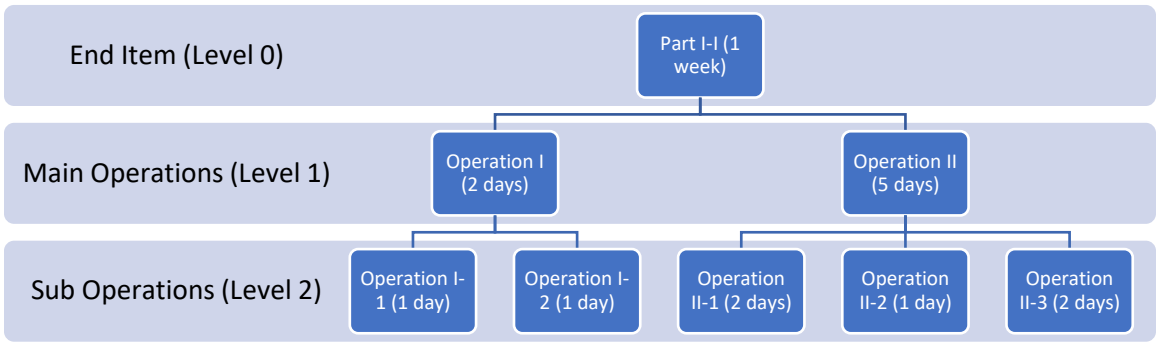


Figure 4.3.2 BOM in days (BOO)

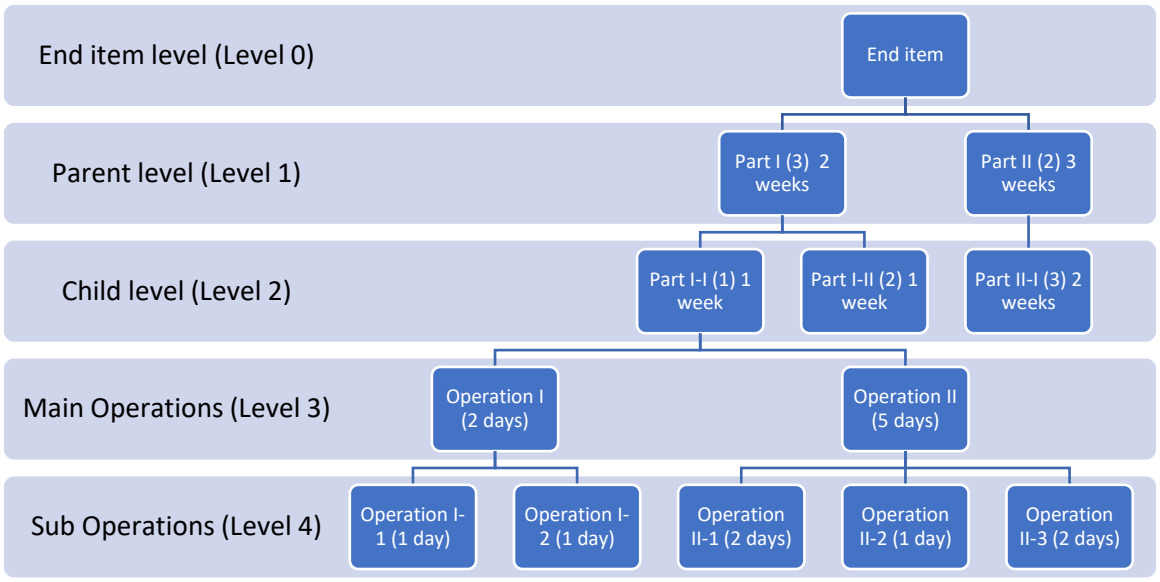


Figure 4.3.3 BOM+BOO

As you can see above, weekly BOM break down into further operations (part-step numbers or PSNs) in the daily BOM (BOO). Therefore, the output from the weekly planning model feeds into the daily execution model. It is technically possible to create a

plan considering both levels, but the orders are not finalized and are considered only as forecasts at the planning phase. Therefore, it is not realistic or necessary to consider every minute detail of operations during the planning. Only the first couple of periods (in our case, 2 weeks) are finalized and fixed in the planning horizon and therefore that plan is fed into the execution model. Once weekly plans are updated with finalized orders, the execution models can be re-run, adjusting to dynamic demand as necessary.

Again, these examples in weeks and days are in line with our current application, but we will refer to them as planning periods and execution periods to be more general for the remainder of the modeling sections. Below are the assumptions implicitly made in the formulation given the length of the periods:

- A single setup is needed for the same part in a period.
- Several processes may be needed to complete a part in one period.
- Lost sales penalty is proportional to the number of periods from the order due date until the end of the planning period and the number of units undelivered. This penalty can be modified by the user.
- The length of the execution period is assumed to be equal to the first two planning periods for simplicity. This is easily generalizable.
- Units Per Parent (UPP) matrix is used to define the BOM. This is the amount given in parentheses in the BOM examples above and only applies to direct children of a parent.
- Units Per Order (UPO) matrix is used to define the number of units of each item needed for an order. This includes indirect children, unlike the UPP. This is

calculated by solving a simple one-period uncapacitated optimization problem using the BOM.

- End items are also referred to as top assemblies for the modeling sections.
- If an order is started in period 1 and its lead time is 2 periods, it will be delivered in period 3. Production may be completed at the end of period 2 but packaging and shipping preparation are also necessary, so it makes intuitive sense to have it ready for the customer on the next period.
- The two models, planning and execution, are connected through vector parameters U^1 and U^2 which capture the planned production for the first and second periods, respectively, given by the planning model, and set the requirements input for the execution model.
- ϵ is a small reward or penalty used to direct the model towards particular objectives, and this can again be fine-tuned by the user.

4.3.1. **Planning Model**

4.3.1.1. **Sets**

$I := Set\ of\ all\ parts$

$I^c := Set\ of\ all\ parts\ that\ require\ constrained\ resource$

$K := Set\ of\ all\ parts\ with\ children\ in\ the\ BOM,$
$used\ to\ define\ Units\ Per\ Parent\ (UPP)\ matrix$

$T := Set\ of\ periods\ in\ the\ planning\ process\ (large\ timescale; e.g.\ weeks)$

$J := Set\ of\ constrained\ resources$

$O := Set\ of\ orders$

$\Gamma := Set\ of\ pairs\ (i,j)\ where\ item\ i\ requires\ a\ setup\ on\ resource\ j$

$\Theta := Set\ of\ pairs\ (i,o)\ where\ item\ i\ is\ required\ to\ complete\ order\ o$

4.3.1.2. **Parameters**

$p := number\ of\ periods\ in\ the\ planning\ horizon$

$Q_o := Number\ of\ units\ demanded\ in\ order\ o$

$B_{o,i} := indicates\ whether\ order\ o\ requires\ part\ i\ as\ final\ customer\ deliverable$

$D_o := Due\ date\ of\ order\ o$

$R_{i,j} := time\ required\ to\ process\ part\ number\ i\ on\ machine\ j$

$S_{i,j} := set\ up\ time\ of\ job\ i\ on\ resource\ j$

$C_{j,t} := Capacity\ of\ constrained\ resource\ j$

$M_{i,j,t} := Maximum\ possible\ production\ of\ part\ i\ on\ machine\ j\ in\ a\ period$

$N_{i,k} := Number\ of\ units\ of\ part\ i\ needed\ to\ build\ a\ unit\ of\ k\ (UPP)$

$N_i^o := Total\ number\ of\ units\ of\ part\ i\ needed\ to\ process\ order\ o,$

$including\ all\ levels\ of\ the\ BOM\ (UPO)$

$L_i := part\ i\ lead\ time$

$A_{i,t}$:

$= part\ availability\ or\ previously\ scheduled\ completion\ of\ part\ i\ in\ initial\ period\ t$

$P_o := order\ o\ shortage\ penalty\ (per\ unit)$

$H_i := part\ i\ production\ revenue\ towards\ completing\ an\ order$

$V_i^0 := initial\ inventory\ of\ part\ i$

$\epsilon := Small\ reward/penalty$

4.3.1.3. **Variables**

$u_{i,o,t} := qtty\ of\ part\ i\ starting\ production\ at\ time\ t\ destined\ for\ order\ o;$

$these\ parts\ will\ be\ completed\ at\ time\ t + L_i$

$w_{o,t} := qtty\ of\ top\ assembly\ in\ order\ o\ ready\ to\ ship\ at\ time\ t$

$a_{i,o,t} := part\ i\ previously\ scheduled\ completions\ at\ time\ t\ assigned\ to\ order\ o$

$V_{i,o,t} := part\ i\ inventory\ at\ time\ t\ destined\ for\ order\ o\ where\ t \in \{0 \cup T\}$

$z_{i,j,t} := 1\ if\ part\ i\ is\ produced\ on\ resource\ j\ in\ period\ t, 0\ otherwise$

4.3.1.4. **Model**

$$minimize\ Penalty: \sum_{o \in O:\, D_o <= p} (P_o(p + 1 - D_o)(Q_o - \sum_{t \in T} w_{o,t})) + \sum_{o \in O, t \in T:\, t > D_o} (P_o(t - D_o)w_{o,t}) - \sum_{o \in O, t \in T:\, t < D_o} \left(\epsilon(D_o - t)w_{o,t}\right) + \sum_{i \in I, o \in O} H_i\left(N_i^o - \sum_{t \in T}\left(a_{i,o,t} + u_{i,o,t}\right)\right) \quad (1)$$

$subject\ to$

$$\sum_{t \in T}\left(a_{i,o,t} + u_{i,o,t}\right) \leq N_i^o\ \forall (i \in I, o \in O) \quad (2)$$

$$\sum_{t \in T} w_{o,t} \leq Q_o\ \forall (o \in O) \quad (3)$$

$$\sum_{i \in I^c}\left(R_{i,j}\left(\sum_{o \in O} u_{i,o,t}\right) + S_{i,j} z_{i,j,t}\right) \leq C_{j,t}\ \forall (t \in T, j \in J) \quad (4)$$

$$u_{i,o,t} \leq min\left(M_{i,j,t}, N_i^o\right) z_{i,j,t}\ \forall \left(i \in I^c, j \in J, o \in O, t \in T: S_{i,j} > 0\right) \quad (5)$$

$$V_{i,o,t} = V_{i,o,t-1} + a_{i,o,t} + u_{i,o,t-L_i} - \sum_{k \in K} N_{i,k} u_{k,o,t} - B_{o,i} w_{o,t}$$

$$\forall(i \in I, o \in O, \ t \in T : t > L_i) \quad (6)$$

$$V_{i,o,t} = V_{i,o,t-1} + a_{i,o,t} - \sum_{k \in K} N_{i,k} u_{k,o,t} - B_{o,i} w_{o,t}$$

$$\forall(i \in I, o \in O, t \in T : t \leq L_i) \quad (7)$$

$$\sum_{o \in O} V_{i,o,0} \leq V_i^0 \ \forall(i \in I) \quad (8)$$

$$\sum_{o \in O} a_{i,o,t} \leq A_{i,t} \ \forall(i \in I, t \in T) \quad (9)$$

Objective function (1) minimizes the total penalty, which is the sum of penalties for the unsatisfied orders at the end of the planning period, orders that are due past their deadline, a small reward for orders satisfied before their deadlines and revenue loss for the production requirements that are unfulfilled. Constraint (2) limits overproduction and constraint (3) limits over-delivery. Constraint (4) establishes capacity for a period for the constrained resources. Constraint (5) ensures that production is not possible without setup. Constraint (6) and (7) are inventory balance constraints based on the given period and lead time of items. If the given period is lower than or equal to the part's lead time, the demand for that part can only be satisfied by the inventory or scheduled completions. If the current period is greater than the part's lead time, demand can be satisfied by the production within the planning horizon. Constraint (8) establishes initial inventory and constraint (9) establishes availability of parts undergoing processing at the beginning of the planning horizon (over the lead time window). Some parts can be scheduled to be available later as a result of a quality non-conformance event or other special circumstances. For full generality, we allow parameter A and variable a to be positive for any period in the planning horizon.

We now present the execution model. We only highlight the differences between the planning and execution model since execution model is based on the planning model but considers additional operational requirements as necessary.

4.3.2. **Execution Model**

4.3.2.1. **Additional/Modified Sets**

$T := Set\ of\ periods\ in\ the\ execution\ process\ (short\ timescale; e.g.\ days)$

$\Gamma^L := Set\ of\ pairs\ (i,j)\ where\ item\ i\ requires\ a\ large\ setup$

$(longer\ than\ an\ execution\ period)\ on\ resource\ j$

4.3.2.2. **Additional/Modified Parameters**

$p^e := Number\ of\ periods\ in\ execution\ horizon$

$S_{i,j}^L$

$:= Number\ of\ periods\ (rounded\ up)\ the\ setup\ of\ part\ i\ takes\ on\ resource\ j, for\ (i,j)$

$\in \Gamma$

$F_{i,j} :=\ 1\ if\ part\ i\ is\ at\ time\ 0\ being\ produced\ in\ machine\ j, 0\ otherwise$

$U_{i,o}^1 := Units\ of\ part\ i\ \ to\ be\ produced$

$/allocated\ for\ order\ o\ on\ planning\ period\ 1$

$U_{i,o}^2 :=\ Units\ of\ part\ i\ to\ be\ produced$

$/allocated\ for\ order\ o\ on\ planning\ \ period\ 2$

$U_{i,o}^1\ and\ U_{i,o}^2\ are\ given\ by\ the\ output\ of\ the\ planning\ model.: U_{(i,o)}^1$

$= u_{i,o,1} +\ a_{i,o,1}\ and\ U_{i,o}^2 = u_{i,o,2} + a_{i,o,2}$

4.3.2.3. **Additional/Modified Variables**

$f_{i,j,t} := 1\ if\ part\ i\ is\ the\ first\ part\ to\ be\ produced\ in\ period\ t,$

$continuing\ from\ period\ t-1, 0\ otherwise$

$l_{i,j,t} := 1\ if\ part\ i\ is\ the\ last\ part\ produced\ in\ period\ t\ continuing\ to\ period\ t + 1, 0\ otherwise$

$\varphi_{i,o}^{11} := underproduction\ relative\ to\ plan\ in\ the\ first\ planning\ period$

$\varphi_{i,o}^{12} :=\ underproduction\ in\ the\ second\ week\ of\ production\ planned\ for\ first\ planning\ period$

$\varphi_{i,o}^{22} := underproduction\ relative\ to\ plan\ in\ the\ second\ planning\ period$

$z_{i,j,t} := 1\ if\ setup\ is\ \boldsymbol{completed}\ for\ part\ i\ on\ resource\ j\ in\ period\ t, 0\ otherwise$

$e := number\ of\ execution\ periods\ per\ planning\ period$

4.3.2.4. **Model**

$$minimize\ Penalty\colon \sum_{i\in I, o\in O} P_o\left(\varphi_{i,o}^{11}+\varphi_{i,o}^{12}+\varphi_{i,o}^{22}\right)+\epsilon\left(\sum_{o\in O\colon D_o\le p} P_o(p+1-D_o)\left(Q_o-\sum_{t\in T} w_{o,t}\right)+\sum_{o\in O, t\in T\colon t>D_o} P_o(t-D_o)w_{o,t}-\sum_{o\in O, t\in T\colon t<D_o}\epsilon(D_o-t)w_{o,t}+\sum_{i\in I, o\in O} H_i\left(N_i^o-\sum_{t\in T}\left(a_{i,o,t}+u_{i,o,t}\right)\right)\right) \quad (10)$$

$subject\ to$

$$\sum_{t\in T}\left(a_{i,o,t}+u_{i,o,t}\right)\le U_{i,o}^1+U_{i,o}^2\ \forall(i\in I, o\in O) \quad (11)$$

$$\sum_{t\in T} w_{o,t}\le Q_o\ \forall(o\in O) \quad (12)$$

$$\sum_{i\in I2} R_{i,j}\left(\sum_{o\in O} u_{i,o,t}\right)+\sum_{i\in I^c\colon (i,j)\in\Gamma-\Gamma^L}\left(S_{i,j}z_{i,j,t}\right)\le C_{j,t}\ \forall(t\in T, j\in J) \quad (13)$$

$$\sum_{s\in\{t-S_{i,j}^L+1..t\}}\left(\sum_{k\in I^c} R_{k,j}\left(\sum_{o\in O} u_{k,o,s}\right)+\sum_{k\in I^c\colon (k,j)\in\Gamma-\Gamma^L} S_{k,j}z_{k,j,s}\right)+S_{i,j}z_{i,j,t}\le \sum_{s\in\{t-S_{i,j}^L+1..t\}} C_{j,s}\ \forall\left((i,j)\in\Gamma^L, t\in\{S_{i,j}^L..p^e\}\right) \quad (14)$$

$$\sum_{i\in I^c} R_{i,j}\left(\sum_{o\in O} u_{i,o,t}\right)+\sum_{i\in I^c\colon (i,j)\in\Gamma-\Gamma^L}\left(S_{i,j}z_{i,j,t}\right)+\sum_{i\in I^c\colon (i,j)\in\Gamma^L}\left(\sum_{s\in\{t+1,..,t+S_{i,j}^L-1\}} C_{j,t}\,z_{i,j,s}+\ z_{i,j,t}max(\ 0,(S_{ij}-\sum_{s\in\{t-S_{i,j}^L+1,..,t-1\}} C_{js}))\right)\le C_{jt}\ \forall(j\in J, t\in T) \quad (15)$$

$$z_{i,j,t} + z_{k,j,s} \leq 1 \ \forall\big((i,j) \in \Gamma^L, (k,j) \in \Gamma^L, t \ \in \{S^L_{i,j}..p^e\}, s \ \in \{t - S^L_{i,j} + 1..t\}: k \neq i\big) \quad (16)$$

$$z_{i,j,t} + z_{i,j,s} \leq 1 \ \forall \Big((i,j) \in \Gamma^L, t \ \in \{S^L_{i,j}..p^e\}, s \ \in \{t - S^L_{i,j} + 1..t-1\Big) \quad (17)$$

$$z_{i,j,t} + z_{k,j,s} \leq 1 \ \forall \begin{pmatrix} (i,j) \in \Gamma^L, (k,j) \\ \in \ \Gamma - \Gamma^L, \quad t \ \in S^L_{i,j}..p^e, s \ \in t - S^L_{i,j} + 1..t-1 \end{pmatrix} \quad (18)$$

$$z_{i,j,t} = 0 \ \forall \ \Big((i,j) \in \Gamma^L, t \ \in \{1..S^L_{i,j} - 1\}\Big) \quad (19)$$

$$u_{i,o,t} \leq min\big(M_{i,j,t}, U^1_{i,o} + U^2_{i,o}\big)\big(z_{i,j,t} + f_{i,j,t}\big) \ \forall\big(i \ \in I^c, j \ \in J, o \ \in O, t \ \in T: S_{i,j} > 0\big) \quad (20)$$

$$V_{i,o,t} = \ V_{i,o,t-1} + \ a_{i,o,t} + u_{i,o,t-L_i} - \textstyle\sum_{k \in K} N_{i,k} u_{k,o,t} \ - \ B_{o,i} w_{o,t} \ \forall(i \ \in I, o \ \in O, \ t \ \in T: t \ > \ L_i) \quad (21)$$

$$V_{i,o,t} = \ V_{i,o,t-1} + \ a_{i,o,t} - \sum_{k \in K} N_{i,k} u_{k,o,t} - B_{o,i} w_{o,t} \ \forall(i \ \in I, o \ \in O, t \ \in T: t \ \leq \ L_i) \quad (22)$$

$$\textstyle\sum_{o \in O} V_{i,o,0} \leq \ V^0_i \ \forall(i \ \in I) \quad (23)$$

$$\textstyle\sum_{o \in O} a_{i,o,t} \leq \ A_{i,t} \ \forall(i \ \in I, t \ \in T) \quad (24)$$

$$l_{i,j,t} \geq \ f_{i,j,t+1} \ \forall(i \ \in I^c, j \ \in J, t \ \in 1..p-1) \quad (25)$$

$$z_{i,j,t} + \textstyle\sum_{o \in O} u_{i,o,t} \geq \ f_{i,j,t+1} \ \forall(i \ \in I^c, j \ \in J, t \ \in 1..p-1) \quad (26)$$

$$z_{k,j,t} \leq \Big(2 - \big(f_{i,j,t} + l_{i,j,t}\big)\Big) \ \forall(i, k \ \in I^c, i \neq k, j \ \in J, t \ \in T) \quad (27)$$

$$\textstyle\sum_{i \in I^c} f_{i,j,t} \leq \ 1 \ \forall(j \ \in J, t \ \in T) \quad (28)$$

$$\textstyle\sum_{i \in I2} l_{i,j,t} \leq \ 1 \ \forall(j \ \in J, t \ \in T) \quad (29)$$

$$f_{i,j,1} = \ F_{i,j} \ \forall(i \ \in I^c, j \ \in J) \quad (30)$$

$$\varphi^{11}_{i,o} \geq U^1_{i,o} - \textstyle\sum_{t \in T: \, t \leq e} u_{i,o,t} \ \forall(i \ \in I, o \ \in O) \quad (31)$$

$$\varphi^{12}_{i,o} \geq U^1_{i,o} - \textstyle\sum_{t \in T: \, t \leq 2e} u_{i,o,t} \ \forall(i \ \in I, o \ \in O) \quad (32)$$

$$\varphi^{22}_{i,o} \geq U^2_{i,o} - \textstyle\sum_{t \in T: \, t \leq 2e} u_{i,o,t} \ \forall(i \ \in I, o \ \in O) \quad (33)$$

Objective function (10) minimizes penalty similar to objective function (1) from planning model with the exception that the main target is to limit the deviation from the planned production plan as much as possible with a small consideration to daily deadlines, delivery rewards and revenue losses. Constraint (11) limits overproduction and constraint

(12) limits over-delivery similar to constraints (2) and (3) from planning model. Constraint (13) establishes capacity for the constrained resources for the items that don't have setups longer than a period. Constraint (14) manages capacity jointly for the items that have setups longer than a period by employing cumulative capacity calculations. Constraint (15) calculates the total utilization of the machine j at execution period t, which adds all production, plus regular part setups, plus full capacity if a large part setup is completed within its setup time, plus the remaining setup if the large setup is completed that same period and enforces it to be less than or equal to daily capacity, which is C_{jt}. Constraint (16) ensures that two large setups (setups longer than a period) cannot be completed within the overlapping setup windows for two different items on the same resource. Constraint (17) is analogous to (16) but concerning setups for the same item. Constraint (18) ensures that a large setup and a small setup cannot be scheduled at the same time (it does allow for one to be completed in a given period and the other one started afterwards that period if capacity allows). Constraint (19) ensures that large setups can only be completed after the required time window. Constraint (20) ensures that an item can be produced only if setup for the item has been completed on that period or it is the first item continuing from previous period (its setup had already been done). Constraint (21) and (22) are inventory balance constraints similar to constraints (6) and (7) from the planning production model. Constraint (23) and (24) establish initial inventory and availability of items within the first couple of tactical periods, similar to constraints (8) and (9) from the planning production model. Constraint (25) ensures that if an item is the first item in period t+1, not needing a setup, it must be the last item in period t. Constraint (26) ensures that if an item is the first item in period t+1, it must have

either completed setup in period t or must have been produced in period t; that is, production can span several periods after a setup. Constraint (27) ensures that if an item is the first and last item in period t, that resource cannot be setup for any other part during that period (note that the length of a period is selected to be short enough that an additional setup for the same product would never be effective). Constraints (28) and (29) ensure that there can only be one first item and one last item, respectively, in a given period and constrained resource. Constraints (30) initialize the first item (current equipment setup) in the first period. Constraints (31)-(33) calculate the deviation from the production plan, assuming e execution periods per planning periods.

4.4. Industrial Test Application, Computational Methods And Results

4.4.1. Description of Industrial Test Application

The size of planning and execution problems are given in Tables 4.4.1 and 4.4.2 below:

Set	**Description**	**Size**	**Units**
O	Orders	450	Orders
I	Items	6795	Items
I^c	Items that require limited resources	1419	Items
J	Limited resources	15	Machines
K	Parent items	2391	Items
N (UPP)	Parent-child relationships (BOM)	14701	Tuples of items
T	Time periods	13	Weeks
Customers	Customers	44	Customers

Table 4.4.1 Planning Instance Set Sizes

Set	Description	Size	Units
O	Orders	296	Orders
I	Items	8744	Items
I^c	Items that require limited resources	3005	Items
J	Limited resources	30	Machines
K	Parent items	4648	Items
N (UPP)	Parent-child relationships (BOM)	15541	Tuples of items
T	Time periods	10	Days
Customers	Customers	39	Customers
Γ^L	Items that require setups that take longer than a day	43	Items
U^1 & U^2	First and second week production requirements from the planning model	1723 each, 3446 total	Items

Table 4.4.2 Execution Instance Set Sizes

A test application instance this size has never been attempted to be solved before in the MLCLSP literature. There are 450x6795x13 = 39,750,750 production variables (u) for the planning problem and 296x8744x10 = 25,882,240 production variables (u) for the execution problem. Observe that while the number of orders is reduced because some are not active in the first two weeks of the plan, the number of parts is significantly higher because each part may have been subdivided into subparts to adapt to the smaller timescale. Furthermore, there are 1419x15x13 = 276,705 binary setup (z) variables for the planning problem and 3005x30x10=901,500 binary setup (z) variables for the execution problem. Similar numbers can be calculated for the rest of the variables (e.g.

sequencing variables in the execution problem, which are also binary) which are equally high.

4.4.2. **Data Pre-Processing**

Please note that this sub-section and sub-section 4.4.4. are based on the work by Levy (2019). A series of pre-processing steps were implemented in order to organize the information according to the model. The steps shown below are applicable to the data structure for the planning model. Pre-processing for the execution model follows a similar method. Recall that the parts in the BOM that require multiple operations over several time periods need to be broken down into part-steps; that is part numbers are broken down into several part-step numbers (PSN's) to allow us to capture the use of capacitated resources each time period. This not only requires changes to the UPP file, but also to the demand file since the final demand is no longer for that part but for its last part-step, and to the initial inventory and current conditions, which again need to describe the processing step (or part-step) that the part is currently at.

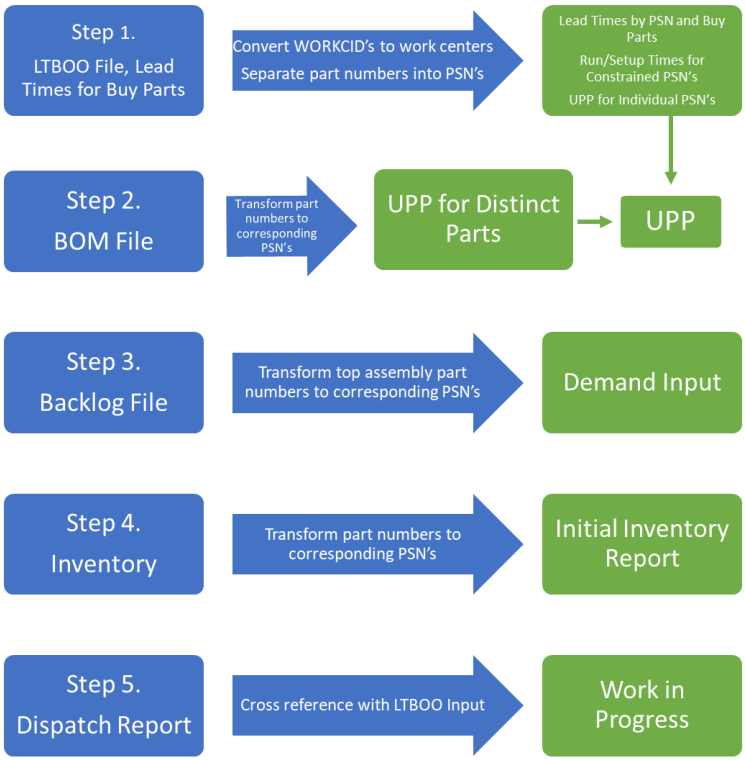


Figure 4.4.1 Data Pre-Processing Steps

As discussed above, the planning model's solutions are based on weekly time buckets. However, the company's routing process data was not formatted in any specific way and simply displayed the amount of time to complete each step of a part's routing process.

For example, let X be the name of a part; Table 4.4.3 shows the routing information of part X according to the company's Lead Time and Bill of Operations (LTBOO) file.

Step	Machine	Run Time	Setup Time	Move Time	Time (hrs)
1	Cutting of Material	0.25		16	16.25
2	Multi-Axis Turning – Small	5	0.25	16	21.25
3	Multi-Axis Turning – Small	5	0.25	16	21.25
4	Multi-Axis Turning – Small	5	0.25	16	21.25
5	Manual Turning	2	2	16	20

6	Deburr		0.16	16	16.16
7	Codemark	0.25	0.03	16	16.28
8	Inspection	0.25		16	16.25

Table 4.4.3 Routing Information for part X

In this example, it is important to note that only steps 2-7 and step 8 require capacitated resources. In other words, some steps along a part's routing process may not have measurable capacities or capacities that the company needs to consider. Meanwhile, in order to transform the previous data into weekly steps, an iterative algorithm was developed and executed in Excel. The table below shows the outcome of this process given the new (weekly) step numbers associated with each processing step on the first column.

Step	Machine	Run Time	Setup Time	Move Time	Time	Cumulative Time (hrs)
1	Cutting of Material	0.25		16	16.25	16.25
1	Multi-Axis Turning - Small	5	0.25	16	21.25	37.5
1	Multi-Axis Turning - Small	5	0.25	16	21.25	58.75
1	Multi-Axis Turning - Small	5	0.25	16	21.25	80
2	Manual Turning	2	2	16	20	20
2	Deburr		0.16	16	16.16	36.16
2	Codemark	0.25	0.03	16	16.28	52.44
2	Inspection	0.25		16	16.25	68.69

Table 4.4.4 Cumulative Calculation of Weekly Requirements

The algorithm takes the cumulative time of each step as long as the value is less than 80 hours, which corresponds to two 40 hour shifts per week. These steps are then grouped

into one step, and then the next iteration begins in the following row. While there are slight exceptions to this rule in order to account for steps without measurable capacities, one can see from the previous example that part X no longer has 8 steps, but rather 2 steps that each take one week. The model interprets each of these steps as a different part, so part X now involves X_1, X_2, etc. This iterative process was applied to each routing process, which was not only almost 57,000 rows of data, but now meant that all of the other data files that referenced part numbers would have to be adjusted according to their new step numbers. These processes are outlined in Figure 4.4.1 by the arrows in steps 2 through 5.

However, many challenges were encountered when trying to format the data as a result of certain gaps and misalignment between the various files. The list below outlines some of these challenges in greater detail.

- Missing data: Instances of missing data often took place in the Bill of Materials (BOM) file which provides the UPP parameter values. Some values were either not available or not whole numbers.

- Multiple variations of data: Instances of repeated variations of data similarly often took place in the BOM file. For example, while a child to parent relationship may appear more than once in the file if it is used in more than one top assembly, the issue arises when the UPP values at each occurrence are different.

- Disagreement between files: The misalignment between files was the most common issue. A popular example was between the BOM and LTBOO file,

where a part in the BOM file which was listed as a buy part also had a corresponding routing process in the LTBOO file.

- Outlier values: This problem was most popular in the LTBOO file, specifically among the data for run times, set up times, queue times, and wait times. Some parts had extremely large values for this data which did not align with the majority of other similar processes.

While these issues may seem negligible at a first glance, the software used to solve the models (AMPL) cannot read the data when it is not completely free of errors, since it is unable to make assumptions to overcome these errors. In order to utilize the data effectively, these errors had to be manually fixed upon discovery, though given that this process is both tedious and inefficient, the idea of an automated pre-processing tool was introduced. This tool also allows us to generate random instances using the company's data, which we will convert into a full computational study in the future. The tool was implemented in MATLAB and does the following data pre-processing steps:

1. Randomly selects n unique orders from backlog file, where n is decided by the user
2. Scans BOM for part number corresponding to first order
3. Selects all relevant UPP line items and populates information in a new small-scale BOM file
4. Generates corresponding UPP line items for part step numbers
5. Repeats steps 2-4 for each order

6. Sends reduced backlog and BOM to a consolidated input file in Excel

4.4.3. **Computational Methods and Results**

As can be seen from Section 4.4.1., real-life industrial-sized instances are behemoths and that is the reasoning behind most of the literature focusing on heuristic methods. However, one aspect of the data structure that we can take advantage of is the sparsity of the item-order and item-machine matrices. In our case, only one item is required by an order (however, parents of that item needed for that order still must be calculated through BOM explosion, which we defined as UPO) and an item requires only a small percentage of the machines. Therefore, we reduce the size of the problem significantly by only defining the production/inventory/setup variables for the item-order and item-machine combinations that exist. We define these combinations as sub-sets in our mathematical model (Γ for item-machine combinations, Θ for item-order combinations) and define the variables through these sub-sets.

To give an example of the efficiency of this linking, 25,882,240 production variables in the execution problem reduce only to 3446 because we only define production variables in the execution problem for the production of the items that are directed by the planning model (U^1 and U^2) because the main objective of the execution model is to meet the production demand calculated from the planning model.

Other than reducing the problem size through linking as described above, our computational approach is to code the mathematical models described above using AMPL IDE commercial optimization software and solving them using Gurobi 7.0 (for planning model) and CPLEX 12.7 (for execution model) on a PC with Intel Core™ i7-

6700 CPU @3.40 GHz and 32 GB RAM. We limited the computation time to 6 hours for planning model and 4 hours for execution model.

Planning model is solved within the time limit with an optimality gap of 0.0464% and execution model is solved by CPLEX in 3.66 seconds with an optimality gap of 2.82%, which makes us optimistic about the implementation of our solution at the manufacturer's site. Time limits are well within the acceptable window: planning horizon is 3 months and execution horizon is 2 weeks, so a couple of hours to solve the planning problem and a couple of seconds to solve the execution problem is matching well with the needs of the industry.

We will now continue with sensitivity analysis on sub-section 4.4.4. Please note that the sub-section 4.4.4. is based on the work by Levy (2019).

4.4.4. **Sensitivity Analysis of Order Linking, Objective Functions and KPIs**

Due to large computation time still required for the planning model, we also developed a simpler version of our model without linking orders to items (by removing the order set from the model), which we call the unpegged model (our current model therefore becomes the pegged model). Given that the unpegged model has significantly less variables, it takes much less time to solve, which makes it the favorable framework for testing and validation. However, an output from the unpegged version is not particularly useful to a company that is seeking to apply more transparency along the production floor and monitor progress of specific customer orders, since it does not connect parts to their corresponding orders. Although the pegged model requires a much larger computation time, it is the ideal framework to use during implementation for its traceability feature.

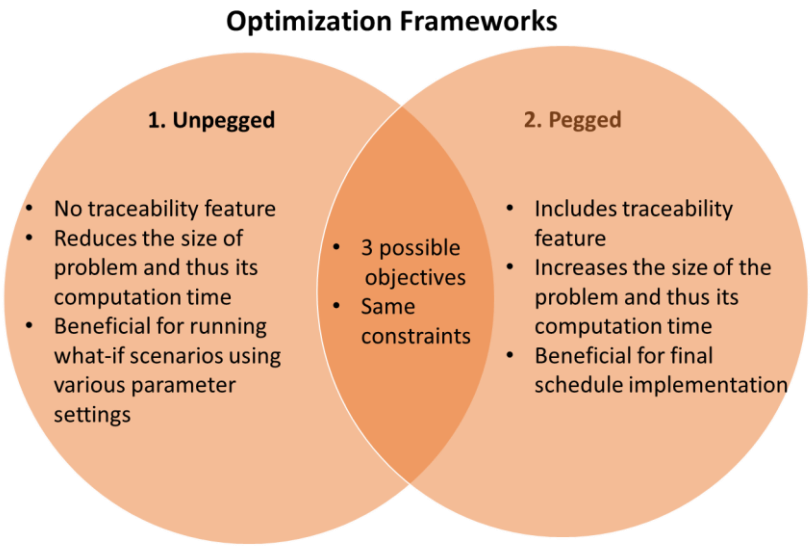


Figure 4.4.2 Comparison of Unpegged vs. Pegged Model

Note that although the unpegged model is better for validation purposes, the pegged model also needs to be validated to prove that it is working as intended. This was achieved through small examples. Due to reduced complexity, small examples are expected to have the same results both for pegged and unpegged model and we were able to receive the exact same results and therefore conclude that our models are validated. For larger, more complex examples, the results are expected to differ since the models consider different complexity levels, as will be seen on our sensitivity analysis below. The differences between the models can also be explained by the optimality gaps. The pegged model has an optimality gap, even though it is very small, whereas the unpegged model is solved to optimality.

Due to the tactical nature of the planning model, we also developed different objective functions based on the preferences of the aerospace parts manufacturer and ran a

sensitivity analysis using KPIs. Below are the three objective functions we have analyzed, including the penalty function described above:

4.4.4.1. **Delivery Model**

The main performance metric that is prioritized in the delivery model is on-time deliveries. Below is the objective function, which maximizes on-time deliveries while penalizing both incomplete and late orders, rewards early production and every step of production towards completing an order, and incentivizes the use of inventory.

$$\begin{aligned} Maximize\ Delivery\colon & \left(\sum_{o:\, D_o \le p} M \sum_{t: t \le D_o} w_{ot}\right) \\ & - \left[\left(\sum_{o:\, D_o \le weeks} \left(P_o(p+1-D_o)\right) \times \left(Q_o - \sum_t w_{ot}\right)\right)\right. \\ & + \left(\sum_{o,t: t > D_o} P_o(t-D_o) w_{o,t}\right) - \left(\sum_{o,t: t < D_o} \varepsilon_1 (D_o - t) w_{o,t}\right) \\ & \left. + \left(\sum_{i,o} H_i \left(N_i^o - \sum_t a_{i,o,t} + u_{i,o,t}\right)\right) + \left(\sum_{i,o,t} \varepsilon_2 (u_{i,o,t})\right)\right] \end{aligned}$$

The first component of the objective function rewards each quantity of an order that is produced before its due date. In order to provide the greatest reward for completion and ensure that this key performance indicator (KPI) is prioritized over anything else, coefficient *M* is typically assigned a very large number, such as 100,000. Meanwhile, ε_1 and ε_2 are typically assigned much smaller values, such as 0.01 and 0.001 to slightly highlight the rewards and penalties previously mentioned.

4.4.4.2. **Revenue Model**

The main KPI that is prioritized in the revenue model is revenue. Below is the objective function, which is the same as the delivery model except for the features of the first component.

$$Maximize\ Revenue: \left(\sum_{o,t} M(Rev_o)(w_{o,t})\right) - \left[\left(\sum_{o:\, D_o \le p} \left(P_o(p+1-D_o)\right) \times \left(Q_o - \sum_t w_{ot}\right)\right) + \left(\sum_{o,t:t>D_o} P_o(t-D_o)w_{o,t}\right) - \left(\sum_{o,t:t<D_o} \varepsilon_1(D_o-t)w_{o,t}\right) + \left(\sum_{i,o} H_i\left(UPO_{i,o} - \sum_t a_{i,o,t} + u_{i,o,t}\right)\right) + \left(\sum_{i,o,t} \varepsilon_2(u_{i,o,t})\right)\right]$$

The function prioritizes the total revenue associated with order completions, which is simply the product of the quantities completed by the revenue associated with each unit in that order. Just as in the delivery model, this KPI is given a high weight by applying a large value to the coefficient M that multiplies the total revenue. As a result, orders with high returns in revenue are pushed through production by the program.

4.4.4.3. **Customer Priority Model**

The last type of model (also used above in our modeling section) allows a company to prioritize a set of orders, typically by customer. Once again, the objective function is displayed below, which has only one different component from the previous two functions.

$$Minimize\ Penalty: \left(\sum_{o:\, D_o \le p}\left(P_o(p+1-D_o)\right) \times \left(Q_o - \sum_t w_{ot}\right)\right) + \left(\sum_{o,t:t>D_o} P_o(t-D_o)w_{o,t}\right) - \left(\sum_{o,t:P_o=M,t<D_o} 0.5 * P_o(D_o-t)w_{o,t}\right) - \left(\sum_{o,t:t<D_o} \varepsilon_1(D_o-t)w_{o,t}\right) + \left(\sum_{i,o} H_i\left(N_i^o - \sum_t a_{i,o,t} + u_{i,o,t}\right)\right) + \left(\sum_{i,o,t} \varepsilon_2(u_{i,o,t})\right)$$

By applying an appropriately large delay penalty parameter, Po, for the set of order of interest, one can easily prioritize completions of those orders. The goal of the objective function is to minimize a penalty which decreases as more and more parts of highly prioritized orders get completed. In addition, the third component of the function rewards early completions of the prioritized orders. The section states that for all orders and time periods such that the order has a specific penalty, M, and the time period is before the due date of the specific order, the penalty decreases by the following factor:

$$0.5 * P_o(D_o - t)w_{o,t}$$

This factor incentivizes early production of the specified orders. Meanwhile, the overall objective also penalizes both incomplete and late orders, rewards early production and every step of production towards completing an order, and incentivizes the use of inventory.

4.4.4.4. **KPI Descriptions**

The table below describes the KPI's used to evaluate the different frameworks and models. Since all tests were run over a 13-week time horizon, each on-time delivery (OTD) performance benchmark was also broken up by month. It is also important to note that the KPI's corresponding to orders represent full order completions, whereas those corresponding to units represent top assembly completions that together make up one order.

KPI	**Description**
OTD_Orders	overall percentage of orders completed on-time
OTD_Orders_Month1	percentage of orders due during first month completed on-time

OTD_Orders_Month2	percentage of orders due during second month completed on-time
OTD_Orders_Month3	percentage of orders due during third month completed on-time
OTD_Units	overall percentage of order units completed on time
Average_OTD_Units	average percentage of order units completed on time
OTD_Units_Month1	percentage of order units due during first month completed on-time
OTD_Units_Month2	percentage of order units due during second month completed on-time
OTD_Units_Month3	percentage of order units due during third month completed on-time
Overdue	total number of incomplete order units
Overdue_Month1	number of incomplete order units during first month
Overdue_Month2	number of incomplete order units during second month
Overdue_Month3	number of incomplete order units during third month
PenaltyIndex	objective function penalty
Revenue	total revenue from completed order units
CustomerOTD_Orders	percentage of orders completed on time per customer
CustomerOTD_Units	percentage of order units completed on time per customer
CustomerProportionSatisfied_Orders	percentage of orders completed per customer
CustomerProportionSatisfied_Units	percentage of order units completed per customer

Table 4.4.5 KPIs

4.4.4.5. **Sensitivity Analysis of Pegged vs. Unpegged Model Performances**

The table below compares the results from the pegged and unpegged frameworks across multiple KPI's and uses colored directional arrows to indicate relative performance.

Framework Analysis			
KPI	**Pegged**	**Unpegged**	**% Difference**
Average_OTD_Units			
Customer Priority	↑	↓	13.66%
On-Time Delivery	↑	↓	6.23%
OTD_Orders			
Customer Priority	↑	↓	13.64%
On-Time Delivery	↑	↓	5.13%
OTD_Orders_Month1			
Customer Priority	→	→	0.00%
On-Time Delivery	→	→	0.00%
OTD_Orders_Month2			
Customer Priority	↑	↓	30.00%
On-Time Delivery	↑	↓	14.81%
OTD_Orders_Month3			
Customer Priority	→	→	0.00%
On-Time Delivery	↓	↑	4.08%
OTD_Units_Month1			
Customer Priority	↑	↓	58.82%
On-Time Delivery	→	→	0.00%
OTD_Units_Month2			
Customer Priority	↑	↓	3.27%
On-Time Delivery	↑	↓	2.86%
OTD_Units_Month3			
Customer Priority	↓	↑	2.45%
On-Time Delivery	↓	↑	1.46%
PenaltyIndex			
Customer Priority	↓	↑	5.30%
On-Time Delivery	↑	↓	0.33%
Revenue			
Customer Priority	↑	↓	0.04%
On-Time Delivery	↓	↑	0.39%

Table 4.4.6 Comparison of Pegged vs. Unpegged Model Performances

The KPI's are also broken down by model; in this case, the delivery and customer priority models are displayed. In terms of on-time delivery of both top assemblies and full orders, the pegged framework produces slightly better results. The pegged framework averages almost 10% higher in the average number of units that are delivered on-time

across all orders, as well as in the total number of orders delivered on time. While there appears to be more drastic differences in performance at the monthly level, it is important to note that the actual values are quite close. For example, the percent difference between OTD_Orders_Month2 for customer priority is 30%; meanwhile, the KPI for the pegged framework is 51% and the KPI for the unpegged framework is 37%. The percent difference between OTD_Units_Month1 for the customer priority model also appears very large, at 58.8%, but simply considering this information is misconstruing. In reality, the KPI for the pegged framework is 2% and the KPI for the unpegged framework is 1.12%. Lastly, the penalty indices only differ on average by 2.8% and the revenue values by 0.4%, which suggests that both frameworks are very comparable in terms of optimality.

The largest relative differences between the two models are on KPIs for customer priority as explained above, mostly favored towards the pegged model. That makes intuitive sense, because the model that can differentiate between different customer orders (pegged model) will be able to satisfy customer priorities better.

The greatest difference between the two frameworks is actual computation time. Given the computing parameters previously outlined at the end of sub-section 4.4.2., Data Preprocessing, while the unpegged model typically took less than 30 seconds to solve, the pegged model often took up to 12 hours. Though this may be manageable for a company that only plans to run the tool every few weeks, the pegged framework is not an ideal version for testing and validation. The pegged framework certainly provides a more beneficial output, whereas the unpegged framework is very useful for running many

different tests to obtain fast results. Meanwhile, it is recommended that both frameworks be adjusted simultaneously.

4.4.4.6. **Sensitivity Analysis of Different Objective Functions**

The table below captures the results from running the three models across the unpegged framework.

KPI	Customer Priority	On-Time Delivery	Revenue
OTD_Orders	25.63%	35.63%	26.25%
OTD_Orders_Month1	3.41%	6.82%	4.55%
OTD_Orders_Month2	37.78%	55.56%	37.78%
OTD_Orders_Month3	80.77%	96.15%	80.77%
Average_OTD_Units	27.09%	36.46%	27.71%
OTD_Units	51.12%	56.25%	52.32%
OTD_Units_Month1	1.12%	7.66%	4.30%
OTD_Units_Month2	72.27%	76.34%	72.27%
OTD_Units_Month3	95.39%	99.71%	95.39%
Overdue	291	362	319
Overdue_Month1	1515	1509	1512
Overdue_Month2	1084	1053	1067
Overdue_Month3	474	455	471
PenaltyIndex	5327327.692	530952.8386	532963.562
Revenue	59604.78007	60078.62334	60263.8587

Table 4.4.7 Comparison of Different Objective Functions

The on-time delivery model excels relative to the other models at completing orders on-time. However, it is important to recognize that the input data has clearly skewed the results, since many orders appear to be due before t=0 or before their estimated lead

times, so it is critical to understand the type of data used to test the model in order to evaluate the results appropriately.

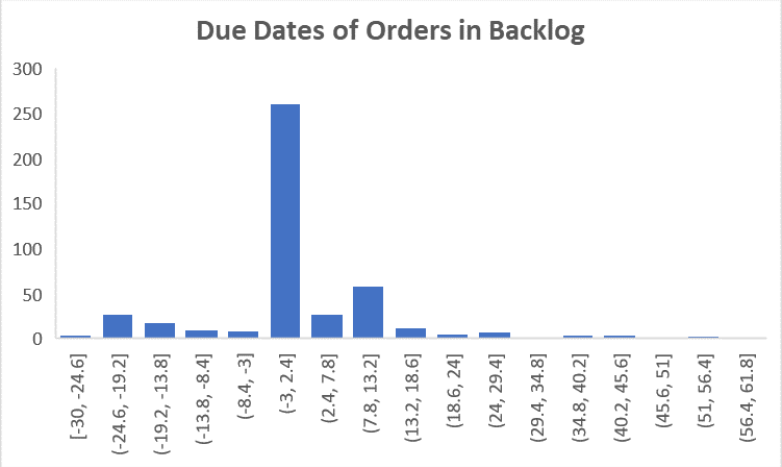


Figure 4.4.3 Due Dates of Orders in Backlog

Figure 4.4.3 shows that not only do more than 250 of the orders considered in the company's backlog have due dates between -3 and 2.4 weeks, but there are even a few orders that appear to have been due up to 30 weeks before the models were even tested. Therefore, if the tool is being given orders that are due in unfeasible time frames, it is wrong to expect that the tool's performance during the first few weeks is going to provide acceptable results.

Fortunately, as the months progress across each KPI category, the values also improve, which suggests that true performance improvement can be seen once the tool has been up and running. Especially during month 2, the discrepancy between OTD_Orders and OTD_units shows that even if orders aren't being fully completed, all models are pushing through production due to a feature across all objective functions that rewards every step towards completing an order.

While the revenue model generates the most revenue from all completed units, it is important to note that the value is only greater than the lowest revenue-generating model by 1%.

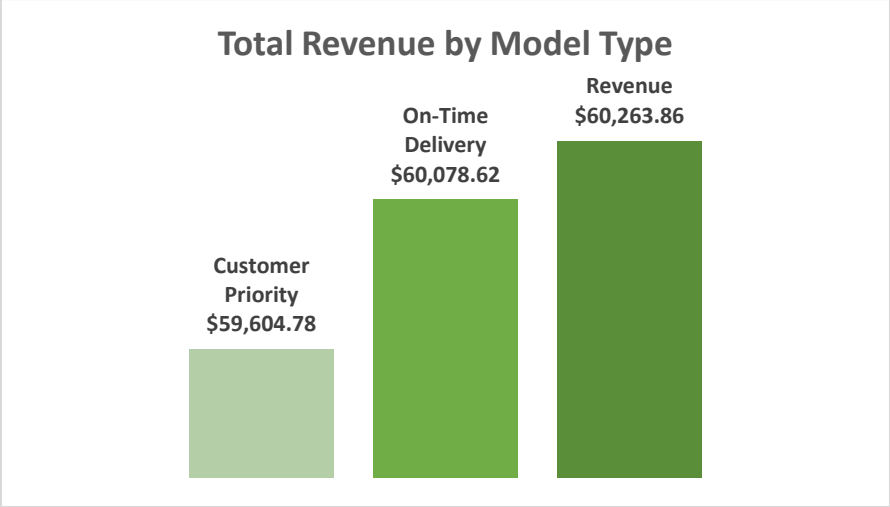


Figure 4.4.4 Total Revenue by Model Type

By increasing the value of M in the objective, a user can increase the incentive for completing highly profitable orders.

The only difference when testing the customer priority model was that customer K2 was given a much larger priority. There appeared to be no significant differences in performance by customer across the three models for 34 out of the 42 customers observed. However, for the remaining 8 customers, the impact of the customer prioritization model is illustrated in Table 4.4.8, which shows the percentage of units satisfied pertaining to specific customers.

Customer	Customer Priority	On-Time Delivery	Revenue
K2	99.85%	81.16%	85.15%

P2	94.54%	97.27%	97.27%
E1	72.22%	100.00%	100.00%
D2	72.22%	75.00%	68.75%
K1	71.53%	86.86%	86.86%
W1	41.18%	64.71%	70.59%
A1	29.24%	24.40%	38.35%
C2	27.78%	11.11%	33.33%

Table 4.4.8 Customer Performance Comparison

A color scheme is applied to each row to highlight the performance across each model. As shown by the table, almost 100 percent of customer K2's units are completed using the customer priority model, whereas only between 81 and 85 percent of the customer's units are typically completed using the default data. Meanwhile, the seven other customers listed must experience slightly less satisfaction as illustrated by the red and yellow cells in the first column. Thus, it is important that companies weigh the impact that the prioritization model could potentially have on the rest of their customers.

4.5. **Conclusion**

In this essay, we have introduced the Hierarchical Job Shop Schedule Planning and Execution Problem (HJSP) which consists of solving two Multi-Level Capacitated Lot-Sizing Problems (MLCLSPs) sequentially at a tactical and then an operational level. HJSP is inspired by a problem observed at an aerospace parts manufacturer. The operational level problem, which we call the execution problem allows continuous setups and setups that take longer than a day. Both problems allow positive lead-times, back-orders and loss sales. Due dates are also considered. We solve a real-life industrial-sized test application using an instance provided by the aerospace manufacturer. We exploit

structure of the problem to reduce the size of the problem and implement our models on a commercial optimization software that solves our problem to near-optimality. A data pre-processing tool on MATLAB is also generated. Finally, we also run a sensitivity analysis of the planning model by comparing a simplified version (unpegged) against the full version (pegged) and different objective functions using KPIs.

CHAPTER 5

EXTENSIONS / FUTURE RESEARCH DIRECTIONS

My advisors Ana Muriel and Hari Balasubramanian have collaborated with me and contributed to the work described in this chapter.

5.1. Extensions of Chapter 2

There are many different future research directions that we propose to take to build upon our work in Chapter 2. To name a few:

- We plan to further investigate and strengthen our findings using more random computational instances of different sizes and allowing for longer solution times for the problems that have already been investigated (Parallel Machine and Flexible Flow Shop Scheduling).
- We also plan to further investigate the relationship between modeling approaches and computational complexity in machine scheduling problems by modeling different problems, such as job shop scheduling and open shop scheduling problems, using direct and relative positional variables.
- We plan to compare and contrast the exact solution methodologies proposed in this study and heuristic/meta-heuristic solution methodologies commonly used in the machine scheduling literature, such as Genetic Algorithms, Lagrangian Relaxation, etc.
- Finally, we plan to find more real-life examples of machine scheduling problems and help the decision makers using our current models or a custom modeling approach/decision tools similar to what we did in this study for Artaic.

5.2. **Extensions of Chapter 3**

5.2.1. **Flexibility in Second Stage (Provider)**

Note that one of the assumptions in team primary care practice model was that the providers were dedicated to patients, i.e. the patient could not see any provider they wanted, they had to go to their dedicated provider. Even though this assumption is important for continuity of care in primary care practices, there are advantages to having flexible providers as well, the main one being the improvements on patient wait time and provider idle time. As we have seen on Chapter 3, flexibility in the first stage (nurse) had significant benefits, especially when the service time variability increases. Therefore, there is a potential benefit to flexibility in the second stage (provider) as well. We plan to investigate this potential using similar experiments to the ones in Chapter 3. We already developed models with and without crossovers, as given below:

5.2.1.1. **Flexible Nurses and Providers with Crossover**

$y_{i,s}^{start}$ Start time of patient *i* with nurse under scenario *s*

$y_{i,s}^{finish}$ Finish time of patient *i* with nurse under scenario *s*

$z_{i,s}^{start}$ Start time of the i^{th} patient to visit with a provider under scenario *s*

$z_{i,s}^{finish}$ Finish time of the i^{th} patient to visit with a provider under scenario *s*

N_{is}^{max} Maximum of the finish times of patients 1,…, *i-1* with nurses under scenario *s*

$P_{i,s}^{max}$ Maximum of the finish times of the first i-1 patients to be seen by the

providers under scenario *s*

X_i Appointment slot assigned to patient *i*, an integer variable in {0,1,2,...}.

Observe that $z_{I,s}^{finish}$ *and* $P_{I,s}^{max}$ represent the times at which the last two patients will finish with the providers, and thus the completion time of the two providers. The second largest logic tells us what is the time at which a nurse is available for patient i, and at the same time it represents the time at which the i-1th patient has finished with the nurses and is available to be seen by a provider. Then once we have patients in the order at which they finish with the nurses, we apply the second largest logic again at the providers step.

$$Minimize$$

$$\frac{1}{S}\left(\alpha\left[\sum_{s}\left(z_{I,s}^{finish}+P_{I,s}^{max}-\sum_{i=1}^{I}\tau_{i,s}^{P}\right)\right]+\beta\left[\sum_{s}\sum_{i=1}^{n}\left(y_{i,s}^{start}-15X_i\right)+\sum_{s}\left(\sum_{i=1}^{I}z_{i,s}^{start}-\sum_{i=1}^{I}y_{i,s}^{finish}\right)\right]\right)$$

$$subject\ to$$

$$y_{1,s}^{start}=0\ \forall s\in S$$

$$y_{2,s}^{start}=0\ \forall s\in S$$

$$z_{0,k,s}^{finish}=0\ \forall k\in K,\forall s\in S$$

$$X_1=0$$

$$X_2=0$$

$$y_{3,s}^{start}\geq\min\left(y_{1,s}^{finish},y_{2,s}^{finish}\right)\ \forall s\in S$$

$$N_{3,s}^{\max}\geq\max\left(y_{1,s}^{finish},y_{2,s}^{finish}\right)\forall s\in S$$

$$N_{i,s}^{\max}\geq\max\left(N_{i-1,s}^{max},y_{i-1,s}^{finish}\right)\forall i\in 4..I\ \forall s\in S$$

$y_{i,s}^{start} \geq \min\left(N_{i-1,s}^{max}, y_{i-1,s}^{finish}\right)\ \forall i \in 4..I\ \forall s \in S$

$y_{i,s}^{finish} = y_{i,s}^{\text{start}} + \tau_{\text{i,s}}^{\text{N}}\ \forall \text{i}\ \forall \text{s}$

$z_{1,s}^{start} \geq \min(y_{1,s}^{finish}, y_{2,s}^{finish}) \qquad \forall s \in S$

$z_{i,s}^{start} \geq \min\left(N_{i+1,s}^{max}, y_{i+1,s}^{finish}\right)\ \forall j \in \{2..I\}, s \in S$

$z_{j,s}^{finish} = z_{j,s}^{start} + \tau_{j,s}^{p} \qquad \forall j \in J_k, s \in S$

$z_{3,s}^{start} \geq \min\left(z_{1,s}^{finish}, z_{2,s}^{finish}\right)\ \forall s \in S$

$P_{3,s}^{\max} \geq \max\left(z_{1,s}^{finish}, z_{2,s}^{finish}\right) \forall s \in S$

$P_{i,s}^{\max} \geq \max\left(P_{i-1,s}^{max}, z_{i-1,s}^{finish}\right) \forall i \in 4..I\ \forall s \in S$

$z_{i,s}^{start} \geq \min\left(P_{i-1,s}^{max}, z_{i-1,s}^{finish}\right)\ \forall i \in 4..I\ \forall s \in S$

$X \geq 0, INT;\ y^{start}, y^{finish}, z^{start}, z^{finish} \geq 0$

Observe that the number of binary variables has not increased!

If we don't allow for crossover then the patients will always be ordered in the sequence of the original practice schedule and the problem can be formulated as:

$$Minimize$$

$$\frac{1}{S}\left(\alpha\left[\sum_{s}\left(z_{I,s}^{finish} + P_{I,s}^{max} - \sum_{i=1}^{I}\tau_{i,s}^{P}\right)\right] + \beta\left[\sum_{s}\sum_{i=1}^{n}\left(y_{i,s}^{start} - 15X_i\right) + \sum_{s}\left(\sum_{i=1}^{I} z_{i,s}^{start} - \sum_{i=1}^{I} y_{i,s}^{finish}\right)\right]\right)$$

$$subject\ to$$

$y_{1,s}^{start} = 0\ \forall s \in S$

$y_{2,s}^{start} = 0\ \forall s \in S$

$z_{0,k,s}^{finish} = 0\ \forall k \in K, \forall s \in S$

$X_1 = 0$

$X_2 = 0$

$$y_{3,s}^{start} \geq \min\left(y_{1,s}^{finish}, y_{2,s}^{finish}\right) \ \forall s \in S$$
$$N_{3,s}^{\max} \geq \max\left(y_{1,s}^{finish}, y_{2,s}^{finish}\right) \forall s \in S$$
$$N_{i,s}^{\max} \geq \max\left(N_{i-1,s}^{max}, y_{i-1,s}^{finish}\right) \forall i \in 4..I \ \forall s \in S$$
$$y_{i,s}^{start} \geq \min\left(N_{i-1,s}^{max}, y_{i-1,s}^{finish}\right) \ \forall i \in 4..I \ \forall s \in S$$
$$y_{i,s}^{finish} = y_{i,s}^{\text{start}} + \tau_{\text{i,s}}^{\text{N}} \ \forall \text{i} \ \forall \text{s}$$
$$z_{i,s}^{start} \geq y_{i,s}^{finish} \ \forall i \in I, s \in S$$
$$z_{j,s}^{finish} = z_{j,s}^{start} + \tau_{j,s}^{p} \qquad \forall j \in J_k, s \in S$$
$$z_{3,s}^{start} \geq \min\left(z_{1,s}^{finish}, z_{2,s}^{finish}\right) \ \forall s \in S$$
$$P_{3,s}^{\max} \geq \max\left(z_{1,s}^{finish}, z_{2,s}^{finish}\right) \forall s \in S$$
$$P_{i,s}^{\max} \geq \max\left(P_{i-1,s}^{max}, z_{i-1,s}^{finish}\right) \forall i \in 4..I \ \forall s \in S$$
$$z_{i,s}^{start} \geq \min\left(P_{i-1,s}^{max}, z_{i-1,s}^{finish}\right) \ \forall i \in 4..I \ \forall s \in S$$
$$X \geq 0, INT; \ y^{start}, y^{finish}, z^{start}, z^{finish} \geq 0$$

Even though the early tests show high optimality gaps for the crossover model, the results are promising. First of all, the big-M parameters can be tightened similar to dedicated provider model. The nurse stage hasn't changed, so the only big-M parameter that must be improved is on the provider stage. Below is a method for doing so, similar to the proofs in the Appendix:

Proof of M1 and stage-based lower bounds are the same as dedicated provider model. For the provider stage-based lower bound, we pick the minimum of the two simulated provider lower bounds from the previous model as the general lower bound for the provider.

5.2.1.2. **Proof of M2**

For constraints $z_{i,s}^{start} \geq \min\left(P_{i-1,s}^{max}, z_{i-1,s}^{finish}\right) \ \forall i \in 4..I \ \forall s \in S$ to be valid, we must ensure that

$$M2_{i,s} \geq \left(P_{i-1,s}^{max} - z_{i-1,s}^{finish}\right)^{+} \qquad \forall i \in 4..I, s \in S$$

where $P_{i-1,s}^{max}$ is the maximum of the finish times of patients 1 through i-2 with a provider for that scenario. That is, M2 must be an upper bound on the difference in finish times with provider of the two patients that are seen by a provider at the time patient i starts service, and it can vary for each patient in the sequence and from scenario to scenario. We consider two cases: In Case1, the finish time of patient *i*-1 with provider is greater than or equal to the maximum of the finish times of patient from 1 to *i*-2 with providers; and in Case2, it is strictly lower.

<u>*Case 1*</u>: $P_{i-1,s}^{max} \leq z_{i-1,s}^{finish}$, the difference $z_{i-1,s}^{finish}$ - $P_{i-1,s}^{max}$ needs to be bound:

In this case, observe that

a. $z_{i-1,s}^{finish} = z_{i-1,s}^{start} + \tau_{i-1,s}^{P}$ and by definition, $z_{i-1,s}^{start} \geq N_{i,s}^{\max}$

b. By definition: $P_{i-1,s}^{max} \geq z_{i-2,s}^{finish} = z_{i-2,s}^{start} + \tau_{i-2,s}^{P}$, and thus $P_{i-1,s}^{max} - \tau_{i-2,s}^{P} \geq z_{i-2,s}^{start}$. Again, by definition, $z_{i-2,s}^{start} \geq N_{i-1,s}^{max}$.

c. Combining the two, the difference in finish times with provider is $z_{i-1,s}^{finish}$ - $P_{i-1,s}^{max}$ $= z_{i-1,s}^{start} + \tau_{i-1,s}^{P} - P_{i-1,s}^{max} \geq N_{i,s}^{\max} + \tau_{i-1,s}^{P} - P_{i-1,s}^{max} \geq N_{i,s}^{\max} + \tau_{i-1,s}^{P} - z_{i-2,s}^{finish} = N_{i,s}^{\max} + \tau_{i-1,s}^{P} - z_{i-2,s}^{start} - \tau_{i-2,s}^{P} \geq N_{i,s}^{\max} + \tau_{i-1,s}^{P} - N_{i-1,s}^{max} - \tau_{i-2,s}^{P}$

d. The difference in the maximum of nurse finish time of patients 1..i-1 ($N_{i,s}^{\max}$) and 1..i-2 ($N_{i-1,s}^{\max}$) is $N_{i,s}^{\max} - N_{i-1,s}^{\max}$. Patients i-1 and i-2 can be at most 30 minutes apart in appointment and have nurse processing times $\tau_{i-1,s}^{N}$ and $\tau_{i-2,s}^{N}$, so the maximum $N_{i,s}^{\max} - N_{i-1,s}^{\max}$ can be is 30+$\tau_{i-1,s}^{N} - \tau_{i-2,s}^{N}$

Thus, the difference $z_{i-1,s}^{finish} - P_{i-1,s}^{max}$ is bound by $\tau_{i-1,s}^{P} + Max\{0,30 + \tau_{i-1,s}^{N} - \tau_{i-2,s}^{N} - \tau_{i-2,s}^{P}\}$.

<u>*Case 2*</u>: $P_{i-1,s}^{max} > z_{i-1,s}^{finish}$, the difference $P_{i-1,s}^{max} - z_{i-1,s}^{finish}$ needs to be bound

In this case, observe that while patient *i*-1 has finished with one provider, say provider1 w.l.o.g., the other provider, provider2, is still busy with an earlier patient. The difference between the two can be calculated depending on which patient is still with provider2. If patient *r* is still with provider2, it means that patients *r*+1, *r*+2, …, through *i*-1 are seen by provider1, we have that:

a. $P_{i-1,s}^{max} = z_{r,s}^{start} + \tau_{r,s}^{P}$

b. $z_{i-1,s}^{finish} \geq z_{r+1,s}^{start} + \tau_{r+1,s}^{P} + \tau_{r+2,s}^{P} + \cdots + \tau_{i-1,s}^{P}$

c. $z_{r,s}^{start} \leq z_{r+1,s}^{start}$ since patients are seen by the provider in the order of their appointment times, $X_1 \leq X_2 \leq \cdots \leq X_I$.

d. Thus, the difference $P_{i-1,s-}^{max} - z_{i-1,s}^{finish} \leq \tau_{r,s}^{P} - \sum_{u=r+1}^{i-1} \tau_{u,s}^{P}$

$\underset{r=1,\ldots,i-2}{Max} \left\{\tau_{r,s}^{P} - \sum_{u=r+1}^{i-1} \tau_{u,s}^{P}\right\}$ will provide the tight bound.

The overall bound on the difference for both cases then is

$$Max\{Case1, Case\ 2\} = Max\left\{\tau_{i-1,s}^{P} + Max\{0,30 + \tau_{i-1,s}^{N} - \tau_{i-2,s}^{N} - \tau_{i-2,s}^{P}\}, \underset{r=1,\ldots,i-2}{Max} \left\{\tau_{r,s}^{P} - \sum_{u=r+1}^{i-1} \tau_{u,s}^{P}\right\}\right\}$$

For future research directions, we propose a similar approach to what has been accomplished in Chapter 3: generate random instances with different sizes and

distributions, solve them and improve upon optimality gaps using improved big-M parameters and the methods from Chapter 3 (tightening constraints, stage-based lower bounds, lower-bounding scheme using separate scenario groups, etc.), come up with scheduling guidelines, run sensitivity analyses and analyze the effect of different problem characteristics, mainly focusing on the difference between dedicated providers and flexible providers.

5.2.2. **Relaxation of Homogeneous Patient Assumption**

One assumption that we had on Chapter 3 was that the patient set was homogeneous, i.e. their service time came from the same distribution. Relaxing this assumption would lead to having different types of patients similar to Oh et al. (2013). The type of patient that we analyzed on Chapter 3 was High Complexity (HC). However, Oh et al. (2013) also introduces Low Complexity (LC) and Same Day (SD) patients. The research problem for team primary care practice becomes how to combine the model from Oh et al. (2013) with the model from Chapter 3 and have a model that can handle different types of patients with multiple nurses and providers. This would introduce sequencing different types patients in addition to scheduling them to time slots. The main challenge would then be to handle crossovers while also tracking the types of patients. This is doable if you only allow one crossover per patient. Below is a mathematical model that we have developed for two nurses and two providers for this purpose:

5.2.2.1. **Sets**

- I: Set of patients that must be scheduled over the time horizon.
- K: Set of providers.

- J_k: Set of patients that will visit provider k, indexed consecutively.
- I_k: Set of patients that will visit provider k, using their original indexes, e.g. patient 1,3,5 will visit provider 1, while patient 2,4,6 will visit provider 2. W.l.o.g. it can be assumed that odd numbered patients will visit provider 1 while even numbered patients will visit provider 2.
- S: Set of scenarios.

5.2.2.2. **Parameters**

- α: Weight of provider idle time in objective function.
- β: Weight of patient wait time in objective function.
- H_{HC}^{k}: Number of high complexity patients of provider k.
- H_{LC}^{k}: Number of low complexity patients of provider k.
- H_{SD}^{k}: Number of same day patients of provider k.
- $\tau_{i,s}^{N,HC}$: Nurse time of patient i under scenario s if patient is type HC.
- $\tau_{i,s}^{N,LC}$: Nurse time of patient i under scenario s if patient is type LC.
- $\tau_{i,s}^{N,SD}$: Nurse time of patient i under scenario s if patient is type SD.
- $\tau_{j,s}^{P_k,HC}$: Provider time of j^{th} patient of provider k under scenario s if patient is type HC.
- $\tau_{j,s}^{P_k,LC}$: Provider time of j^{th} patient of provider k under scenario s if patient is type LC.

- $\tau_{j,s}^{P_k,SD}$: Provider time of j^{th} patient of provider k under scenario s if patient is type SD.

- $f[j,k]$: Patient index of j^{th} patient of provider k in the overall set of patients in the practice.

- M: A very large number.

5.2.2.3. **Variables**

- X_i: Appointment slot of patient i, an integer between 0 and 15.

- $A_i = \begin{cases} 1 \; if \; patient \; i \; is \; assigned \; as \; type \; HC \\ 0 \; otherwise \end{cases}$

- $B_i = \begin{cases} 1 \; if \; patient \; i \; is \; assigned \; as \; type \; LC \\ 0 \; otherwise \end{cases}$

- $C_i = \begin{cases} 1 \; if \; patient \; i \; is \; assigned \; as \; type \; SD \\ 0 \; otherwise \end{cases}$

- $D_j^k = \begin{cases} 1 \; if \; j^{th} \; patient \; of \; provider \; k \; is \; originally \; (before \; crossovers) \\ type \; HC \\ 0 \; otherwise \end{cases}$

- $E_j^k = \begin{cases} 1 \; if \; j^{th} \; patient \; of \; provider \; k \; is \; originally \; (before \; crossovers) \\ type \; LC \\ 0 \; otherwise \end{cases}$

- $F_j^k = \begin{cases} 1 \; if \; j^{th} \; patient \; of \; provider \; k \; is \; originally \; (before \; crossovers) \\ type \; SD \\ 0 \; otherwise \end{cases}$

- $Q_{j,s}^k = \begin{cases} 1 \; if \; j^{th} \; patient \; to \; visit \; provider \; k \; (after \; crossovers) \\ type \; HC \; under \; scenario \; s \\ 0 \; otherwise \end{cases}$

- $W_{j,s}^{k} = \begin{cases} 1 \; if \; j^{th} \; patient \; to \; visit \; provider \; k \; (after \; crossovers) \; type \; LC \; under \\ scenario \; s \\ 0 \; otherwise \end{cases}$

- $U_{j,s}^{k} = \begin{cases} 1 \; if \; j^{th} \; patient \; to \; visit \; provider \; k \; (after \; crossovers) \; type \; SD \; under \\ scenario \; s \\ 0 \; otherwise \end{cases}$

- $n_{i,s} = \begin{cases} 1 \; if \; the \; earliest \; nurse \; available \; to \; see \; patient \; i \; is \; the \; one \\ that \; serves \; patient \; i-1, that \; is, there \; is \; some \; earlier \; patient \\ that \; is \; still \; seeing \; the \; other \; nurse, under \; scenario \; s \\ 0 \; otherwise \end{cases}$

- $p_{j,s}^{k} = \begin{cases} 1 \; if \; the \; crossover \; occurs, that \; is, the \; j^{th} patient \; to \; see \; provider \; k \\ is \; the \; j+1^{st} patient \; in \; their \; appointment \; schedule, under \; scenario \; s \\ 0 \; otherwise \end{cases}$

- $y_{i,s}^{start}$: Start time of patient i with a nurse, under scenario s

- $\tau_{i,s}^{N}$: Service time of patient i with a nurse, under scenario s

- $y_{i,s}^{finish}$: Finish time of patient i with a nurse, under scenario s

- $t_{j,s}^{k}$: Finish time of j^{th} patient of provider k with a nurse, under scenario s

- $N_{i,s}^{max}$: Maximum of the finish time of patients $1..i-1$, under scenario s

- $z_{j,s}^{k,start}$: Start time of j^{th} patient to visit provider k, under scenario s

- $\tau_{j,s}^{P_k}$: Service time of j^{th} patient to visit provider k with their provider, under scenario s

- $z_{j,s}^{k,finish}$: Finish time of j^{th} patient to visit provider k, under scenario s

5.2.2.4. **Model**

Minimize

$$\frac{1}{S}(\alpha[\sum_s((z_{J1,s}^{1,finish} - \sum_{j=1}^{J1} \tau_{j,s}^{P1}) + (z_{J2,s}^{2,finish} - \sum_{j=1}^{J2} \tau_{j,s}^{P2}))] + \beta[\sum_s \sum_{i=1}^{n}(y_{i,s}^{start} - 15X_i) + \sum_s(\sum_{j=1}^{J1}(z_{j,s}^{1,start} - t_{j,s}^{1}) + \sum_{j=1}^{J2}(z_{j,s}^{2,start} - t_{j,s}^{2})]) \quad (1)$$

subject to

$$y_{1,s}^{start} = 0 \; \forall s \in S \quad (2)$$

$$y_{2,s}^{start} = 0 \; \forall s \in S \quad (3)$$

$$z_{0,s}^{k,finish} = 0 \; \forall k \in K \; \forall s \in S \quad (4)$$

$$X_1 = 0 \quad (5)$$

$$X_2 = 0 \quad (6)$$

$$y_{3,s}^{start} \geq \min(y_{1,s}^{finish}, y_{2,s}^{finish}) \; \forall s \in S \quad (7)$$

$$y_{3,s}^{start} \geq y_{1,s}^{finish} - M_{3,s}^{1} n_{3,s} \; \forall s \in S \quad (7\text{-}1)$$

$$y_{3,s}^{start} \geq y_{2,s}^{finish} - M_{3,s}^{1}(1 - n_{3,s}) \forall s \in S \quad (7\text{-}2)$$

$$N_{3,s}^{max} \geq \max(y_{1,s}^{finish}, y_{2,s}^{finish}) \; \forall s \in S \quad (8)$$

$$N_{i,s}^{max} \geq \max(N_{i-1,s}^{max}, y_{i-1,s}^{finish}) \; \forall i \in 4..I, s \in S \quad (9)$$

$$y_{i,s}^{start} \geq \min(N_{i-1,s}^{max}, y_{i-1,s}^{finish}) \; \forall i \in 4..I, s \in S \quad (10)$$

$$y_{i,s}^{start} \geq N_{i-1,s}^{max} - M_{i,s}^{1} n_{i,s} \; \forall i \in 4..I, s \in S \quad (10\text{-}1)$$

$$y_{i,s}^{start} \geq y_{i-1,s}^{finish} - M_{i,s}^{1}(1 - n_{i,s}) \; \forall i \in 4..I, s \in S \quad (10\text{-}2)$$

$$y_{i,s}^{finish} = y_{i,s}^{start} + \tau_{i,s}^{N} \; \forall i \in I, s \in S \quad (11)$$

$$y_{i,s}^{start} \geq 15X_i \; \forall i \in I, s \in S \quad (12)$$

$$A_i + B_i + C_i = 1 \; \forall i \in I \quad (13)$$

$$\sum_{i \; in \; I_k} A_i = H_{HC}^{k} \forall k \in K \quad (14)$$

$$\sum_{i \; in \; I_k} B_i = H_{LC}^{k} \forall k \in K \quad (15)$$

$$\sum_{i \; in \; I_k} C_i = H_{SD}^{k} \forall k \in K \quad (16)$$

$$\tau_{i,s}^{N} = \tau_{i,s}^{N,HC} A_i + \tau_{i,s}^{N,LC} B_i + \tau_{i,s}^{N,SD} C_i \; \forall i \in I, \forall s \in S \quad (17)$$

$$t_{j,s}^{k} = y_{f[j,k],s}^{finish} \; \forall k \in K, j \in J_k, s \in S \quad (18)$$

$$z_{j,s}^{k,start} \geq \max\left(t_{j-1,s}^{k}, min\left(t_{j,s}^{k}, t_{j+1,s}^{k}\right)\right) \forall k \in K, j \in J_k, s \in S \quad (19)$$

$$z_{j,s}^{k,start} \geq \; t_{j-1,s}^{k} - M_{j,s}^{2,k}\left(1 - p_{j-1,s}^{k}\right) \forall k \in K, j \in 2..J_k, s \in S \quad (19\text{-}1)$$

$$z_{j,s}^{k,start} \geq \; t_{j,s}^{k} - M_{j,s}^{2,k}\left(p_{j-1,s}^{k}\right) - M_{j,s}^{2,k}\left(p_{j,s}^{k}\right) \forall k \in K, j \in 2..J_k - 1, s \in S \quad (19\text{-}2)$$

$$z_{1,s}^{k,start} \geq t_{1,s}^{k} - M_{1,s}^{2,k} p_{1,s}^{k} \; \forall k \in K, s \in S \quad (19\text{-}3)$$

$$z_{J_k,s}^{k,start} \geq t_{1,s}^{k} - M_{J_k-1,s}^{2,k} p_{J_k-1,s}^{k} \; \forall k \in K, s \in S \quad (19\text{-}4)$$

$$z_{j,s}^{k,start} \geq \; t_{j+1,s}^{k} - M_{j,s}^{2,k}\left(1 - p_{j,s}^{k}\right) \forall k \in K, j \in 1..J_k - 1, s \in S \quad (19\text{-}5)$$

$$z_{j,s}^{k,finish} = z_{j,s}^{k,start} + \tau_{j,s}^{P_k} \; \forall k \in K, j \in J_k, s \in S \quad (20)$$

$$z_{j,s}^{k,start} \geq \; z_{j-1,s}^{k,finish} \; \forall k \in K, j \in J_k, s \in S \quad (21)$$

$$D_j^k = A_{f[j,k]} \; \forall k \in K, j \in J_k \tag{22}$$

$$E_j^k = B_{f[j,k]} \; \forall k \in K, j \in J_k \tag{23}$$

$$F_j^k = C_{f[j,k]} \; \forall k \in K, j \in J_k \tag{24}$$

$$Q_{j,s}^k \geq \max\left(D_{j-1,s}^k, min\left(D_{j,s}^k, D_{j+1,s}^k\right)\right) \forall k \in K, j \in J_k, s \in S \tag{25}$$

$$Q_{j,s}^k \geq D_{j-1,s}^k - \left(1 - p_{j-1,s}^k\right) \forall k \in K, j \in 2..J_k, s \in S \tag{25-1}$$

$$Q_{j,s}^k \geq D_{j,s}^k - \left(p_{j-1,s}^k\right) - \left(p_{j,s}^k\right) \forall k \in K, j \in 2..J_k - 1, s \in S \tag{25-2}$$

$$Q_{1,s}^k \geq D_{1,s}^k - p_{1,s}^k \; \forall k \in K, s \in S \tag{25-3}$$

$$Q_{J_k,s}^k \geq D_{1,s}^k - p_{J_k-1,s}^k \; \forall k \in K, s \in S \tag{25-4}$$

$$Q_{j,s}^k \geq D_{j+1,s}^k - \left(1 - p_{j,s}^k\right) \forall k \in K, j \in 1..J_k - 1, s \in S \tag{25-5}$$

$$W_{j,s}^k \geq \max\left(E_{j-1,s}^k, min\left(E_{j,s}^k, E_{j+1,s}^k\right)\right) \forall k \in K, j \in J_k, s \in S \tag{26}$$

$$W_{j,s}^k \geq E_{j-1,s}^k - \left(1 - p_{j-1,s}^k\right) \forall k \in K, j \in 2..J_k, s \in S \tag{26-1}$$

$$W_{j,s}^k \geq E_{j,s}^k - \left(p_{j-1,s}^k\right) - \left(p_{j,s}^k\right) \forall k \in K, j \in 2..J_k - 1, s \in S \tag{26-2}$$

$$W_{1,s}^k \geq E_{1,s}^k - p_{1,s}^k \; \forall k \in K, s \in S \tag{26-3}$$

$$W_{J_k,s}^k \geq E_{1,s}^k - p_{J_k-1,s}^k \; \forall k \in K, s \in S \tag{26-4}$$

$$W_{j,s}^k \geq E_{j+1,s}^k - \left(1 - p_{j,s}^k\right) \forall k \in K, j \in 1..J_k - 1, s \in S \tag{26-5}$$

$$U_{j,s}^k \geq \max\left(F_{j-1,s}^k, min\left(F_{j,s}^k, F_{j+1,s}^k\right)\right) \forall k \in K, j \in J_k, s \in S \tag{27}$$

$$U_{j,s}^k \geq F_{j-1,s}^k - \left(1 - p_{j-1,s}^k\right) \forall k \in K, j \in 2..J_k, s \in S \tag{27-1}$$

$$U_{j,s}^{k} \geq F_{j,s}^{k} - \left(p_{j-1,s}^{k}\right) - \left(p_{j,s}^{k}\right) \forall k \in K, j \in 2..J_k - 1, s \in S \quad (27\text{-}2)$$

$$U_{1,s}^{k} \geq F_{1,s}^{k} - p_{1,s}^{k} \; \forall k \in K, s \in S \quad (27\text{-}3)$$

$$U_{J_k,s}^{k} \geq F_{1,s}^{k} - p_{J_k-1,s}^{k} \; \forall k \in K, s \in S \quad (27\text{-}4)$$

$$U_{j,s}^{k} \geq F_{j+1,s}^{k} - \left(1 - p_{j,s}^{k}\right) \forall k \in K, j \in 1..J_k - 1, s \in S \quad (27\text{-}5)$$

$$\tau_{j,s}^{P_k} = \tau_{j,s}^{P_k,HC} Q_{j,s}^{k} + \tau_{j,s}^{P_k,LC} W_{j,s}^{k} + \tau_{j,s}^{P_k,SD} U_{j,s}^{k} \; \forall k \in K, j \in J_k, \forall s \in S \quad (28)$$

$$Q_{j,s}^{k} + W_{j,s}^{k} + U_{j,s}^{k} = 1 \; \forall k \in K, j \in J_k, \forall s \in S \quad (29)$$

$$p_{j,s}^{k} + p_{j+1,s}^{k} \leq 1 \forall k \in K, j \in 1..J_k - 1, s \in S \quad (30)$$

Objective function (1) minimizes the expected weighted provider idle time and patient wait time. Constraints (2) and (3) make sure that first two patients start their nurse time at the beginning of the planning horizon, since there are two nurses in the practice who can see them. Constraints (4) initializes the artificial zeroth patient finish times as zero. Constraints (5) and (6) sets the appointment times of the first two patients to the very first appointment. Constraints (7) ensure that the third patient's nurse start time is after the minimum of the first two patients, since that is the earliest time that a nurse becomes available. Constraints (8) ensure the maximum finish time with nurse of the first two patients is captured by variables $N_{3,s}^{max}$. Constraints (9) have a similar purpose of calculating the maximum finish time with nurse of patients 1 through $i - 1$. Constraints (10) calculate the nurse start time of patient i as after the minimum of nurse finish time of patient $i - 1$ and the maximum of nurse finish times of patients 1 through $i - 2$, since this is when one of the two nurses becomes available. Constraints (11) calculate the nurse finish times of patients as the addition of the nurse start time and the nurse service time.

Constraints (12) ensure that the nurse start times of patients are after the arrival of patients. Constraints (13) assign exactly one type of patient to each patient index in the schedule. Constraints (14), (15) and (16) ensure that all types of patients of all providers are assigned to their corresponding patient indexes. Constraints (17) calculate the nurse service time of patients according to their assigned types. Constraints (18) capture the nurse finish time of patients of provider *k* in the $t_{j,s}^{k}$ variables. These constraints basically match the nurse finish times from nurse indexes to provider indexes through *f[j,k]* parameters in their original order. Then, constraints (19) calculate the provider start time of j^{th} patient to be seen by provider *k* according to possible crossovers. In our calculations, we limit the number of crossovers at the provider step to 1. This allows us to trace the patients according to their types, which allows us to calculate their correct service times. One-step crossover on provider step is actually a very reasonable assumption since it allows up to 3 crossovers in the nurse step. To illustrate, let's say 1,3,5 is first provider's patients and 2,4,6 is second provider's patients. If patient 1 had two crossovers before them in the provider step, it means actually 4 people crossed over them in the nurse step. Thus, they are seen at patient 5's position on provider step and patients 2,3,4 and 5 were finished before them in nurse step. Considering 15-minute slots, hence on average 15 minutes per patient, this corresponds to 60 minutes of nurse time for patient 1 on average, which is almost unrealistically long. Even though there are some cases that this might happen, we assume the patient with the later appointment time is waited for a bit and patient with earlier appointment time is allowed in first. How it is allowed using constraints (19) is that the patient that's going to be seen in j^{th} position can either be $j-1^{st}, j^{th}, or\ j+1^{st}$ originally. Constraints (19-1) are active if patient j

crosses over patient $j-1$ and patient $j-1$ is seen at j^{th} position. Constraints (19-5) are active if patient $j+1$ crosses over patient j and patient $j+1$ is seen at j^{th} position. Constraints (19-2), (19-3) and (19-4) are active if no crossovers happen, (19-3) and (19-4) being specifically designed for the very first and last patient of provider *k*. Constraints (20) calculate the provider finish time of patients as the addition of provider start time and provider service time. Constraints (21) ensure the start time of patient *j* of provider *k* is after finish time of patient $j-1$ of the same provider. Constraints (22)-(24) match the patient types from the nurse step to the provider step using their original indexes. Then, constraints (25)-(27) assign the actual patient types according to the order they are seen in the provider step, using a similar procedure to constraints (19). Constraints (28) calculate the provider service times of patients according to their corresponding types. Constraints (29) ensure there is only one type of patient assigned to each index at provider step and constraints (30) ensure the number of crossovers is limited to 1 in the provider step.

Initial tests show that this problem is actually more difficult to solve than our problem in Chapter 3 resulting in higher optimality gaps and smaller solvable instances, possibly due to higher number of binary variables in the second stage. However, since the complexity comes from tracking patient types for crossover purposes, a model without crossover (where patients are seen in the order that they are scheduled) seems promising. Also, the tightening constraints, big-M parameter improvements and lower-bounding techniques from Chapter 3 can be adapted to be applied to this problem as well, promising potential improvements to this model.

Therefore, for future research directions, we propose to model this problem without crossovers, generate random instances for this problem and for the problem without

crossovers and solve them, apply tightening constraints, big-M parameter improvements and lower-bounding techniques similar to the ones from Chapter 3 and improve the solvable problem sizes and optimality gaps, perform sensitivity analyses and an investigation on problem characteristics such as the effects of service time variability, nurse and provider flexibility and patient crossovers, similar to the ones from Chapter 3 and come up with scheduling guidelines.

5.3. **Extensions of Chapter 4**

Future research directions for this chapter include:

- Consideration of machine availability / maintenance activities similar to Ramezanian, Saidi-Mehrabad and Fattahi (2013).
- Generalization of setup carry overs to sequence-dependent setups similar to Mohammadi et al. (2010)
- Implementation of our models and methods on the benchmark instances from the literature.
- Generation and analysis of random instances based on our test application.
- Comparison of our models and methods to heuristics from the literature on our test application.
- Connection of execution models going from one execution time-horizon to another, guided by the planning model and implementation of our tools at the manufacturing site.
- Comparison of current scheduling guidelines performed at the site to our methods.

- Analysis of optimal solutions and derivation of easy-to-implement scheduling guidelines.

CHAPTER 6

CONCLUSION

In this dissertation, we presented three essays on mathematical models and solution techniques driven by data for scheduling optimization for novel problems in manufacturing and healthcare.

In Chapter 2, we presented an essay regarding computational comparison of two exact optimization modeling techniques (direct positional and relative positional variables) on a family of machine scheduling problems. We presented different mathematical models using these techniques for Parallel Machine and Flexible Flow Shop Scheduling problems, with and without sequence-dependent setups. We generated random instances of different sizes driven by data from a manufacturing company (Artaic) and solved them using CPLEX Studio 12.7 optimization package by coding the models in this environment for our computational study. We observed that one modeling technique (direct positional variables) dominates the other (relative positional variables) significantly using a statistical test (t-test). We also implemented the most practical model using Excel and Node.js, a cloud computing software.

In Chapter 3, we built upon Alvarez Oh (2015)'s work and presented another essay regarding a challenging scheduling problem in healthcare: the team primary care practice. Our contributions include a literature review, generating a lower-bounding technique that can solve larger problem sizes and decrease optimality gaps, generalizing her results and guidelines using a new set of problem instances from a different service time distribution, sensitivity analyses of different problem characteristics such as cost ratio, service time variability, nurse flexibility and patient crossovers. We used the same computational

setting (CPLEX 12.6 optimization package) as Alvarez Oh (2015) for our computational experiments.

In Chapter 4, we introduced a generalization of Multi-Level Capacitated Lot-Sizing Problem (MLCLSP) called Hierarchical Job Shop Scheduling Planning and Execution Problem (HJSP), inspired by a real-life example observed in aerospace industry. We proposed mathematical models and implemented our methods on real-life data instances gathered from the aerospace parts manufacturer. Due to the size of the problem, we have introduced methods to reduce the size of the problem that exploit aspects of the data structure and were able to solve the instances within a reasonable computational time window to optimality. We have performed additional sensitivity analyses based on different modeling techniques and objective functions using KPIs relevant for the company.

In Chapter 5, we presented extensions of the research in Chapters 2, 3 and 4, outlined future research directions and proposed further research. These include additional computational experiments similar to the ones on Chapter 2, generalization of modeling techniques to more machine scheduling problem families such as Job Shop and Open Shop Scheduling and comparison of exact methodologies to heuristics and meta-heuristics. They also include generalization of models from Chapter 3 by including flexibility in second stage (provider) and relaxing the homogeneous patient assumption. The preliminary models are presented, and further computational experiments are proposed. Finally, research on Chapter 4 can be extended further by solving benchmark instances from the literature and comparing heuristics against our exact methods. Comparisons between current scheduling methods and our tools can also be made.

Models can be generalized by considering additional aspects of the problem. Easy-to-implement scheduling guidelines can be derived by analyzing the optimal solutions. Finally, implementation can be further improved upon and made useful by connecting different execution horizons together using the outputs from the planning model and execution model and iteratively solving the execution model.

APPENDIX A

TP MODEL WITHOUT PATIENT CROSSOVERS

The model without crossovers is identical for the nurse step, but lends itself to a much easier formulation of the provider step as the sequence of patients is fixed regardless of the scenario.

$$Min.\ \frac{1}{S}\left(\alpha\left[\sum_{k}\sum_{s}\left(\left(z_{J_k,s}^{k,finish}-\sum_{k=1}^{J_k}\tau_{j,s}^{P_k}\right)\right)\right]+\beta\left[\sum_{s}\sum_{i=1}^{n}\left(y_{i,s}^{start}-15X_i\right)+\sum_{s}\left(\sum_{k}\sum_{j=1}^{J_k}\left(z_{j,s}^{k,start}-t_{j,s}^{k}\right)\right)\right]\right) \quad (1)$$

Subject to.

$$y_{i,s}^{start}=0 \quad \forall s\in S, i=1,2 \quad (2)$$

$$z_{0,s}^{k,finish}=0 \quad \forall k\in K, s\in S \quad (3)$$

$$X_i=0 \quad i=1,2 \quad (4)$$

$$y_{3,s}^{start}\geq \min(y_{1,s}^{finish}, y_{2,s}^{finish}) \quad \forall s\in S \quad (5)$$

$$N_{3,s}^{max}\geq \max(y_{1,s}^{finish}, y_{2,s}^{finish}) \quad \forall s\in S \quad (6)$$

$$N_{i,s}^{max}\geq \max(N_{i-1,s}^{max}, y_{i-1,s}^{finish}) \quad \forall i\in 4..I, s\in S \quad (7)$$

$$y_{i,s}^{start}\geq \min(N_{i-1,s}^{max}, y_{i-1,s}^{finish}) \quad \forall i\in 4..I, s\in S \quad (8)$$

$$y_{i,s}^{finish}=y_{i,s}^{start}+\tau_{i,s}^{N} \quad \forall i\in I, s\in S \quad (9)$$

$$y_{i,s}^{start}\geq 15X_i \quad \forall i\in I, s\in S \quad (10)$$

$$t_{j,s}^{k}=y_{f[j,k],s}^{finish} \quad \forall k\in K, j\in J_k, s\in S \quad (11)$$

$$z_{j,s}^{k,start}\geq t_{j,s}^{k} \quad \forall k\in K, j\in J_k, s\in S \quad (12)$$

$$z_{j,s}^{k,finish}=z_{j,s}^{k,start}+\tau_{j,s}^{P_k} \quad \forall k\in K, j\in J_k, s\in S \quad (13)$$

$$z_{j,s}^{k,start}\geq z_{j-1,s}^{k,finish} \quad \forall k\in K, j\in J_k, s\in S \quad (14)$$

$$X\geq 0, INT;\ y^{start}, y^{finish}, z^{start}, z^{finish}\geq 0$$

APPENDIX B

COMPUTATIONAL RESULTS FOR LOGNORMALLY DISTRIBUTED NURSE AND PROVIDER SERVICE TIMES

Computational Performance as Service Time Variance Increases

Without Scenario-Groups Lower Bounds			
Gaps	Regular Variance	Doubled Variance	Quadrupled Variance
Medium Instances	12.71%	10.64%	7.12%
Large Instances	22.41%	18.39%	11.15%
With Scenario-Groups Lower Bounds			
Gaps	Regular Variance	Doubled Variance	Quadrupled Variance
Medium Instances	3.81%	3.66%	3.44%
Large Instances	5.05%	5.14%	4.32%

Table B.1 Optimality Gaps for Medium (8 patients per provider) and Large Instances (10 patients per provider) with Lognormally Distributed Service Times with and without Lower Bounds Created by Solving 100 Groups of 10-Scenario Problems

Schedule Sensitivity to Service Time Variance and Idle vs. Wait Time Cost Ratio

The optimal schedules displayed below show 1) when idle time is prioritized with a cost ratio of 4, fewer empty slots are needed as variability increases, and 2) as a heavier weight is placed on wait time, with cost ratios of 2 and 1, more slack is added and variability does not affect the optimal schedule.

	Schedule for Regular Variance		**Schedule for Doubled Variance**		**Schedule for Quadrupled Variance**	
Time	PCP 1	PCP 2	PCP 1	PCP 2	PCP 1	PCP 2
0:00	■	■	■	■	■	■
0:15	■	■	■	■	■	■
0:30	■		■	■	■	■
0:45	■	■	■		■	
1:00		■		■	■	■
1:15	■		■	■		■
1:30		■				

Table B.2 Schedules for Small Instances (5 patients per provider) with Lognormally Distributed Service Times for Cost Ratio 4 (0.8:0.2 idle time/wait time) and for Different Service Time Variance

	Schedule for Regular Variance		**Schedule for Doubled Variance**		**Schedule for Quadrupled Variance**	
Time	PCP 1	PCP 2	PCP 1	PCP 2	PCP 1	PCP 2
0:00	■	■	■	■	■	■
0:15	■	■	■	■	■	■
0:30	■		■		■	
0:45		■		■		■
1:00	■	■	■		■	
1:15	■			■	■	■
1:30		■	■	■		■

Table B.3 Schedules for Small Instances (5 patients per provider) with Lognormally Distributed Service Times for Cost Ratio 2 (0.67:0.33 idle time/wait time) and for Different Service Time Variance

	Schedule for Regular Variance		Schedule for Doubled Variance		Schedule for Quadrupled Variance	
Time	PCP 1	PCP 2	PCP 1	PCP 2	PCP 1	PCP 2
0:00	■	■	■	■	■	■
0:15	■		■		■	
0:30		■		■		■
0:45	■		■		■	
1:00		■		■		■
1:15	■		■		■	
1:30		■		■		■
1:45	■		■		■	
2:00		■		■		■

Table B.4 Schedules for Small Instances (5 patients per provider) with Lognormally Distributed Service Times for Cost Ratio 1 (0.5:0.5 idle time/wait time) and for Different Service Time Variance

	CR1			CR2			CR4		
	Mean	Median	90th Percentile	Mean	Median	90th Percentile	Mean	Median	90th Percentile
	Regular Variance								
Wait	3.69	1.7	9.3	9.44	6.4	21.2	10.35	7.2	23.52
Idle	56.39	55.5	78	36.08	34.5	54.5	34.92	33.5	53.55
	Objective Function: 74.84			Objective Function: 79.56			Objective Function: 76.57		
	Doubled Variance								
Wait	5.59	2.7	14.31	10.05	6.1	23.92	14.41	9.6	32.62
Idle	61.12	60.25	89.5	46.06	43	73	38.32	34.5	65
	Objective Function: 89.07			Objective Function: 94.90			Objective Function: 90.14		
	Quadrupled Variance								

Wait	7.98	3.4	20.51	13.3	7.25	33.03	17.79	10.7	43.22
Idle	66.9	64	103.05	49.93	44.5	85.1	43.23	36.5	80.05
	Objective Function: 106.81			Objective Function: 110.89			Objective Function: 104.75		

Table B.5 Wait time vs. Idle Time(min) for Small Instances (5 patients per provider) with Lognormally Distributed Service Times for different Cost Ratios (4 (0.8:0.2 idle time/wait time), 2 (0.67:0.33 idle time/wait time) and 1 (0.5:0.5 idle time/wait time))

Effect of Crossovers

	No crossover			Crossover		
	Mean	Median	90th Percentile	Mean	Median	90th Percentile
	Regular Variance					
Wait	10.56	7.1	24.21	10.35	7.2	23.52
Idle	37.28	35.5	58	34.92	33.5	53.55
	Objective Function: 80.77			Objective Function: 76.57		
	Doubled Variance					
Wait	17.07	11.35	39.62	14.41	9.6	32.62
Idle	41.08	37	69.5	38.32	34.5	65
	Objective Function: 99.87			Objective Function: 90.13		
	Quadrupled Variance					
Wait	21.75	13.65	50.55	17.79	10.7	43.22
Idle	48.95	41	89	43.23	36.5	80.05
	Objective Function: 121.82			Objective Function: 104.75		

Table B.6 Wait time vs. Idle Time (min) for Small Instances (5 patients per provider) with Lognormally Distributed Service Times for Models with and without Crossovers for Cost Ratio of 4 (0.8:0.2 weights on idle time/wait time)

	No crossover			Crossover		
	Mean	Median	90th Percentile	Mean	Median	90th Percentile
	Regular Variance					
Wait	4.46	2	11.41	3.69	1.7	9.3

Idle	57.02	56.5	79	56.39	55.5	78
	Objective Function: 79.32			Objective Function: 74.84		
	Doubled Variance					
Wait	7.51	3.5	19.13	5.59	2.7	14.31
Idle	62.83	61	92	61.12	60.25	89.5
	Objective Function: 100.38			Objective Function: 89.07		
	Quadrupled Variance					
Wait	12.08	5.4	31.21	7.98	3.4	20.51
Idle	70.04	66	108.05	66.9	64	103.05
	Objective Function: 130.44			Objective Function: 106.81		

Table B.7 Wait time vs. Idle Time (min) for Small Instances (5 patients per provider) with Lognormally Distributed Service Times for Models with and without Crossovers for Cost Ratio of 1 (0.5:0.5 weights on idle time/wait time)

Instance	Crossover Percentage
Small instance, regular variance	5.35%
Small instance, doubled variance	10.34%
Small instance, quadrupled variance	13.40%
Medium instance, regular variance	7.49%
Medium instance, doubled variance	12.39%
Medium instance, quadrupled variance	13.70%
Large instance, regular variance	8.95%
Large instance, doubled variance	18.64%
Large instance, quadrupled variance	15.13%

Table B.8 Percentages of Crossovers for Instances with Lognormally Distributed Service Times

Effect of Decreasing the Length Appointment Time Slots

Our model considers 15min appointment slots; that is, patients will only be given appointments at the quarters of the hour (e.g. 8AM, 8:15AM, 8:30AM, etc.). Below, we

first compare the performance of these 15min slot appointments to 5min slots (e.g. 8AM, 8:05AM, 8:10AM, etc.) As an extreme case, we also consider the continuous appointment problem, where patients can be given appointments at any time (e.g. 8:03AM).

	5 minutes Appointment			15 minutes Appointment		
	Mean	Median	90th Percentile	Mean	Median	90th Percentile
Wait	9.91	6.6	23.3	10.35	7.2	23.52
Idle	34.53	33	53.5	34.92	33.5	53.55
	Objective Function: 75.07			Objective Function: 76.57		

Table B.9 Wait Time vs. Idle Time for Small Instance (5 patients per provider) with Lognormally Distributed Service Times with 5-minute vs. 15-minute Appointment Intervals

	Continuous Appointment			15 minutes Appointment		
	Mean	Median	90th Percentile	Mean	Median	90th Percentile
	Regular Variance					
Wait	10.45	7	24.4	10.35	7.2	23.52
Idle	33.68	32	53	34.92	33.5	53.55
	Objective Function: 74.8			Objective Function: 76.57		
	Doubled Variance					
Wait	13	8.2	31.93	14.41	9.6	32.62
Idle	39.15	35.5	64.55	38.32	34.5	65
	Objective Function: 88.64			Objective Function: 90.13		
	Quadrupled Variance					
Wait	15.79	9.05	39.81	17.79	10.7	43.22
Idle	44.86	38	82	43.23	36.5	80.05
	Objective Function: 103.36			Objective Function: 104.75		

Table B.10 Wait Time vs. Idle Time for Small Instances (5 patients per provider) with Lognormally Distributed Service Times with Continuous vs. 15-minute Appointment Intervals

APPENDIX C

PROOF OF THEOREMS 1 AND 2 (FROM ALVAREZ OH (2015))

Proof of Theorem 1

For constraints (7) to be valid, we must ensure that

$$M_{i,s}^{1} \geq \left(N_{i-1,s}^{max} - y_{i-1,s}^{finish}\right)^{+} \qquad \forall i \in 4..I, s \in S$$

where $N_{i-1,s}^{max}$ is the maximum of the finish times of patients 1 through i-2 with a nurse for that scenario. That is, M^1 must be an upper bound on the difference in finish times with nurse of the two patients that are seen by a nurse at the time patient i-1 starts service, and it can vary for each patient in the sequence and from scenario to scenario. We consider two cases: In Case1, the finish time of patient *i*-1 with nurse is greater than or equal to the maximum of the finish times of patient from 1 to *i*-2 with nurses; and in Case2, it is strictly lower.

Case 1: $N_{i-1,s}^{max} \leq y_{i-1,s}^{finish}$

In this case, observe that

e. The appointment time of patient *i*-1 is at most 30 minutes after that of patient *i*-2, and thus $X_{i-1} \leq y_{i-2,s}^{start} + 30$.

f. By definition: $N_{i-1,s}^{max} \geq y_{i-2,s}^{finish} = y_{i-2,s}^{start} + \tau_{i-2,s}^{N}$, and thus $N_{i-1,s}^{max} - \tau_{i-2,s}^{N} \geq y_{i-2,s}^{start}$.

g. Combining the two, we get that patient *i*-1 is available at time:

$X_{i-1} \leq y_{i-2,s}^{start} + 30 \leq N_{i-1,s}^{max} - \tau_{i-2,s}^{N} + 30.$

h. A nurse will be available to serve patient *i*-1 at time $N_{i-1,s}^{max}$ or earlier.

i. The start time of patient *i*-1 with the nurse is

$y_{i-1,s}^{start} \leq Max\{ N_{i-1,s}^{max}, N_{i-1,s}^{max} - \tau_{i-2,s}^{N} + 30\}.$

Thus, the difference $y_{i-1,s}^{finish} - N_{i-1,s}^{max}$ is bounded by $\tau_{i-1,s}^{N} + Max\{0,30 - \tau_{i-2,s}^{N}\}$.

Case 2: $N_{i-1,s}^{max} > y_{i-1,s}^{finish}$

In this case, observe that while patient *i*-1 has finished with one nurse, say nurse1 w.l.o.g., the other nurse, nurse2, is still busy with an earlier patient. The difference between the two can be calculated depending on which patient is still with nurse2. If patient *r* is still with nurse2, it means that patients *r*+1, *r*+2, …, through *i*-1 are seen by nurse1, we have that:

e. $N_{i-1,s}^{max} = y_{r,s}^{start} + \tau_{r,s}^{N}$

f. $y_{i-1,s}^{finish} \geq y_{r+1,s}^{start} + \tau_{r+1,s}^{N} + \tau_{r+2,s}^{N} + \cdots + \tau_{i-1,s}^{N}$

g. $y_{r,s}^{start} \leq y_{r+1,s}^{start}$ since patients are seen by the nurse in the order of their appointment times, $X_1 \leq X_2 \leq \cdots \leq X_I$.

h. Thus, the difference $N_{i-1,s-}^{max} - y_{i-1,s}^{finish} \leq \tau_{r,s}^{N} - \sum_{u=r+1}^{i-1} \tau_{u,s}^{N}$

$\underset{r=1,\ldots,i-2}{Max} \left\{\tau_{r,s}^{N} - \sum_{u=r+1}^{i-1} \tau_{u,s}^{N}\right\}$ will provide the tight bound.

The overall bound on the difference for both cases then is

$Max\{Case1, Case\ 2\} = Max\left\{\tau_{i-1,s}^{N} + Max\{0,30 - \tau_{i-2,s}^{N}\}, \underset{r=1,\ldots,i-2}{Max} \{\tau_{r,s}^{N} - \sum_{u=r+1}^{i-1} \tau_{u,s}^{N}\}\right\}$

Proof of Theorem 2

For the M2 constraints to be valid we must ensure that

$$M_{j,s}^{2k} \geq \left(P_{j,s}^{k,max} - y_{i+2,s}^{k,finish}\right)^{+}$$

where $P_{j,s}^{k,max}$ is the maximum of the finish times at the nurse step of patients 1,…, *j* of provider *k*'s panel under scenario *s*. That is, M2 is a bound on the difference of nurse finish times of subsequent patients seen by provider *k*, for *j*=1, 2, …, J_k, for provider *k*. Observe that if $t_{j,s} = y_{i,s}^{finish}$ then $t_{j+1,s} = y_{i+2,s}^{finish}$ where *j* is the *j*th patient in provider's *k* panel, who is the *i*th patient in the practice. We again consider two cases: In Case1 the nurse finish time of patient *i*+2 is greater than or equal to the maximum of the nurse finish times of patients of provider k up to patient *j*; and in Case2 it is lower.

Case 1: $P_{j,s}^{k,max} \leq y_{i+2,s}^{finish}$

In this case, observe that

a. The appointment time of patient *i*+2 is at most 30 minutes after that of patient *i*; thus

$$X_{i+2} \leq y_{i,s}^{start} + 30$$

b. By definition: $P_{j,s}^{k,max} \geq y_{i,s}^{finish} = y_{i,s}^{start} + \tau_{i,s}^{N}$, and thus $P_{j,s}^{k,max} - \tau_{i,s}^{N} \geq y_{i,s}^{start}$

c. Combining the two, we get that patient *i*+2 is available at time:

$$X_{i+2} \leq y_{i,s}^{start} + 30 \leq P_{j,s}^{k,max} - \tau_{i,s}^{N} + 30$$

d. Patient *i*+1 (from the other provider's panel) will be seen by a nurse at a time no later than $Max\{P_{j,s}^{k,max}, P_{j,s}^{k,max} - \tau_{i,s}^{N} + 30\}$. This is using that consecutive patients arrive at most 30 minutes apart to the practice.

e. A nurse will be available for patient *i*+2 at time $Max\{P_{j,s}^{max}, P_{j,s}^{max} - \tau_{i,s}^{N} + 30\} + \tau_{i+1,s}^{N}$, or earlier. This is a bound on the time a nurse will be available if patient *i*+2 is scheduled to see a nurse right after patient *i+1* finishes.

f. The start time of patient *i*+2 with the nurse is

$$y_{i+2,s}^{start} \leq Max\{ P_{j,s}^{max} - \tau_{i,s}^{N} + 30, Max\{ P_{j,s}^{max}, P_{j,s}^{max} - \tau_{i,s}^{N} + 30\} + \tau_{i+1,s}^{N}\}$$

$$= \tau_{i+1,s}^{N} + P_{j,s}^{max} + Max\{ 0, \tau_{i,s}^{N} + 30\}$$

Thus, the difference $y_{i+2,s}^{finish} - P_{j,s}^{max}$ is bounded by $\tau_{i+2,s}^{N} + \tau_{i+1,s}^{N} + Max\{ 0, -\tau_{i,s}^{N} + 30\}$

Case 2: $P_{j,s}^{max} > y_{i+2,s}^{finish}$

In this case, observe that while patient *i*+2 has finished with one nurse, say nurse1 w.l.o.g., the other nurse, nurse2, is still busy with an earlier patient $r \leq i$ from the same provider. The difference between the two can be calculated depending on which patient is still with nurse2. If patient *r* is still with nurse2, it means that patients *r*+1, *r*+2, …, through *i*+1 were seen by nurse1, we have that:

a. $P_{j,s}^{max} = y_{r,s}^{start} + \tau_{r,s}^{N}$

b. $y_{i+2,s}^{finish} \geq y_{r+1,s}^{start} + \tau_{r+1,s}^{N} + \tau_{r+2,s}^{N} + \cdots + \tau_{i+2,s}^{N}$

c. $y_{r+1,s}^{start} \geq y_{r,s}^{start}$ since patients are seen in the order of their appointment times, $X_1 \leq X_2 \leq \cdots \leq X_J$.

d. Thus, the difference $P_{j,s}^{max} - y_{i+2,s}^{finish} \leq \tau_{r,s}^{N} - \sum_{u=r+1}^{i+2} \tau_{u,s}^{N}$

The maximum in $\underset{r=1,\ldots,j,r\ in\ provider's\ k\ panel}{Max} \left\{\tau_{r,s}^{N} - \sum_{u=r+1}^{i+2} \tau_{u,s}^{N}\right\}$ will give us the bound we are looking for in this case.

The overall bound on the difference for both cases then is

$$Max\{Case1, Case\ 2\} = Max\left\{\tau_{i+2,s}^{N} + \tau_{i+1,s}^{N} + Max\{\ 0, -\tau_{i,s}^{N} + 30\}, \underset{r=1,\ldots,j,r\ in\ provider's\ k\ panel}{Max} \left\{\tau_{r,s}^{N} - \sum_{u=r+1}^{i+2} \tau_{u,s}^{N}\right\}\right\}$$

BIBLIOGRAPHY

Ahmadi-Javid, A., Jalali, Z., & Klassen, K. J. (2017). Outpatient appointment systems in healthcare: A review of optimization studies. *European Journal of Operational Research*, *258*(1), 3-34.

Alvarez Oh, H.J. (2015). *Guidelines for Scheduling in Primary Care: An Empirically Driven Mathematical Programming Approach* (Doctoral dissertation). Retrieved from https://scholarworks.umass.edu/cgi/viewcontent.cgi?article=1410&context=dissertations_2

Alvarez-Oh, H. J., Balasubramanian, H., Koker, E., & Muriel, A. (2018). Stochastic appointment scheduling in a team primary care practice with two flexible nurses and two dedicated providers. *Service Science*, *10*(3), 241-260.

Bai, M., Storer R.H. & Tonkay, G.L. (2016). A sample gradient-based algorithm for a multiple-OR and PACU surgery scheduling problem, *IISE Transactions*, DOI: 10.1080/0740817X.2016.1237061

Balasubramanian, H., Biehl, S., Dai, L., & Muriel, A. (2014). Dynamic allocation of same-day requests in multi-physician primary care practices in the presence of prescheduled appointments. *Health Care Management Science, 17*(1), 31–48.

Billington, P. J., McClain, J. O., & Thomas, L. J. (1983). Mathematical programming approaches to capacity-constrained MRP systems: Review, formulation and problem reduction. *Management Science*, *29*(10), 1126-1141.

Bodenheimer, T., and Pham, H. H. (2010) Primary care: current problems and proposed solutions. *Health Affairs (ProjectHope)*, 29, 799–805.

Buschkühl, L., Sahling, F., Helber, S., & Tempelmeier, H. (2010). Dynamic capacitated lot-sizing problems: a classification and review of solution approaches. *Or Spectrum*, *32*(2), 231-261.

Castaing, J., Cohn, A., Denton, B. T., & Weizer, A. (2016). A stochastic programming approach to reduce patient wait times and overtime in an outpatient infusion center. *IIE Transactions on Healthcare Systems Engineering*, *6*(3), 111-125.

Castro, E., & Petrovic, S. (2012). Combined mathematical programming and heuristics for a radiotherapy pre-treatment scheduling problem. *Journal of Scheduling*, *15*(3), 333–346.

Cayirli, T., and Veral, E. (2003) Outpatient scheduling in health care: A review of literature. Production and Operations Management: an International Journal of the Production and Operations Management Society, 12, 519-549.

Chen, H., & Chu, C. (2003, September). Supply chain planning with order/setup costs and capacity constraints a new Lagrangian relaxation approach. In *2003 IEEE international conference on robotics and automation (Cat. no. 03CH37422)* (Vol. 2, pp. 1743-1748). IEEE.

Conforti, D., Guerriero, F., & Guido, R. (2010). Non-block scheduling with priority for radiotherapy treatments. *European Journal of Operational Research, 201* (1), 289–296.

Demir, Y., Isleyen, K. Evaluation of mathematical models for flexible job-shop scheduling problems. *Applied Mathematical Modelling* 37 (2013) 977–988.

El-Sharo, M. D., Zheng, B., Yoon, S. W., & Khasawneh, M. T. (2015). An overbooking scheduling model for outpatient appointments in a multi-provider clinic. *Operations Research for Health Care, 6*, 1–10.

Hahn-Goldberg, S., Carter, M. W., Beck, J. C., Trudeau, M., Sousa, P., & Beattie, K. (2014). Dynamic optimization of chemotherapy outpatient scheduling with uncertainty. *Health Care Management Science, 17*(4), 379–392.

Harris F.W. (1913) How many parts to make at once. *Factory, The Magazine of Management*, 10(2), 135–6.

Helber, S., & Sahling, F. (2010). A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *International Journal of Production Economics*, *123*(2), 247-256.

Karimi, B., Ghomi, S. F., & Wilson, J. M. (2003). The capacitated lot sizing problem: a review of models and algorithms. *Omega*, *31*(5), 365-378.

Keha, A.B., Khowala, K. Fowler, J.W. (2009). Mixed integer programming formulations for single machine scheduling problems. *Computers & Industrial Engineering, 56*, 357–367.

Kuiper, A., & Mandjes, M. (2015). Appointment scheduling in tandem-type service systems. *Omega*, *57*, 145-156.

Levy, I. (2019). *An optimization tool to solve a capacitated multi-level job-shop scheduling problem with setup times* (Unpublished honor's thesis). University of Massachusetts Amherst, Amherst, MA.

Liang, B., & Turkcan, A. (2015). Acuity-based nurse assignment and patient scheduling in oncology clinics. *Health Care Management Science, 19*(3), 207–226.

Lin, C. K. Y. (2015). An adaptive scheduling heuristic with memory for the block appointment system of an outpatient specialty clinic. *International Journal of Production Research, 53*(24), 7488–7516.

Manne, A. S. (1958). Programming of economic lot sizes. *Management science*, *4*(2), 115-135.

Mohammadi, M., Ghomi, S. F., Karimi, B., & Torabi, S. A. (2010). Rolling-horizon and fix-and-relax heuristics for the multi-product multi-level capacitated lotsizing problem with sequence-dependent setups. *Journal of Intelligent Manufacturing*, *21*(4), 501-510.

Neyshabouri, S. & Berg B.P. (2016). Two-stage robust optimization approach to elective surgery and downstream capacity planning, *European Journal of Operational Research*, http://dx.doi.org/10.1016/j.ejor.2016.11.043

Oh, H. J., Muriel, A., and Balasubramanian H., Atkinson, K., and Ptaszkiewicz, T. (2013) Guidelines for scheduling in primary care under different patient types and stochastic nurse and provider service times. *IIE Transactions on Healthcare Systems Engineering* 3, 4, 263-279.

Pérez, E., Ntaimo, L., Malavé, C. O., Bailey, C., & McCormack, P. (2013). Stochastic online appointment scheduling of multi-step sequential procedures in nuclear medicine. *Health care management science*, 16(4), 281-299.

Pérez, E., Ntaimo, L., Wilhelm, W. E., Bailey, C., & McCormack, P. (2011). Patient and resource scheduling of multi-step medical procedures in nuclear medicine. *IIE Transactions on Healthcare Systems Engineering, 1* (3), 168–184.

Ramezanian, R., Saidi-Mehrabad, M., & Fattahi, P. (2013). MIP formulation and heuristics for multi-stage capacitated lot-sizing and scheduling problem with availability constraints. *Journal of Manufacturing Systems*, *32*(2), 392-401.

Robinson Jr, E. P., & Lawrence, F. B. (2004). Coordinated capacitated lot-sizing problem with dynamic demand: A Lagrangian heuristic. *Decision Sciences*, *35*(1), 25-53.

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

Santos, D.L., Hunsucker, J.L. & Deal, D.E. (1995). Global lower bounds for flow shops with multiple processors. *European Journal of Operational Research,* 80, 1, 112-120.

Saremi, A., Jula, P., ElMekkawy, T., & Wang, G. G. (2013). Appointment scheduling of outpatient surgical services in a multistage operating room department. *International Journal of Production Economics, 141* (2), 646–658.

Stadtler, H. (2003). Multilevel lot sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. *Operations Research*, *51*(3), 487-502.

Stafford Jr, E. F., Tseng, F. T., & Gupta, J. N. (2005). Comparative evaluation of MILP flowshop models. *Journal of the Operational Research Society*, *56*(1), 88-101.

Tempelmeier, H., & Derstroff, M. (1996). A Lagrangean-based heuristic for dynamic multilevel multiitem constrained lotsizing with setup times. *Management Science*, *42*(5), 738-757.

Topaloglu, S. (2006). A multi-objective programming model for scheduling emergency medicine residents. *Computers & Industrial Engineering*, *51*(3), 375-388.

Tsai, P. F. J., & Teng, G. Y. (2014). A stochastic appointment scheduling system on multiple resources with dynamic call-in sequence and patient no-shows for an outpatient clinic. *European Journal of Operational Research*, *239*(2), 427-436.

Unlu, Y., & Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers & Industrial Engineering*, *58*(4), 785-800.

Wagelmans, A., Van Hoesel, S., & Kolen, A. (1992). Economic lot sizing: an O (n log n) algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research*, *40*(1-supplement-1), S145-S156.

Wagner, H. M., & Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management science*, *5*(1), 89-96.

Wang, J., & Fung, R. Y. (2014a). Adaptive dynamic programming algorithms for sequential appointment scheduling with patient preferences. *Artificial Intelligence in Medicine*, *63*(1), 33–40.

Wang, J., & Fung, R. Y. (2014b). An integer programming formulation for outpatient scheduling with patient preference. *Industrial Engineering & Management Systems, 13*(2), 193–202.

Wang, J., Fung, R. Y., & Chan, H. K. (2015). Dynamic appointment scheduling with patient preferences and choices. *Industrial Management & Data Systems, 115*(4), 700–717.

Wang, W. Y., & Gupta, D. (2011). Adaptive appointment systems with patient preferences. *Manufacturing & Service Operations Management*, *13*(3), 373–389.

Yu, S. H., & Hung, Y. F. (2016, December). Comparisons of three mixed integer programming models for parallel machine scheduling. In *2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (pp. 917-921). IEEE.

Zacharias, C., & Armony, M. (2016). Joint panel sizing and appointment scheduling in outpatient care. *Management Science*, *63*(11), 3978-3997.