Novel Approaches for Some Stochastic and Deterministic Scheduling Problems

Lingrui Liao

Dissertation submitted to the Faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Industrial and Systems Engineering

Subhash C. Sarin, Chair
Ebru K. Bish
Raghu Pasupathy
Hanif D. Sherali

June 10, 2011

Blacksburg, Virginia

Keywords: Stochastic scheduling, expectation-variance evaluation, conditional-value-at-risk, scenario generation, integrated lot-sizing and scheduling, high multiplicity asymmetric traveling salesman problem, column generation.

Novel Approaches for Some Stochastic and Deterministic Scheduling Problems

Lingrui Liao

(ABSTRACT)

In this dissertation, we develop novel approaches to independently address two issues that are commonly encountered in machine scheduling problems: uncertainty of problem parameters (in particular, due to job processing times), and batching of jobs for processing on capacitated machines.

Our approach to address the uncertainty issue regards the indeterminate parameters as random variables, and explicitly considers the resulting variability of a performance measure. To incorporate variability into the schedule selection process, we develop a method to evaluate both the expectation and variance of various performance measures for a given schedule. Our method is based on the use of mixture models to approximate a variety of distribution types. The Expectation-Maximization algorithm of Dempster et al. (1977) is applied to derive mixture models of processing time distributions. Our method, then, utilizes these mixture models to calculate the distributions of other random variables in order to derive the expectation and variance of various scheduling performance measures, assuming that the job sequencing decisions are known a priori. To make our method more computationally efficient, we adapt a mixture reduction method to control the number of mixture components used in the intermediate steps. We apply our method to two different scheduling problems: the job shop makespan scheduling problem and the single machine total weighted tardiness scheduling problem, and compare its performance with that of Monte-Carlo method. The results show the efficacy of our mixture approximation method. It generates fairly accurate results while requiring significantly less CPU times.

The proposed method offers a good compromise between the Monte Carlo method, which requires extensive effort, and use of simple normal approximation, which produces lower-quality results.

Next, we introduce and demonstrate for the first time in the literature the use of conditional-value-at-risk (CVaR) as a criterion for stochastic scheduling problems in order to obtain risk-averse solutions. This criterion has the tendency of minimizing both the expectation and variance of a performance measure simultaneously, which is an attractive feature in the scheduling area as most of the literature in this area considers the expectation and variance of a performance measure separately. Also, the CVaR has an added advantage of maintaining a linear objective function. We develop a scenario-based mixed integer programming formulation to minimize CVaR for the general scheduling problem involving various performance measures, and employ a decomposition-based approach for its solution. Furthermore, a set of valid inequalities are incorporated to strengthen the relaxed master problem of this decomposition scheme. The proposed approach is demonstrated on the single machine total weighted tardiness scheduling problem. Our computational investigation reveals the efficacy of the proposed decomposition approach and the effectiveness of using the CVaR as an optimization criterion for scheduling problems. Besides providing an exact approach to solve our stochastic scheduling problem, we also develop an efficient heuristic method to enable the use of CVaR for large-sized problems. To that end, we modify the Dynasearch method of Grosso et al. (2004) to minimize CVaR for a stochastic scheduling problem. Furthermore, we extend the application of CVaR to a parallel-machine total weighted tardiness problem. The use of CVaR appears to be quite promising for simultaneously controlling both the expected value and variability of a performance measure in a stochastic scheduling environment.

Scenario-based formulations have frequently been used for stochastic scheduling problems. However, the determination of a lower bound can be a time-consuming task for this approach. Next, we develop a new method for scenario generation that is

computationally competitive and that assures attainment of an exact lower bound. Our approach is based on discretization of random parameter distributions of job processing times. We use the idea of Recursive Stratified Sampling to partition the probability space, so that the conditional expectations in each region yield scenario-wise parameter values. These scenarios are, then, used to formulate a two-stage stochastic program, which yields a lower bound for the original stochastic problem. We provide theoretical basis of our bounding approach for both the expectation and CVaR objectives. Our discrete bounding method generates exact lower bounds, as against the probabilistic bounds generated by Sample Average Approximation. We also present results of our numerical experimentation to compare the performances of these two approaches in terms of the bound value obtained and the CPU time required.

The problem pertaining to integrated batching and scheduling of jobs on capacitated parallel machines that we consider arises in the primary manufacturing sector of a pharmaceutical supply chain. We, first, develop a comprehensive mathematical programming model that can accommodate various realistic features of this problem. These features include batch production, sequence-dependent setup time/cost, and inter-period carryover of setup status. We further derive several valid inequalities that are based on the embedded subproblem structure. We also consider an alternative formulation (termed the Plant Location model) based on the lot-sizing perspective of the problem. Noting the resemblance of the campaign sequencing subproblem to the high multiplicity asymmetric traveling salesman problem (HMATSP), we adapt various ideas from the HMATSP to enforce the connectivity of the sequencing graph. Due to the complexity of this problem, we also explore the possibility of applying column generation technique for its solution. Various schemes of problem decomposition are considered, along with the use of dual stabilization technique to improve the convergence of the column generation procedure. We also develop heuristic methods to generate initial feasible solutions that further enhance the performance of the column generation method. A computational experimentation has been

conducted on a data set that mimics real-life problem instances. It illustrates the effectiveness of using the proposed column generation method.

To my late grandfather Xiujun Liao, my father Taikang Liao,
and my mother Fengming Wu

Acknowledgements

First, I would like to sincerely thank my advisor, Dr. Subhash C. Sarin, for his guidance, patience and friendship during my stay at Virginia Tech. He has introduced me to various research areas; spent tremendous amount of effort on helping me develop research proficiency, and has provided most important directions and suggestions for this dissertation, which would not be possible without his mentorship.

I would also like to thank the other members of my advisory committee for their help and guidance in my graduate courses and research work. In particular, I would to thank Dr. Hanif D. Sherali for his guidance in mathematic programming models and solution techniques, Dr. Ebru K. Bish for introducing me to the area of stochastic programming, and Dr. Raghu Pasupathy for providing insightful perspectives on simulation and statistics.

I would like to thank all the faculty and staff in the ISE department for their caring and support during my stay. My gratitude goes on to the fellow graduate students as well, especially Dr. Cheng Guo, Ming Chen, and Jason Judd. Also, I shall not forget my wonderful roommates, Yanfeng Li, Juqi Liu, Song Xue, and Liugang Sheng in particular, for sparking my idea on the application of mixture models.

Last, but certainly not least, I would like to thank my parents for their constant encouragement, understanding and unconditional support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Scope of Research

Machine scheduling is an important category of decision problems that are commonly encountered in production as well as in other environments. Even though decades of research in machine scheduling has yielded a wealth of results for the solution of a variety of problems, yet there are practical issues that have not been addressed satisfactorily, a fact which hampers the application of these results in practice. Two issues stand out in the regard. One of these issues pertains to the stochastic nature of the practical environments. A typical approach to address the presence of a stochastic element in machine scheduling has been to optimize the expected value of a performance measure. However, such a schedule may perform poorly in practice if the associated performance measure has a large variability. It is, therefore, useful to have knowledge about the variability of the performance measure of a schedule, or better still, to determine a schedule that simultaneously optimizes both the expectation and variance of that measure. The second issue pertains to the treatment of a scheduling problem as an isolated problem. However, there are many instances (especially caused by shorter production cycles and intense competition) in which an organization manufactures a variety of products, and therefore,

finds scheduling to be intrinsically linked to the determination of production batches in the face of limited resources. In order to determine an effective solution for these instances, the scheduling problem must be solved in view of an integrated framework instead of as an isolated problem.

In this dissertation, we develop novel approaches to address both of the above mentioned issues for machine scheduling problems. First, we investigate both evaluation of a given schedule and determination of an optimal machine schedule when job processing times are assumed to be stochastic. Then, we consider an integrated lot-sizing and scheduling problem in the presence of parallel and limited resources.

In order to cope with uncertainty in a scheduling problem, we consider three different aspects. First, we address the basic question of how to evaluate the quality of a given schedule in an uncertain environment. Our approach models uncertain parameters as random variables. Consequently, any performance measure of interest is also a random variable, as it depends on the random parameters. Since the variability of a performance measure is of great importance to a risk-averse decision maker, both the expectation and variance of the performance measure need to be evaluated. Towards this goal, we develop a mixture model-based approach, which approximates the distributions of random parameters using mixtures of normal components. Given a group of g random variables $\{X_k\}$, $k = 1, \dots, g$, a new random variable X can be defined based on the value of X_Z, where Z is an independent random variable with a categorical distribution over the range $\{1, \dots, g\}$. Assuming continuous distributions for $\{X_k\}$, $k = 1, \dots, g$, the probability density function of X can be written as

$$f_X(x) = \sum_{k=1}^{g} p_k f_{X_k}(x),$$

where $p_k = P\{Z = k\}$, $\forall k = 1, \dots, g$, and $\sum_{k=1}^{g} p_k = 1$. Random variables $\{X_k\}$ are called components of the mixture, and $\{p_k\}$ are mixing proportions. We assume $f_{X_k}(x)$, $\forall k = 1, \dots, g$, to be a normal distribution. Having determined a mixture representation for the processing time of a job, we further derive mixture representations for intermediate

variables in the schedule, and eventually determine the expectation and variance of the performance measure based on the mixture distributions derived. Our approach benefits from a mixture model's versatility in representing different distribution types. However, the number of mixture components can grow rapidly with the number of jobs in a schedule. We utilize a mixture reduction method to control this growth in the number of components. It also helps in reducing computational effort required. Compared with the approaches of normal approximation and Monte Carlo method, our new approximation method is both accurate and computationally efficient. We have demonstrated its implementation on the job shop scheduling problem for the objective of minimizing the makespan and the single-machine total weighted tardiness problem.

Second, in order to simultaneously minimize the expectation and variance of a performance measure, we introduce the use of conditional-value-at-risk (CVaR) to the machine scheduling problems. For a given probability level $\alpha \in (0,1)$, the associated CVaR of a random variable can be regarded as the average value of its $(1-\alpha)\cdot 100\%$ largest outcomes. It is consistent with Second Degree Stochastic Dominance and can be incorporated into an optimization framework with a linear objective function. These attractive properties make CVaR a promising objective function for stochastic scheduling problems in order to simultaneously minimize the expected value and variability of a performance measure. We use a generic form of stochastic scheduling problem to derive a mixed-integer programming formulation for the minimization of CVaR. We demonstrate the efficacy of this approach through its application to the single-machine total weighted tardiness scheduling problem. Comparing its performance with the expected value objective, we show that the minimization of CVaR leads to a significant reduction in the variance of the performance measure with only a slight increment in expected value. In order to improve the computational effectiveness of our approach, we develop valid inequalities and a decomposition-based method for solving the generic formulation to optimality. In addition, we present a dynamic programming-based heuristic method to apply our approach to large-sized instances of the single-machine total weighted tardiness

problem. We have also demonstrated the use of CVaR for a more general parallel-machine total weighted tardiness problem.

The third aspect pertains to the use of scenario-based representation of random parameters to obtain optimal solutions for stochastic scheduling problems. In this approach, the original stochastic problem is converted into its *deterministic equivalent* where the stochasticity of parameters is represented by a fixed set of scenarios with associated probabilities. Each scenario represents a particular realization of all the parameter values, and accordingly, the optimization objective is formulated based on the scenario-wise outcomes of the performance measure. The most commonly used approach of scenario generation is Monte Carlo sampling, for which parameter values are fixed by sampling from their original distributions. Due to the randomness of sampling procedure, the optimal objective value of the resulting deterministic equivalent is also a random variable, and its expected value provides a lower bound for the original stochastic problem. Hence, the above sampling and solution procedure needs to be repeated multiple times so that an optimality gap for the original stochastic problem can be established with a given confidence level. This approach tends to be computationally expensive for scheduling problems, as the solution of a scenario-based model can take a long CPU time in each repetition. To overcome this difficulty, we develop an alternative scenario generation method that yields an exact lower bound with less computational effort. Our new method belongs to the realm of discrete bounding approximation, which applies a discrete distribution to approximate the original distribution of parameters. Such a discrete distribution is obtained by first partitioning the sample space of parameters into multiple regions, and then, taking the conditional expectation of parameters in each region. Due to Jensen's inequality, we are able to establish increasing convex ordering between the discrete distribution and the original one. Consequently, we apply this discrete distribution to formulate two scenario-based models, which yield exact lower bounds for the expected value and conditional-value-at-risk objectives, respectively. Unlike an existing method of discrete bounding, our new method generates the partition of sample space in a recursive

manner, and avoids the repeated solution of a scenario-based model. This allows us to obtain competitive bound values with less computational effort. We demonstrate the effectiveness of our method on the single-machine total weighted tardiness problem.

The integrated lot-sizing and scheduling problem, that we consider, arises in the operational control of primary pharmaceutical manufacturing facilities. This environment is characterized by a large number of product types and multiple flexible machines working in parallel. Furthermore, the production is implemented in batches, and a sequence-dependent setup time/cost is required between the batches of different product families. These features result in complex subproblem structures that resemble the capacitated lot-sizing problem and the high multiplicity asymmetric traveling salesman problem. First, we develop a mathematical programming formulation to accommodate the realistic features of this problem that include Bill-of-Material relations between the products and carryover setups. We investigate alternative formulations for the lot-sizing subproblem and the graph connectivity constraints associated with the sequencing subproblem. In addition, several types of valid inequalities are developed by exploiting the inherent problem structure in order to tighten the model formulation. These improvements significantly reduce the CPU time required to solve the problem. In order to solve large-sized instances of the problem, we consider the method of column generation, which decomposes the overall problem into more easily solvable subproblems. We compare several different ways for decomposing the problem and investigate further refinements of the column generation method, including the use of an advance-start solution and use of dual stabilization techniques. Our computational investigation on a testbed of data, which are generated to mimic the real-life operation of Boehringer-Ingelheim's primary pharmaceutical manufacturing facility in Petersburg, Virginia, reveals the efficacy of our column generation method.

1.2 Organization of Dissertation

The remainder of this dissertation is organized as follows. Chapters 2, 3, and 4 are focused on solving stochastic scheduling problems. In Chapter 2, we develop an approximation method to evaluate both the expectation and variance of various performance measures for a given schedule. Chapter 3 introduces the use of conditional-value-at-risk (CVaR) as a criterion for stochastic scheduling problems, and it presents both optimum-seeking and heuristic methods to solve single-machine and parallel-machine total weighted tardiness problems. Chapter 4 presents a new scenario generation method to determine an exact lower bound. We also compare its performance with other approaches with regard to the lower bound value obtained and the CPU time required when applied to the single-machine total weighted tardiness problem. In Chapter 5, we investigate an integrated lot-sizing and sequencing problem that arises in the primary manufacturing sector of a pharmaceutical supply chain. Various modeling approaches and solution methodologies are presented for this problem.

Due to the variety of problems included in this work, in principle, the scope of nomenclature is restricted to individual chapters. However, since Chapters 2, 3, and 4 all deal with issues in stochastic scheduling problems, we have tried to avoid conflicts of notation among these chapters.

Chapter 2

Determination of the Expectation and Variance of Scheduling Performance Measures Using Mixture Models

Deterministic machine scheduling problems have been studied extensively over several decades. However, as pointed out by McKay et al. (1988), the practical applicability of deterministic scheduling models is often hampered by the disregard of uncertainty issues, which are encountered frequently in practice. As a result, various approaches have been developed to address uncertainty in scheduling. *Robust scheduling* strives to provide the best protection against worst-case scenarios (Daniels and Kouvelis, 1995; Kouvelis, Daniels, and Vairaktarakis, 2000). *Reactive scheduling* modifies a baseline schedule after uncertainties are revealed during its implementation (Sabuncuoglu and Bayiz, 2000; Vieira, Herrmann, and Lin, 2003). *Fuzzy scheduling* is suitable for situations in which available data are insufficient to construct viable probabilistic models. It employs fuzzy numbers to represent uncertain parameters, such as job processing times (Balasubramanian and Grossmann, 2003). *Stochastic scheduling*, on the other hand, optimizes a performance measure assuming that the uncertain parameters are random variables with known distributions.

Since the value of a performance measure depends on problem parameters, randomness of these parameters implies that the performance measure is also a random variable. To compare performances of different schedules, some distributional property of the performance measure is usually adopted as the objective function to optimize. A commonly-used distributional property is the expected value, which can be regarded as the long-run average performance of a schedule (see Pinedo (2001) and Skutella and Uetz (2005), for example). As pointed out by Daniels and Kouvelis (1995), a critical disadvantage of optimizing the expectation of a performance measure is that it cannot accommodate the risk-averse attitude of a decision-maker. De, Ghosh, and Wells (1992) proposed to consider both the expectation and variance of a performance measure in a bicriterion mathematical programming framework and developed a methodology for the flow time performance measure. In this chapter, we also consider the expectation and variance of scheduling performance measures. However, we focus on the determination of their values for a given schedule, which is described by a complete job assignment and sequencing solution. The results obtained will facilitate the simultaneous optimization of expectation and variance.

Our method of determining the expectation and variance of a scheduling criterion is based on the use of mixture models (or mixture distributions) to represent random variables. The idea of a mixture distribution can be explained as follows. Consider a group of g random variables $\{X_k\}$, $k = 1, \dots, g$. A new random variable X is defined based on the value of X_Z, where the subscript Z is an independent random variable with a categorical distribution falling in the range $\{1, \dots, g\}$. In other words, to obtain a sample of X, we first take a sample of Z, and then, regard the value of X_Z as the value of X. Assuming continuous distributions for $\{X_k\}$, $k = 1, \dots, g$, the probability density function (PDF) of X can be written as

$$f_X(x) = \sum_{k=1}^{g} p_k f_{X_k}(x),$$

where $p_k = P(Z = k)$, $\forall k = 1, \dots, g$, and $\sum_{k=1}^{g} p_k = 1$. Random variables $\{X_k\}$ are called components of the mixture, and $\{p_k\}$ are called mixing proportions (or weights).

A mixture distribution can take various shapes. In fact, any continuous distribution can be approximated arbitrarily well by a finite mixture of normal densities with a common variance (McLachlan and Peel 2000). Due to the flexibility and simple structure, mixture distributions have been widely used in biology, medicine, economics and other fields to model various random variables. In our case, job processing times are assumed to be independent random variables with general distributions. We approximate these processing time distributions by using mixtures of normal components, and apply mixture models to determine the expectation and variance of performance measures.

The remainder of this chapter is organized as follows. In Section 2.1, we provide a literature review on the approximation problem and various mixture reduction methods. Section 2.2 is devoted to preliminaries for deriving mixture models for general processing time distributions and for calculating various operations using mixture distributions. Sections 2.3 and 2.4, then, contain applications of mixture models to approximate the expectation and variance of makespan for a job shop schedule and total weighted tardiness for a single-machine schedule, respectively.

2.1 Literature Review

To provide motivation and preparation for the work presented in this chapter, we first review the existing literature on the following two pertinent topics: variability of scheduling performance measures, and mixture reduction. We present a method to approximate a processing time distribution via a mixture distribution in Section 2.2. However, for approximating the distributions of random variables pertaining to a schedule, the number of components for a mixture representation can become quite large. It is, therefore, essential to reduce the number of mixture components. Mixture reduction is the process of reducing the number of these components.

2.1.1 Variability of Scheduling Performance Measures

In the machine scheduling literature, quite a few different approaches have been developed to control the variability of a schedule. Dodin (1996) defined a new concept of optimal sequence based on *Optimality Index*, which is the probability that a sequence yields the best objective value among all possible solutions. Portougal and Trietsch (1998) further defined two other optimality criteria, and suggested that variance reduction should be explicitly considered in the objective function. Ayhan and Olsen (2000) used heuristic procedures to minimize the variance of throughput time in a multi-class single server system. Their assumption of dynamic job arrival is different from that of a static machine scheduling problem, where a set of jobs is given a priori. However, all the above mentioned research results have indicated that scheduling variability is an important factor in decision-making.

With respect to the simultaneous minimization of expectation and variance in stochastic scheduling, Jung, Nagasawa, and Nishiyama (1990) considered the single-machine total flow time problem and provided a heuristic method to find non-dominated solutions. Nagasawa and Shing (1998) further extended their result to the parallel machine case and developed an interactive system to help a manager in selecting a schedule from non-dominated solutions. De, Ghosh, and Wells (1992) considered a different approach. Instead of trying to determine non-dominated solutions, they searched for expectation-variance efficient sequences for a single-machine flow time problem, where the efficiency is defined based on a linear combination of expectation and variance. Sarin et al. (2007) focused on the determination of expectation and variance of performance measures for a given schedule. Various completion time-based and due date-based scheduling criteria were considered. For some of the due date-based criteria, their results are based on the assumption of normally-distributed processing times.

2.1.2 Mixture Reduction

As noted earlier, mixture reduction is the process of reducing the number of components in

a mixture distribution while keeping the resulting distribution as close as possible to the original mixture. Applying mixture reduction to a complex mixture model tends to simplify subsequent calculations without significantly affecting the accuracy of the results. As such, mixture reduction has been a topic of interest in radar target tracking, pattern recognition, and other applications. Salmond (1988) proposed a Gaussian mixture reduction method, which preserves the mean and covariance of the original mixture while progressively merging components or groups of components. Williams (2003) regarded the reduction of a Gaussian mixture as an optimization problem. He applied an iterative optimization method to determine the parameters of the reduced mixture by minimizing the Integral Square Difference between the original mixture and the reduced one. Runnalls (2007) adopted Salmond (1988)'s progressive merging method, but used a difference criterion to choose the components to merge at each iteration. This new criterion was shown to be an upper bound for the Kullback-Leibler divergence between the original mixture and the reduced mixture. Nielsen and Nock (2009) investigated the clustering of multi-dimensional normal distributions. Their approach is based on information-theoretic divergence measures and k-means clustering method. Bruneau, Gelgon, and Picarougne (2010) applied Bayesian estimation to the problem of mixture reduction. Their approach is parsimonious in the sense that it reduces the mixture to the strictly necessary number of components.

2.2 Basic Operations to Implement a Mixture Model

Different performance measures for a scheduling problem can be regarded as functions of job processing times. We first present methods to approximate processing time distributions with mixture distributions, and to perform basic operations on these mixture distributions. These methods constitute building blocks for determining the distributions of other random variables pertaining to a schedule, and eventually, the expectation and variance of performance measures.

2.2.1 Mixture Approximation of Processing Time Distributions

Our mixture approximation method is based on the assumption that the job processing times are independent random variables, and their distributions are given at the outset. This enables us to approximate these processing time distributions individually, using mixtures of normal components. We denote the probability density function (PDF) of a normal distribution by $f(\mathbf{x}; \boldsymbol{\mu}, \mathbf{S})$, where $\boldsymbol{\mu}$ is the expectation vector and $\mathbf{S}$ is the covariance matrix. In the case of univariate normal, we have

$$f(x; \mu, \sigma^2) \equiv \frac{1}{\sqrt{2\pi} \cdot \sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

The PDF of a normal mixture (or Gaussian mixture) can be written as:

$$f_X(x) = \sum_{k=1}^{g} p_k f(x; \mu_k, \sigma_k^2).$$

where p_k, μ_k, and σ_k^2 are, respectively, the weight, expectation, and variance of the k^{th} normal component, $\forall k = 1, \cdots, g$.

There are different ways to fit a normal mixture model to a given set of random data. While the method of moments has been applied in the seminal work of Pearson (1894), it is less tractable for our application, which has more components in the mixture, and hence, more parameter values to estimate simultaneously. On the other hand, Dempster, Laird, and Rubin (1977), proposed the Expectation-Maximization (EM) algorithm to obtain Maximum Likelihood Estimators of the mixture parameters. This algorithm consists of two iterative steps, namely, the Expectation-step (E-step) and the Maximization-step (M-step). Given sample data along with values of the component parameters and the corresponding weights, the E-step estimates the posterior probability that each data point belongs to a particular mixture component. The M-step, then, updates parameter values according to their maximum likelihood estimators while regarding the posterior probability values as observed data. The EM algorithm converges monotonically if the model is identifiable; and it is relatively easy to implement when all the components of the mixture follow normal distributions. Hence, given the univariate processing time distribution of a job operation

and the number of components in the mixture, we can determine the Most Likelihood Estimators of p_k, μ_k, and σ_k^2 for all the components. Our overall procedure to achieve this goal can be divided into three phases:

I. Generate data points from the original distribution by using Monte Carlo sampling or other methods;

II. Use the EM algorithm to determine the mixture parameters;

III. Adjust the mixture parameter values to match the expectation and variance of the original distribution.

The EM algorithm used in Phase II is adapted from McLachlan and Peel (2000). We present it here for the sake of completeness.

EM-Normal Algorithm

Input: Sample data points $\{x_j\}$, $j = 1, \dots, N$.

Output: Mixture proportion (p_k), mean (μ_k) and standard deviation (σ_k) of component k, $\forall k = 1, \dots, g$.

Step 0. (Initialization)

Initialize the iteration counter $q = 0$. Calculate the sample mean and sample standard deviation of $\{x_j\}$, and denote them as $\bar{x}$ and s_N, respectively.

Let $p_k^{(0)} = g^{-1}$, $\sigma_k^{(0)} = s_N$, and determine the value of $\mu_k^{(0)}$ by sampling from $\mathcal{N}(\bar{x}, s_N^2)$, $\forall k = 1, \dots, g$.

Step 1. (E-step)

Calculate the posterior probability that sample point x_j comes from the k^{th} component as follows:

$$\tau_{k,j}^{(q)} = \frac{p_k^{(q)} f\left(x_j; \mu_k^{(q)}, \left(\sigma_k^{(q)}\right)^2\right)}{\sum_{h=1}^{g} p_h^{(q)} f\left(x_j; \mu_h^{(q)}, \left(\sigma_h^{(q)}\right)^2\right)}, \forall k = 1, \dots g; j = 1, \dots, N.$$

Step 2. (M-step)

Update the mixing proportions as

$$p_k^{(q+1)} = \frac{\sum_{j=1}^{N} \tau_{k,j}^{(q)}}{N}, \forall k = 1, \dots, g;$$

Update the component parameters as

$$\mu_k^{(q+1)} = \frac{\sum_{j=1}^{N} \tau_{k,j}^{(q)} \, x_j}{\sum_{j=1}^{N} \tau_{k,j}^{(q)}}, \forall k = 1, \dots, g,$$

$$\left(\sigma_k^{(q+1)}\right)^2 = \frac{\sum_{j=1}^{N} \tau_{k,j}^{(q)} \left(x_j - \mu_k^{(q)}\right)^2}{\sum_{j=1}^{N} \tau_{k,j}^{(q)}}, \forall k = 1, \dots, g.$$

Step 3. Calculate the log-likelihood value as

$$\log L\left(\mathbf{p}^{(q+1)}, \boldsymbol{\mu}^{(q+1)}, \boldsymbol{\sigma}^{(q+1)} \middle| \mathbf{x}\right) = \sum_{j=1}^{N} \log \sum_{k=1}^{g} p_k^{(q+1)} f\left(x_j; \mu_k^{(q+1)}, \left(\sigma_k^{(q+1)}\right)^2\right).$$

If the value converges, stop the algorithm with $p_k = p_k^{(q+1)}$, $\mu_k = \mu_k^{(q+1)}$, and $\sigma_k = \sigma_k^{(q+1)}$, $\forall k = 1, \cdots, g$; otherwise, let $q = q + 1$, and go to Step 1.

Note that the expectation and variance of the mixture distribution obtained in Phase II can be calculated as:

$$\mu_{\hat{X}} = \sum_{k=1}^{g} p_k \mu_k \,,$$

$$\sigma_{\hat{X}}^2 = \sum_{k=1}^{g} p_k \sigma_k^2 + \sum_{k=1}^{g} p_k \mu_k^2 - \mu_{\hat{X}}^2.$$

Suppose the expectation and variance of the original distribution are, respectively, μ_0 and σ_0^2. It is desirable that the corresponding mixture model also has the same expectation and variance values. To that end, we adjust the parameters in Phase III as follows:

$$\tilde{p}_k = p_k, \qquad \tilde{\mu}_k = \mu_0 + \mu_k - \mu_{\hat{X}},$$

$$\tilde{\sigma}_k^2 = \frac{\sigma_k^2}{\sum_{h=1}^{g} p_h \sigma_h^2} \left(\sigma_0^2 + \mu_{\hat{X}}^2 - \sum_{h=1}^{g} p_h \mu_h^2 \right), \forall k = 1, \dots, g.$$

It can be verified that

$$\mu_{\tilde{X}} = \sum_{k=1}^{g} \tilde{p}_k \tilde{\mu}_k = \mu_0,$$

$$\sigma_{\tilde{X}}^2 = \sum_{k=1}^{g} \tilde{p}_k \tilde{\sigma}_k^2 + \sum_{k=1}^{g} \tilde{p}_k \tilde{\mu}_k^2 - \mu_0^2 = \sigma_0^2.$$

These adjusted parameter values will be used in our approximation method to further determine the expectation and variance of various scheduling performance measures.

To illustrate the above procedure, we apply it to obtain mixture models for the following different processing time distributions:

$\mathcal{U}(2,8)$:	Continuous uniform distribution on the interval [2,8];
Exp(1/3):	Exponential distribution with the rate parameter $\lambda = 1/3$;
Beta(1,2):	Beta distribution with the shape parameters $\alpha = 1$ and $\beta = 2$;
Beta(5,2):	Beta distribution with the shape parameters $\alpha = 5$ and $\beta = 2$.

We used stratified pseudo-samples of the above distributions as input and obtained 5-component normal mixtures by using our approximation procedure. The density functions of the resulting distributions are plotted in Figure 2.1. The dashed lines and solid lines represent the original distributions and the obtained mixture distributions, respectively. Note that, our mixture representation for a given continuous distribution is similar in form to its Gaussian kernel density estimation (KDE) (Silverman, 1986):

$$f_{\tilde{X}}(x) = \sum_{i=1}^{k} \frac{1}{k\sigma} \varphi\left(\frac{x - X_i}{\sigma}\right) = \sum_{i=1}^{k} \frac{1}{k} f(x; X_i, \sigma^2),$$

where k is the number of samples and $\{X_i : i = 1, \dots, k\}$ are the sample values. Note that the KDE follows a mixture distribution with a common variance, σ^2, for all components. Although the derivation of KDE is different from our mixture representation, an analysis of its behavior may yield insights on the accuracy of mixture approximation. We provide such an analysis in Appendix A.

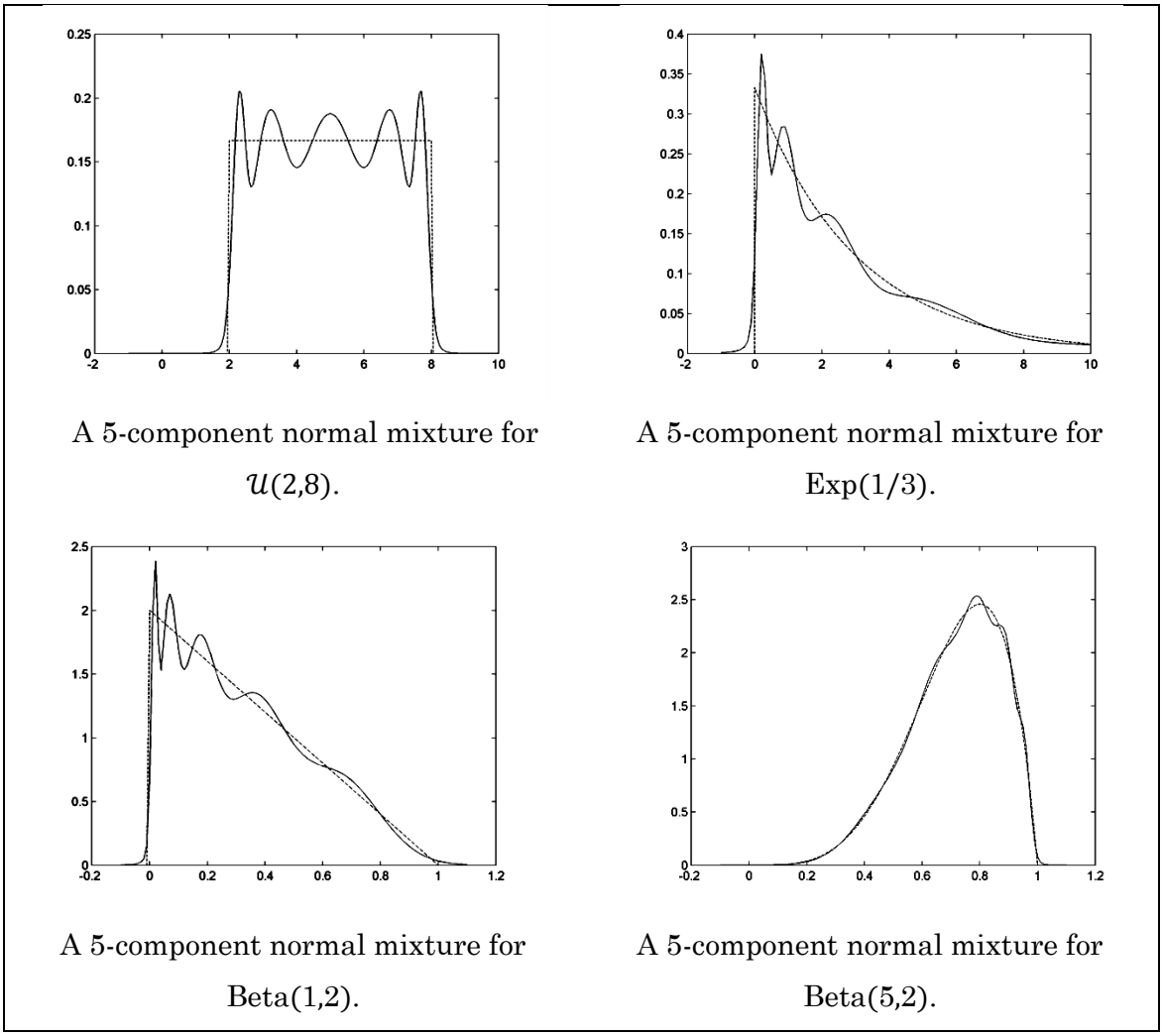


Figure 2.1: Probability density functions of various distributions and their mixture models

2.2.2 Summation of Mixture Distributions

Consider two independent random variables X and Y, expressed as mixture normal distributions:

$$X \sim f_X(x) = \sum_{k=1}^{g_X} p_{X_k} f(x; \mu_{X_k}, \sigma_{X_k}^2),$$

$$Y \sim f_Y(y) = \sum_{k=1}^{g_Y} p_{Y_k} f(y; \mu_{Y_k}, \sigma_{Y_k}^2).$$

The summation of these two variables $Z = X + Y$ also follows a mixture normal distribution:

$$Z \sim f_Z(x) = \sum_{k=1}^{g_Z} p_{Z_k} f(x; \mu_{Z_k}, \sigma_{Z_k}^2),$$

where

$$\begin{aligned} g_Z &= g_X \times g_Y, & p_{Z_k} &= p_{X_{k'}} \times p_{Y_{k''}}, \\ \mu_{Z_k} &= \mu_{X_{k'}} + \mu_{Y_{k''}}, & \sigma_{Z_k}^2 &= \sigma_{X_{k'}}^2 + \sigma_{Y_{k''}}^2, \\ k &= (k'-1)g_Y + k'', & \forall\ k' &= 1, \cdots, g_X;\ k'' = 1, \cdots, g_Y. \end{aligned}$$

2.2.3 Left-censoring of a Mixture Distribution

Given a random variable with a mixture normal distribution:

$$X \sim f_X(x) = \sum_{k=1}^{g} p_k f(x; \mu_{X_k}, \sigma_{X_k}^2),$$

the left-censored random variable $X^+ \equiv \max\{X, 0\}$ can be approximated by the following mixture distribution:

$$f_{X^+}(x) = \sum_{k=1}^{g} p_k f(x; \mu_{X_k^+}, \sigma_{X_k^+}^2),$$

where $X_k^+ \equiv \max\{X_k, 0\}$ is the k^{th} component of X^+, and we use normal density $f\left(x; \mu_{X_k^+}, \sigma_{X_k^+}^2\right)$ to approximate of its PDF. The mean and variance of X_k^+ can be determined by Clark (1961)'s formulae while regarding 0 as a degenerate normal variable:

$$\mu_{X_k^+} = \mu_{X_k} \Phi\left(\frac{\mu_{X_k}}{\sigma_{X_k}}\right) + \sigma_{X_k} \varphi\left(\frac{\mu_{X_k}}{\sigma_{X_k}}\right),$$

$$\sigma_{X_k^+}^2 = \left(\mu_{X_k}^2 + \sigma_{X_k}^2\right) \Phi\left(\frac{\mu_{X_k}}{\sigma_{X_k}}\right) + \mu_{X_k} \sigma_{X_k} \varphi\left(\frac{\mu_{X_k}}{\sigma_{X_k}}\right) - \mu_{X_k^+}^2,$$

where $\Phi(z)$ and $\varphi(z)$ are the cumulative distribution function and probability density function of the standard normal distribution, respectively. Note that, although the maximum of two normal random variables does not necessarily follow a normal distribution, Clark's formulae yield exact values of its expectation and variance. Hence, the following results hold analytically:

$$E[X^+] = \sum_{k=1}^{g} p_k \mu_{X_k^+};$$
$$Var[X^+] = \sum_{k=1}^{g} p_k \sigma_{X_k^+}^2 + \sum_{k=1}^{g} p_k \mu_{X_k^+}^2 - E[X^+]^2.$$

2.2.4 Mixture Reduction

Consider a multivariate normal mixture:

$$\mathbf{X} \sim f_{\mathbf{X}}(\mathbf{x}) = \sum_{k=1}^{g} p_k f_{\mathbf{X}_k}(\mathbf{x}) = \sum_{k=1}^{g} p_k f(\mathbf{x}; \boldsymbol{\mu}_k, \mathbf{S}_k),$$

where $\boldsymbol{\mu}_k$ and $\mathbf{S}_k$ are, respectively, the expectation vector and covariance matrix of its k^{th} component, $\mathbf{X}_k$.To reduce the number of components from g to h $(0 < h < g)$, an iterative joining (or merging) algorithm can be implemented by selecting two remaining components in each iteration, and merging them into one normal component. The merging algorithm stops after $(g - h)$ iterations. This approach was first proposed by Salmond (1988) for radar target tracking. The merging components, say $\mathbf{X}_i$ and $\mathbf{X}_j$, are chosen so that they have the shortest distance between them; and the distance measure is defined as:

$$D_s^2 \equiv \frac{p_i p_j}{p_i + p_j} (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T \mathbf{S}^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j),$$

where $\mathbf{S}$ is the covariance matrix of $\mathbf{X}$, which can be calculated as:

$$\mathbf{S} = \sum_{k=1}^{g} p_k \mathbf{S}_k + \sum_{k=1}^{g} p_k (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T.$$

The merger results in a new component, $\mathbf{X}_{i,j}$, with the following parameter values:

$$p_{i,j} = p_i + p_j,$$

$$\boldsymbol{\mu}_{i,j} = E[\mathbf{X}_{i,j}] = \frac{p_i \boldsymbol{\mu}_i + p_j \boldsymbol{\mu}_j}{p_{i,j}},$$

$$\mathbf{S}_{i,j} = Var[\mathbf{X}_{i,j}] = \frac{p_i \mathbf{S}_i + p_j \mathbf{S}_j}{p_{i,j}} + \frac{p_i \cdot p_j}{p_{i,j}^2}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T.$$

Note that the expectation and covariance of the mixture are kept the same during the merger.

Runnalls (2007) adopted the same iterative joining procedure, but used a different distance measure to choose the merging components. The distance between components $\mathbf{X}_i$ and $\mathbf{X}_j$ are calculated as:

$$D_r \equiv p_{i,j} \log\det(\mathbf{S}_{i,j}) - p_i \log\det(\mathbf{S}_i) - p_j \log\det(\mathbf{S}_j).$$

Runnalls showed that D_r is an upper bounding approximation of the Kullback-Leibler divergence, which measures the information-theoretic difference between the original mixture and the reduced mixture. Note that, although Kullback-Leibler divergence is a non-symmetric measure, the distance measure D_r is symmetric with respect to the interchange of $\mathbf{X}_i$ and $\mathbf{X}_j$.

2.3 Application to the Job Shop Makespan Minimization Problem

We consider the scheduling problem in a job shop environment, where n jobs are processed on m machines following their individual routes. The route of a job is defined by a sequence of operations, each of which is to be completed on a prescribed machine following the order of the sequence. A complete solution of a job shop problem specifies the processing sequences for operations on all the machines. We assume that such a solution is given a priori, and our goal is to determine the expectation and variance of the makespan, assuming independently distributed processing times for all job operations. Note that the

makespan of a job shop schedule is defined as the difference between the completion time of the last operation and the start time of the first operation.

Suppose that an operation of job t is to be performed in the j^{th} position on machine i. This operation, denoted as $O_{[i,j]}$, can only start after the completion of the previous operation of job t, and also, after the completion of the operation performed in the $(j-1)^{\text{th}}$ position on machine i (i.e., $O_{[i,j-1]}$). In order to determine the completion time distribution of $O_{[i,j]}$, we need the following information: (i) the distribution of its processing time, $P_{[i,j]}$; and (ii) the joint distribution of the completion time of the previous operation of job t, which we denote as $C_{[p,q]}$, and the completion time of $O_{[i,j-1]}$, denoted as $C_{[i,j-1]}$. Note that, we have already derived mixture models for job processing times in Section 2.2.1. If we can further represent the joint distribution of $C_{[p,q]}$ and $C_{[i,j-1]}$ by using a two-dimensional normal mixture distribution, the distribution of $C_{[i,j]}$ can be determined consequently. Based on this idea, we develop an iterative method to determine the join distribution of job completion times, which are, then, used to determine the expectation and variance of makespan.

At any time point in the schedule, we consider the set of latest operations, O_λ, which is defined as follows:

$$O_\lambda \equiv O_M \cup O_J,$$

where $O_M \equiv \{\text{the last operation scheduled on machine } i : i = 1, \dots, m\}$, and
$O_J \equiv \{\text{the last scheduled operation of job } t : t = 1, \dots, n\}$.

For example, consider a job shop with 3 machines and 3 jobs. The current partial schedule is depicted in Figure 2.2, where $O_{(x,y)}$ denotes the y^{th} operation of job x. According to our definition, we have:

$$\begin{aligned} O_M &= \{O_{(1,1)}, O_{(1,2)}, O_{(3,1)}\}, \\ O_J &= \{O_{(1,2)}, O_{(2,1)}, O_{(3,1)}\}, \\ O_\lambda &= \{O_{(1,1)}, O_{(1,2)}, O_{(3,1)}, O_{(2,1)}\}. \end{aligned}$$

Then, after scheduling the next operation, say, $O_{(3,2)}$, the updated sets of operations are

$$\begin{aligned} \hat{O}_M &= \{O_{(3,2)}, O_{(1,2)}, O_{(3,1)}\}, \\ \hat{O}_J &= \{O_{(1,2)}, O_{(2,1)}, O_{(3,2)}\}, \end{aligned}$$

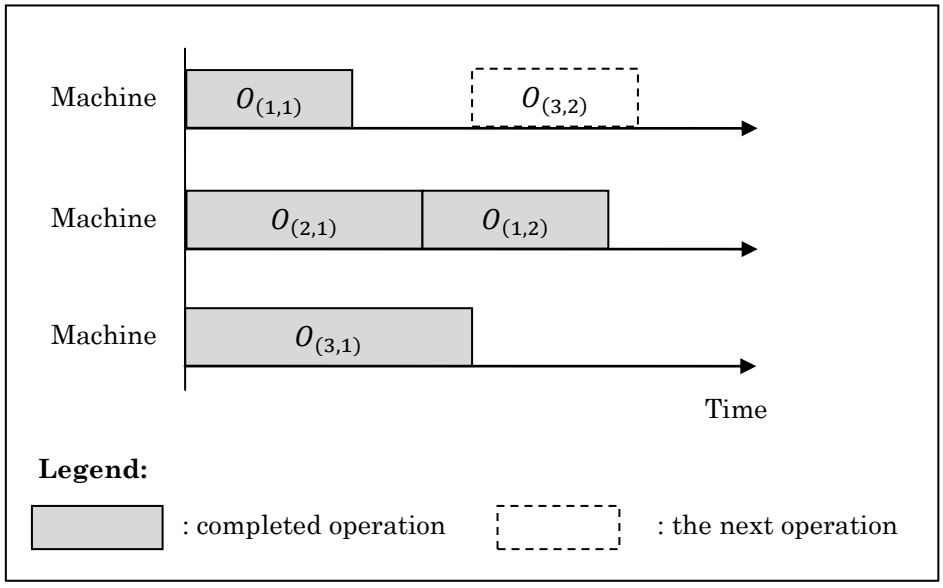


Figure 2.2: Sample of a job shop (partial) schedule

$$\hat{O}_\lambda = \{O_{(3,2)}, O_{(1,2)}, O_{(3,1)}, O_{(2,1)}\}.$$

Denoting the completion times of the operations in O_λ by $\mathbf{C}_\lambda$, our approximation method uses a multivariate normal mixture to represent the joint distribution of $\mathbf{C}_\lambda$, and updates its mixture representation whenever a new operation is added to the partial schedule. As an induction hypothesis, we assume that the joint distribution of $\mathbf{C}_\lambda$ can be expressed as:

$$f_\lambda(\mathbf{c}) = \sum_{k=1}^{g_\lambda} \pi_k^\lambda f(\mathbf{c}; \boldsymbol{\mu}_k^\lambda, \mathbf{S}_k^\lambda),$$

where $\mathbf{c}$ is a realization of random vector $\mathbf{C}_\lambda$, and π_k^λ, $\boldsymbol{\mu}_k^\lambda$ and $\mathbf{S}_k^\lambda$ are, respectively, the weight, the expectation vector and the covariance matrix of the k^{th} component. We also assume that the processing time of the next operation, $P_{[i,j]}$, follows a mixture distribution:

$$f_P(x) = \sum_{k=1}^{g_P} p_k f\left(x; \mu_k^P, \left(\sigma_k^P\right)^2\right).$$

After the addition of this operation, we obtain the updated set of latest operations as

$$\hat{O}_\lambda = \left(O_\lambda - \{O_{[p,q]}, O_{[i,j-1]}\}\right) \cup \{O_{[i,j]}\},$$

where $O_{[p,q]}$ is the previous operation of $O_{[i,j]}$ in the routing sequence, and $O_{[i,j-1]}$ is the last operation processed on machine i before the insertion of $O_{[i,j]}$. If $O_{[i,j]}$ is the first operation of job t, then $O_{[p,q]}$ represents the release time of job t. If $O_{[i,j]}$ is the first operation scheduled on machine i, (i.e., $j = 1$), then $O_{[i,j-1]}$ represents the ready time of machine i. The updated completion time vector, $\hat{\mathbf{C}}_\lambda$, follows the mixture distribution:

$$\hat{f}_\lambda(\hat{\mathbf{c}}) = \sum_{k=1}^{\hat{g}_\lambda} \hat{\pi}_k^\lambda f(\hat{\mathbf{c}}; \hat{\boldsymbol{\mu}}_k^\lambda, \hat{\mathbf{S}}_k^\lambda), \tag{2.1}$$

where $\hat{g}_\lambda = g_\lambda \times g_P$ and $\hat{\pi}_k^\lambda = \pi_{k'}^\lambda \times p_{k''}$, assuming that $\hat{\mathbf{C}}_\lambda^k$, the k^{th} normal component of $\hat{\mathbf{C}}_\lambda$, is determined according to the $(k')^{\text{th}}$ component of $\mathbf{C}_\lambda$ and the $(k'')^{\text{th}}$ component of $P_{[i,j]}$.

In order to determine $\hat{f}_\lambda(\hat{\mathbf{c}})$, we need the parameter values of $\hat{\mathbf{C}}_\lambda^k$ (i.e., $\hat{\boldsymbol{\mu}}_k^\lambda$ and $\hat{\mathbf{S}}_k^\lambda$), $\forall k = 1, \cdots, \hat{g}_\lambda$. Since most of the elements of $\hat{O}_\lambda$ come from O_λ, the determination of the updated expectation vector $\left(\hat{\boldsymbol{\mu}}_k^\lambda\right)$ only requires the calculation of $E\left[\hat{C}_{[i,j]}^k\right]$, which is the expected completion time of $O_{[i,j]}$ with respect to the k^{th} component of $\hat{\mathbf{C}}_\lambda$. Similarly, to determine the updated covariance matrix $\left(\hat{\mathbf{S}}_k^\lambda\right)$, we only need to determine the variance of $\hat{C}_{[i,j]}^k$ and its covariance with the other elements in $\hat{\mathbf{C}}_\lambda^k$.

Note that, $\hat{C}_{[i,j]}^k = \hat{S}_{[i,j]}^{k'} + P_{[i,j]}^{k''}$, where start time $\hat{S}_{[i,j]}^{k'} \equiv \max\left\{C_{[i,j-1]}^{k'}, C_{[p,q]}^{k'}\right\}$. Since the processing time, $P_{[i,j]}$, is independent of the completion times of previous operations, the expectation and variance of $\hat{C}_{[i,j]}^k$ can be determined as:

$$E\left[\hat{C}_{[i,j]}^k\right] = E\left[\hat{S}_{[i,j]}^{k'}\right] + E\left[P_{[i,j]}^{k''}\right],$$
$$Var\left[\hat{C}_{[i,j]}^k\right] = Var\left[\hat{S}_{[i,j]}^{k'}\right] + Var\left[P_{[i,j]}^{k''}\right].$$

where $E\left[P_{[i,j]}^{k''}\right] = \mu_{k''}^P$ and $Var\left[P_{[i,j]}^{k''}\right] = \left(\sigma_{k''}^P\right)^2$. Since the joint distribution of $C_{[i,j-1]}^{k'}$ and $C_{[p,q]}^{k'}$ can be directly extracted from $\boldsymbol{\mu}_{k'}^\lambda$ and $\mathbf{S}_{k'}^\lambda$, the expectation and variance of $\hat{S}_{[i,j]}^{k'}$ can be determined by using Clark (1961)'s formulae, which also provide the covariance between $\hat{S}_{[i,j]}^{k'}$ and the elements of $\hat{\mathbf{C}}_\lambda^k$. The covariance between $\hat{C}_{[i,j]}^k$ and the other elements of $\hat{\mathbf{C}}_\lambda^k$ can be determined consequently, noting that $P_{[i,j]}^{k''}$ is independent of $\hat{\mathbf{C}}_\lambda^k$. This completes the

calculation of the parameters of $\hat{\mathbf{C}}_\lambda^k$.

We repeat the above procedure for each new operation scheduled on the machines. When a full schedule is obtained in the end, O_λ consists of the last operations of all the jobs and the last operations on all the machines. Let l be the corresponding cardinality of O_λ. The distribution of $\mathbf{C}_\lambda$ is represented as an l-dimensional normal mixture. We derive the expectation and variance of the makespan as follows.

We focus on the k^{th} component, $\mathbf{C}_\lambda^k, \forall k = 1, \dots, g_\lambda$. The makespan corresponding to this component is

$$C_{\max}^k = \max\left\{\left(C_\lambda^k\right)_1, \dots, \left(C_\lambda^k\right)_l\right\},$$

where $\left(C_\lambda^k\right)_i$ denotes the i^{th} element of vector $\mathbf{C}_\lambda^k$, $\forall i = 1, \dots, l$. Since $\mathbf{C}_\lambda^k$ follows an l-dimensional normal distribution, the expectation and variance of $C_{\max}^k$ can be determined as follows:

$$\begin{aligned} C_{\max}^k &= \max\left\{\left(C_\lambda^k\right)_1, \dots, \left(C_\lambda^k\right)_l\right\} \\ &= \max\left\{\max\left\{\dots \max\left\{\left(C_\lambda^k\right)_1, \left(C_\lambda^k\right)_2\right\}, \dots\right\}, \left(C_\lambda^k\right)_l\right\}. \end{aligned}$$

In particular, we progressively combine the elements of $\mathbf{C}_\lambda^k$ using the maximum operator. For each maximum that we take, the two operands are assumed to follow a joint normal distribution. To begin with, the joint distribution of $\left(C_\lambda^k\right)_1$ and $\left(C_\lambda^k\right)_2$ can be determined by projecting the distribution of $\mathbf{C}_\lambda^k$ on its first two elements. Given this joint normal distribution, the maximum, $F_2^k \equiv \max\left\{\left(C_\lambda^k\right)_1, \left(C_\lambda^k\right)_2\right\}$, can be approximated by a normal distribution, whose parameters are determined by using Clark (1961)'s formulae. Furthermore, Clark's formulae also give the covariance between F_2^k and $\left(C_\lambda^k\right)_i$, $\forall i = 3, \dots, l$. Therefore, we can iteratively determine the distribution of

$$F_i^k \equiv \max\left\{\left(C_\lambda^k\right)_1, \dots, \left(C_\lambda^k\right)_i\right\} = \max\left\{F_{i-1}^k, \left(C_\lambda^k\right)_i\right\}, \forall i = 3, \dots, l.$$

The end result yields $F_l^k \equiv C_{\max}^k$. Hence, we have determined the makespan according to the k^{th} component of $\mathbf{C}_\lambda$. Finally, according to the Laws of Total Expectation and Total Variation, the expectation and variance of the makespan, $C_{\max}$, are calculated as:

$$E[C_{\max}] = \sum_{k=1}^{g_\lambda} p_k E[C_{\max}^k]\,; \tag{2.2}$$

$$Var[C_{\max}] = \sum_{k=1}^{g_\lambda} p_k Var[C_{\max}^k] + \sum_{k=1}^{g} p_k E[C_{\max}^k]^2 - E[C_{\max}]^2. \tag{2.3}$$

Our approximation method can be summarized as the following algorithm.

Mixture-Job-Shop Algorithm

Notation:

Ω: The set of unscheduled operations;

ξ: Prescribed sequences of operations on all the machines.

Step 0. Let $\Omega = \{\text{All operations}\}$, $O_\lambda = \{\text{Job release times, machine ready times}\}$.

Step 1. Select an operation $O_{[i,j]}$ from Ω such that $O_{[i,j]}$ is ready to be processed according to ξ. If no such an operation exists, go to Step 4.

Step 2. Let $\hat{O}_\lambda = \left(O_\lambda - \{O_{[p,q]}, O_{[i,j-1]}\}\right) \cup \{O_{[i,j]}\}$. Calculate the mixture distribution of $\hat{\mathbf{C}}_\lambda$ according to the distributions of $\mathbf{C}_\lambda$ and $P_{[i,j]}$.

Step 3. Let $\Omega = \Omega - \{O_{[i,j]}\}$, $O_\lambda = \hat{O}_\lambda$. Go to Step 1.

Step 4. Calculate the expectation and variance of makespan ($E[C_{\max}^k]$ and $Var[C_{\max}^k]$) according to the k^{th} component of $\mathbf{C}_\lambda$, $\forall k = 1, \ldots, g_\lambda$.

Step 5. Determine the values of $E[C_{\max}]$ and $Var[C_{\max}]$ by using Equations (2.2) and (2.3).

Note that, in view of Equation (2.1), the number of components in the mixture distribution of $\mathbf{C}_\lambda$ increases exponentially with the number of scheduled operations. To make our approximation method more computationally efficient, we incorporate the mixture reduction methods discussed in Section 2.2.4 to control the number of mixture components. More specifically, Step 2 of Mixture-Job-Shop algorithm is modified as follow:

Step 2. Let $\hat{O}_\lambda = \left(O_\lambda - \{O_{[p,q]}, O_{[i,j-1]}\}\right) \cup \{O_{[i,j]}\}$. Calculate the mixture distribution of $\hat{\mathbf{C}}_\lambda$ according to the distributions of $\mathbf{C}_\lambda$ and $P_{[i,j]}$. If the number of components in $\hat{\mathbf{C}}_\lambda$

exceeds the prescribed value of h, apply a reduction algorithm on its mixture distribution to reduce the number of components to h.

Next, we conducted a numerical experimentation to demonstrate the efficacy of our method. Consider a job shop problem with 3 jobs and 2 machines to minimize makespan. The problem data is presented in Tables 2.1, 2.2 and 2.3.

Table 2.1: Processing time distributions of job operations

	Job 1	Job 2	Job 3
Machine 1	$\mathcal{U}[3,6]$	$\mathcal{U}[2,7]$	$\mathcal{U}[9,17]$
Machine 2	$\mathcal{U}[6,7]$	$\mathcal{U}[5,11]$	$\mathcal{U}[1,8]$

Table 2.2: Routings of jobs

Job	Routing (machine numbers)	
Job 1	2	1
Job 2	1	2
Job 3	1	2

Table 2.3: Machine processing sequences

Machines	Processing Sequence (job numbers)		
Machine 1	2	3	1
Machine 2	1	2	3

To provide a point of reference, we used Monte Carlo simulation with 100,000 replications to estimate the expectation and variance of makespan. In our approximation method, we represent the processing time of each operation with a mixture of 5 normal components. The control parameter associated with the mixture reduction algorithm is set to be $h = 10$. Besides using mixture approximation, we also considered simple normal

approximation, which is a special case of our mixture approximation method with only one component in each mixture model. The results are presented in Table 2.4.

The results obtained with our mixture approximation method (without mixture reduction) closely matched the simulation results, while normal approximation generated less accurate values. When mixture reduction is applied, the approximation accuracy is not significantly affected, and there is no significant difference between the expectation and variance values obtained for each of the three different reduction approaches used. However, Williams's approach required a much longer CPU time than the other two methods. By incorporating Salmond's or Runnalls's approach of mixture reduction into the Mixture-Job-Shop algorithm, we are able to significantly reduce the CPU time while obtaining good approximation accuracy.

Table 2.4: Results for a job shop problem to minimize makespan

Method		$E[C_{\max}]$	$Var[C_{\max}]$	CPU Time* (seconds)
Monte Carlo Simulation		23.145	7.891	97.58
Normal Approximation		23.111	8.138	Negligible
Mixture Approximation	Without Reduction	23.141	7.904	21.89
	Salmond's Approach	23.138	7.962	0.28
	Williams's Approach	23.155	8.023	374.30
	Runnalls's Approach	23.143	7.998	0.31

* CPU times are obtained by using MATLAB 7.6 on a computer with a 2.66 GHz Core™2 Duo CPU and 2 GByte memory.

2.4 Application to the Single-Machine Total Weighted Tardiness Problem

The tardiness of job j in a scheduling problem is defined as

$$T_j \equiv \max\{C_j - d_j, 0\},$$

where C_j and d_j are, respectively, the completion time and due date of job j.

In this section, we approximate the expectation and variance of the total weighted tardiness for a single-machine scheduling problem. The performance measure of our interest is defined as

$$\text{TWT} \equiv \sum_{j=1}^{n} w_j T_j,$$

where w_j is the importance factor (or weight) of job j, $\forall j = 1, \cdots, n$. We assume that job processing times, $\{P_j\}$, are independently distributed random variables, and a processing sequence has been given. Note that, if we use subscript "$[j]$" to denote the j^{th} job in the processing sequence, the performance measure can be rewritten as:

$$\text{TWT} \equiv \sum_{j=1}^{n} w_j T_j = \sum_{j=1}^{n} w_{[j]} T_{[j]}.$$

Therefore, the expectation and variance of total weighted tardiness are

$$E[\text{TWT}] = E\left[\sum_{j=1}^{n} w_{[j]} T_{[j]}\right] = \sum_{j=1}^{n} w_{[j]} \mu_{T[j]}, \tag{2.4}$$

$$Var[\text{TWT}] = Var\left[\sum_{j=1}^{n} w_{[j]} T_{[j]}\right] = \sum_{j=1}^{n} w_{[j]}^2 \sigma_{T[j]}^2 + 2\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{[i]} w_{[j]} \sigma_{T[ij]}, \tag{2.5}$$

where $\mu_{T[j]} = E\left[T_{[j]}\right]$, $\sigma_{T[j]}^2 = Var\left[T_{[j]}\right]$ and $\sigma_{T[ij]} = Cov\left[T_{[i]}, T_{[j]}\right]$.

2.4.1 Approximation Method

Sarin et al. (2007) have used normal distributions to approximate random job processing times and to determine the expectation and variance of total weighted tardiness. We improve upon their approach and use mixtures of normal distributions to represent job processing times. Since the mixture components are normal random variables, we apply the results of Sarin et al. (2007) in component-wise calculations.

Our new approximation method for expectation and variance of total weighted tardiness can be divided into three phrases:

I. Calculate expectation and variance of job tardiness, $T_{[j]}$;

II. Calculate covariance of tardiness variables, $Cov[T_{[i]}, T_{[j]}]$;

III. Calculate the expectation and variance of total weighted tardiness according to Equations (2.4) and (2.5).

In Phase I, we calculate the mixture distributions of job completion times iteratively. As an inductive hypothesis, we assume that the completion time of the j^{th} job ($C_{[j]}$) is represented by a normal mixture:

$$f_{C_{[j]}}(x) = \sum_{k=1}^{g^C_{[j]}} \pi^k_{[j]} f_{C^k_{[j]}}(x),$$

where $C^k_{[j]}$ is the k^{th} normal component with associated weight $\pi^k_{[j]}$, $\forall k = 1, \cdots, g^C_{[j]}$. The mixture representation of processing time $P_{[j+1]}$ has been determined in Section 2.2.1 as follows:

$$f_{P_{[j+1]}}(x) = \sum_{k=1}^{g^P_{[j+1]}} p^k_{[j+1]} f_{P^k_{[j+1]}}(x).$$

Since $C_{[j+1]} = C_{[j]} + P_{[j+1]}$, the recursive expression for the PDF of $C_{[j+1]}$ can be determined, using the method developed in Section 2.2.2, as follows:

$$f_{C_{[j+1]}}(x) = \textstyle\sum_{k=1}^{g^C_{[j+1]}} \pi^k_{[j+1]} f_{C^k_{[j+1]}}(x)\ ,$$

where

$$\begin{aligned} g^C_{[j+1]} &= g^C_{[j]} \times g^P_{[j+1]}\ , \quad \pi^k_{[j+1]} = \pi^{k'}_{[j]} \times p^{k''}_{[j+1]}\ , \\ C^k_{[j+1]} &= C^{k'}_{[j]} + P^{k''}_{[j+1]}\ , \qquad k = (k'-1) g^P_{[j+1]} + k''\ , \\ \forall k' &= 1, \dots, g^C_{[j]}\ ; \qquad k'' = 1, \dots, g^P_{[j+1]}. \end{aligned}$$

If the resulting number of components, $g^C_{[j+1]}$, exceeds a prescribed upper limit of h, the current mixture representation of $C_{[j+1]}$ is reduced to a h-component normal mixture using Runnalls' mixture reduction algorithm.

Since $d_{[j]}$ is a given constant, job lateness, $L_{[j]} \equiv C_{[j]} - d_{[j]}$, also follows a mixture normal distribution:

$$f_{L_{[j]}}(x) = \sum_{k=1}^{g_{[j]}^{C}} \pi_{[j]}^{k} f_{L_{[j]}^{k}}(x),$$

with $E\left[L_{[j]}^{k}\right] = E\left[C_{[j]}^{k}\right] - d_{[j]}$ and $Var\left[L_{[j]}^{k}\right] = Var\left[C_{[j]}^{k}\right]$. Note that $T_{[j]} = \min\{C_{[j]} - d_{[j]}, 0\} = \min\{L_{[j]}, 0\} = L_{[j]}^{+}$. Hence, we can apply the method developed in Section 2.2.3 to determine the expectation and variance of $T_{[j]}$.

In phase II of the approximation method, we define $C_{\langle i,j\rangle} \equiv C_{[j]} - C_{[i]}$ and apply the following results from Sarin et al. (2007):

$$E\left[T_{[i]}T_{[j]}\right] = E\left[\max(X - d_{[i]}, 0) \cdot \max(Y + X - d_{[j]})\right]$$
$$= \int_{x=d_{[i]}}^{\infty} \int_{y=d_{[j]}-x}^{\infty} \left(x - d_{[i]}\right) \cdot \left(y + x - d_{[j]}\right) \cdot f_Y(y) \cdot f_X(x) \mathrm{d}y \, \mathrm{d}x, \qquad (2.6)$$

$$\forall 1 \le i < j \le n$$

where $X = C_{[i]}$ and $Y = C_{\langle i,j\rangle}$. The mixture presentation of X has been determined in Phase I. Since $Y = \sum_{k=i+1}^{j} P_{[k]}$, mixture representations of Y can be determined using an approach similar to the calculation of $C_{[j]} = \sum_{k=1}^{j} P_{[k]}$. Note that the relations depicted in Figure 2.3 can be used to arrange the calculations. Whenever the number of mixture components for $C_{\langle i,j\rangle}$ exceeds h, mixture reduction is applied to compact the model.

Note that, when $(j - i)$ is a large number, $C_{\langle i,j\rangle}$ tends to follow a normal distribution due to the Central Limit Theorem. In that case, we can further reduce computational effort by regarding $C_{\langle i,j\rangle}$ as a normal random variable, and determine its parameters as follows:

$$E\left[C_{\langle i,j\rangle}\right] = \sum_{k=i+1}^{j} E\left[P_{[k]}\right], \qquad Var\left[C_{\langle i,j\rangle}\right] = \sum_{k=i+1}^{j} Var\left[P_{[k]}\right], \qquad \forall 1 \le i < j \le n.$$

Having determined the mixture representations for both $X = C_{[i]}$ and $Y = C_{\langle i,j\rangle}$, we have

$$E\left[T_{[i]}T_{[j]}\right] = \sum_{k'=1}^{g^X} \sum_{k''=1}^{g^Y} \pi_X^{k'} \pi_Y^{k''} E[\max\left(X^{k'} - d_{[i]}, 0\right) \cdot \max(X^{k'} + Y^{k''} - d_{[j]}, 0)],$$

where g^X and g^Y are, respectively, the numbers of components for X and Y. Since the components $X^{k'}$ and $Y^{k''}$ are normal random variables, we apply the second part of Equation (2.6) by assuming $X = X^{k'}$ and $Y = Y^{k''}$, and then, numerically calculate the integration using the approach discussed in Sarin et al. (2007). Consequently, we have

$$Cov\left[T_{[i]}T_{[j]}\right] = E\left[T_{[i]}T_{[j]}\right] - E\left[T_{[i]}\right] \cdot E\left[T_{[j]}\right].$$

Phase III of the approximation method is a direct application of Equations (2.4) and (2.5). Note that, the incorporation of mixture reduction in the first two phases makes our new approximation procedure more efficient for large-sized problems. Next, we provide results of our numerical experimentation, and compare the performance of our mixture approximation method with that of Monte Carlo method.

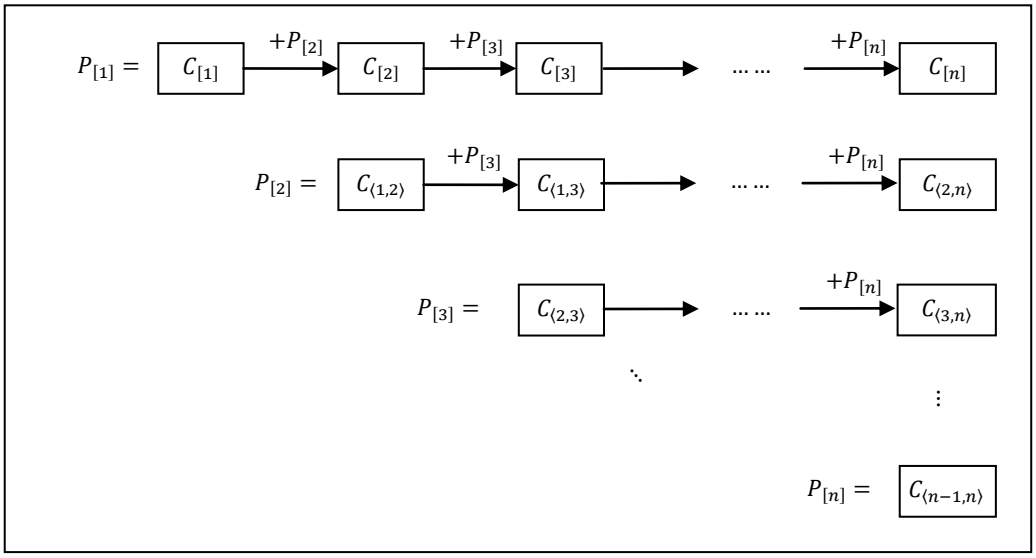


Figure 2.3: Calculation of $C_{[i]}$ and $C_{\langle i,j\rangle}$

2.4.2 Numerical Experimentation

Since it is impractical to determine analytical values of expectation and variance of total weighted tardiness, we compare the results of our approximation algorithm with those

obtained using Monte-Carlo simulation. We considered test problems with 25 jobs. For each problem instance, one of the following options is used to determine the processing time distributions:

1) $P_i \sim \mathcal{U}(a_i, b_i)$

   Fix values of μ_i and r_i by sampling from $\mathcal{U}(3,20)$ and $\mathcal{U}(0,1)$, respectively.

   Let $a_i = r_i \cdot \mu_i$, $b_i = 2\mu_i - a_i$.Note that $E[P_i] = \frac{a_i + b_i}{2} = \mu_i$.

2) $P_i \sim \text{Exp}(\lambda_i)$

   Determine values of μ_i by sampling from $\mathcal{U}(3,20)$. Let $\lambda_i = 1/\mu_i$.

3) $P_i \sim a_i + (b_i - a_i) \cdot \text{Beta}(\alpha, \beta)$, with $\alpha = 1, \beta = 2$

   Fix values of μ_i and r_i by sampling from $\mathcal{U}(3,20)$ and $\mathcal{U}(0,1)$, respectively.

   Let $a_i = r_i \cdot \mu_i$, $b_i = \mu_i + \frac{\beta}{\alpha}(\mu_i - a_i)$.

   Note that $E[P_i] = a_i + \frac{\alpha}{\alpha+\beta}(b_i - a_i) = \frac{\alpha}{\alpha+\beta} b_i + \frac{\beta}{\alpha+\beta} a_i = \mu_i$

4) $P_i \sim a_i + (b_i - a_i) \cdot \text{Beta}(\alpha, \beta)$, with $\alpha = 5, \beta = 2$.

Furthermore, job due dates are generated as follows (Koulamas, 1996):

$$\mathcal{U}\left(1 - \text{TF} - \frac{\text{RDD}}{2}, 1 - \text{TF} + \frac{\text{RDD}}{2}\right) \cdot \sum_{i=1}^{n} \mu_i ,$$

where the tardiness factor $\text{TF} = 0.8$, and the relative due date $\text{RDD} = 0.4$. The weights of jobs are generated from a discrete uniform distribution over the range of $\{1, \dots, 5\}$. We also generated a job sequence independently from the above procedure. It is used for all the following test runs.

Once a problem instance is fixed, a set of K random scenarios are generated using the given processing time distributions. The value of total weighted tardiness (TWT) is calculated and recorded as τ_i for each scenario i, $\forall i = 1, \dots, K$. Sample mean (μ_τ) and sample variance (σ_τ^2) of $\{\tau_i\}$ are recorded as the values of $E[\text{TWT}]$ and $Var[\text{TWT}]$ for the Monte Carlo simulation approach. We also calculate 95% confidence intervals (CI) as follows:

$$E[\text{TWT}] \sim \left(\mu_\tau - t_{n-1,0.975}\frac{\sigma_\tau}{\sqrt{n}}, \mu_\tau + t_{n-1,0.975}\frac{\sigma_\tau}{\sqrt{n}}\right),$$

$Var[\text{TWT}] \sim \left(\frac{n-1}{\chi^2_{n-1,0.975}}\sigma_\tau^2, \frac{n-1}{\chi^2_{n-1,0.975}}\sigma_\tau^2\right)$.

As the first step, a sample size of $K = 100{,}000$ is used to establish relatively high quality estimates. We, then, apply our mixture approximation algorithm with the following parameter settings:

1) Number of mixture components for processing time distributions : $g = \{1,3,5\}$;
2) Upper limit on the number of components for completion times : $h = g$;
3) Threshold value of $(j - i)$ beyond which $C_{\langle i,j\rangle}$ is assumed to be normal: 10.

Since the sample size of $K = 100{,}000$ tends to induce long CPU times, we also considered simulation with a sample size of $K = 10{,}000$ for performance comparison with our approximation algorithm. For each of the four options of determining processing time distributions, 30 problem instances were generated to test the performances of different approaches. We considered the expectation and variance values obtained from simulation of 100,000 scenarios as the baseline values. Relative deviations from these baseline values were calculated for the other approaches. The average values of relative deviations over 30 problem instances are presented in Table 2.5. Our experimentation was conducted using MATLAB 7.0 on a Pentium D 3.2 GHz computer with 2 GByte memory.

Note that our approximation algorithm takes significantly less CPU time than that required by Monte Carlo simulation with 10,000 scenarios, while giving better values of $E[\text{TWT}]$ and $Var[\text{TWT}]$. With $g = 1$, our mixture approximation method is equivalent to the normal approximation method of Sarin et al. (2007). Although, normal approximation yields acceptably accurate results for the Uniform and transformed Beta distributions, this is not the case for the Exponential distribution. By using our mixture approximation method, the accuracy of results tends to improve as g increases. Since the baseline values of expectation and variance are obtained from sample estimates, it is not unusual to encounter some variation when the values of relative deviations are small. Therefore, the relative deviations are not strictly decreasing with an increment in g for all the cases. However, better results are obtained for $g = 5$ in general. The results for Exponential distributions

(30 problem instances) are further illustrated in Figures 2.4 and 2.5 for expectation and variance, respectively. Confidence intervals, which are calculated based on Monte Carlo simulation with 100,000 scenarios, are depicted with dashed lines. It is clear that the quality of mixture approximation improves with an increment in the number of components. The use of normal approximation (i.e., $g = 1$), tends to induce a negative bias in the approximated value of variance. This is caused by the long tails of exponential distributions, which induce a bigger variation in total weighted tardiness. Approximating an exponential processing time with a normal distribution tends to underestimate variance of tardiness. However, with a 5-component mixture distribution (i.e., $g = 5$), the approximations for both variance and expectation values are superior.

Table 2.5: Comparison of results obtained using mixture approximation and Monte Carlo method

Distribution		Uniform			Exponential		
Measure		E[TWT]	Var[TWT]	Time (sec)	E[TWT]	Var[TWT]	Time (sec)
Mixture Approximation	$g = 1$	0.041%	0.308%	0.17	0.253%	5.304%	0.18
	$g = 3$	0.039%	0.306%	1.29	0.148%	1.959%	1.33
	$g = 5$	0.039%	0.314%	4.53	0.122%	0.847%	4.67
Simulation ($K = 10{,}000$)		0.092%	1.131%	31.12	0.301%	1.355%	30.35
Distribution		Beta(1,2) Transformed			Beta(5,2) Transformed		
Measure		E[TWT]	Var[TWT]	Time (sec)	E[TWT]	Var[TWT]	Time (sec)
Mixture Approximation	$g = 1$	0.043%	0.834%	0.17	0.013%	0.436%	0.20
	$g = 3$	0.040%	0.407%	1.29	0.012%	0.355%	1.36
	$g = 5$	0.039%	0.348%	4.54	0.012%	0.361%	4.68
Simulation ($K = 10{,}000$)		0.097%	1.030%	31.32	0.033%	1.064%	31.14

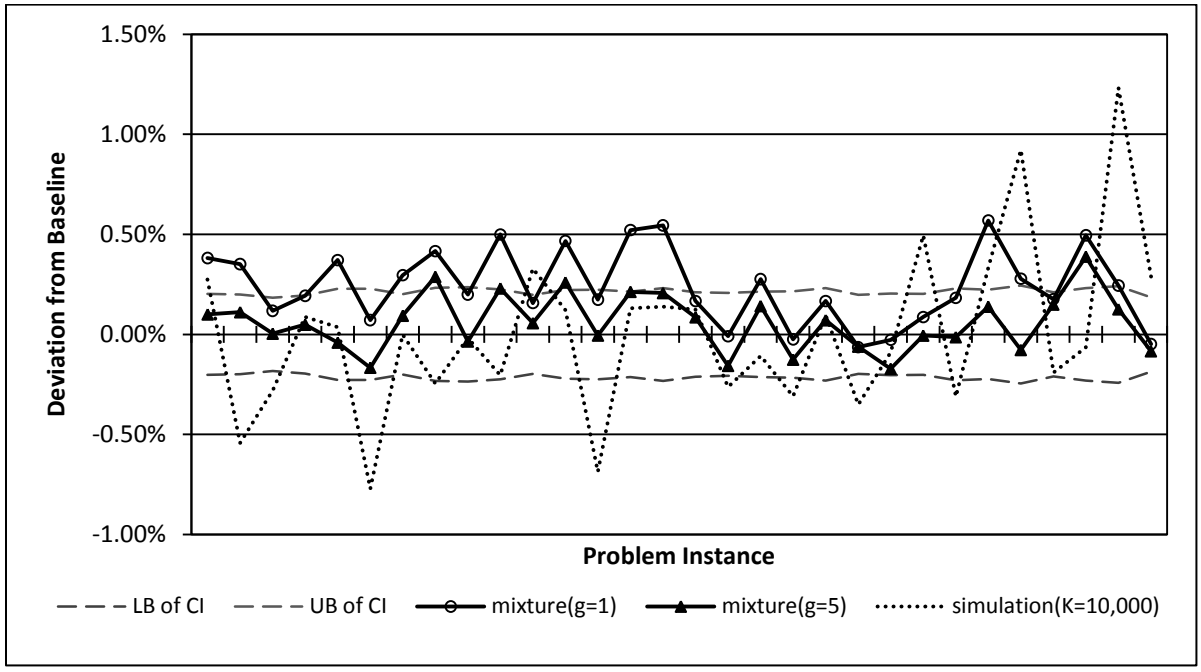


Figure 2.4: Relative deviations of expectation values from baseline results

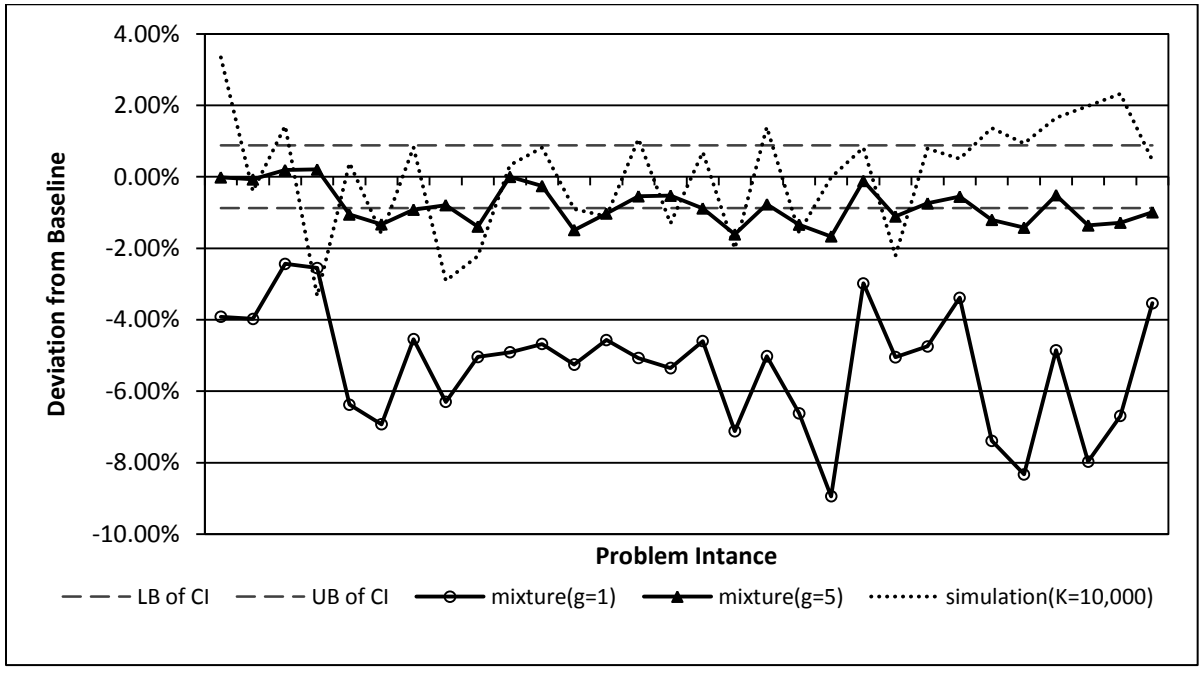


Figure 2.5: Relative deviation of variance values from baseline results

Chapter 3

Minimizing Conditional-Value-at-Risk for Stochastic Scheduling Problems

In Chapter 2, we have developed approximation methods to evaluate both the expectation and variance of a performance measure for a given schedule. Consideration of variance provides insight regarding the stochastic nature of a schedule, and also, it helps in avoiding riskier solutions. However, variance is not easy to handle if it is directly included in the objective function of an optimization problem. First, except for some special cases (such as the single-machine flow time problem discussed by De, Ghosh, and Wells (1992)), it is difficult to derive analytical expressions for the variances of various performance measures. In case that a scenario-based approach is used to formulate the underlying optimization problem, the sample variance of a performance measure will constitute a quadratic expression, which makes the problem difficult to solve. Second, minimizing variance of a random variable equally penalizes positive and negative deviations from its mean value. Suppose that two solutions yield identical expectation and variance values of the objective function, and the distribution of the objective function for the first solution has a longer tail on the right side of its PDF. For a minimization problem, a risk-averse decision maker would choose the second solution to avoid risk of large outcomes. But, based on the values of expectation and variance alone, it is not possible to distinguish between these two

solutions. To avoid the above drawbacks of variance, we introduce Conditional-Value-at-Risk (CVaR) as a criterion for stochastic scheduling. Minimizing this criterion has an added advantage due to its tendency of minimizing the expectation of a performance measure as well.

In the remainder of this chapter, we introduce CVaR and establish a general formulation for its minimization in Section 3.1. Section 3.2 contains an application of the above formulation to the single-machine, total weighted tardiness scheduling problem, and also, presents an optimization approach for the solution of this problem. Also, a heuristic procedure is presented for solving its large-sized instances. Results of numerical experimentation are provided to demonstrate the effectiveness of using CVaR. An extension of the use of CVaR to a parallel machine total weighted tardiness problem is presented in Section 3.3.

3.1 Optimizing CVaR in Stochastic Scheduling

In this section, we present motivation and general methodology for optimizing Conditional-Value-at-Risk of a performance measure in a stochastic scheduling problem.

3.1.1 Introduction

Definition 3.1. *Given a random variable X and a probability level* $\alpha \in (0,1)$*, Value-at-Risk (VaR) and Conditional-Value-at-Risk (CVaR) are, respectively, defined as follows:*

$$\eta_X(\alpha) = \inf \{x: F_X(x) \geq \alpha\}$$

and

$$\phi_X(\alpha) = E[X|X \geq \eta_X(\alpha)] = \frac{1}{1-\alpha} \int_{\eta_X(\alpha)}^{+\infty} x\, f_X(x)\, dx,$$

where $f_X(x)$ *and* $F_X(x)$ *are, respectively, the probability density function and the cumulative distribution function of X.*

Loosely speaking, $\phi_X(\alpha)$ is the average value of the $(1-\alpha)\cdot 100\%$ largest outcomes of X. As a risk measure, CVaR possesses several useful properties, which have enabled it to gain popularity in the fields of insurance and finance. First of all, CVaR is consistent with Second Degree Stochastic Dominance (SSD) (see Ogryczak and Ruszczynski (2002)). Specifically, if a random variable X dominates a random variable Y under the SSD rule (denoted as $X \succcurlyeq_{\text{SSD}} Y$), then X is better than Y with respect to all risk-averse nondecreasing utility functions. CVaR is consistent with SSD in the sense that

$$X \succcurlyeq_{\text{SSD}} Y \Rightarrow \phi_X(\alpha) \leq \phi_Y(\alpha), \qquad \forall \alpha \in (0,1).$$

Interestingly, this property does not always hold for the variance (see Porter and Gaumnitz (1972)). Secondly, CVaR is a *coherent* risk measure as defined by Artzner et al. (1999), in that it satisfies the four axioms: translation invariance, subadditivity, positive homogeneity, and monotonicity. VaR, however, does not satisfy subadditivity. Also, note that, for any random variable, X, with finite values of $\phi_X(\alpha)$ and $E[X]$, we always have $\phi_X(\alpha) \geq E[X], \forall \alpha \in (0,1)$. Therefore, minimizing the CVaR of a random variable tends to reduce its expected value as well. Furthermore, it can be shown that (see Rockafellar and Uryasev (2000)):

$$\phi_X(\alpha) = \min_{\eta \in \mathbb{R}} \left\{ \eta + \frac{1}{1-\alpha} E[(X-\eta)^+] \right\}, \tag{3.1}$$

where $(\xi)^+$ denotes $\max\{0, \xi\}$. This representation allows the incorporation of CVaR into an optimization framework. Wang and Ahmed (2008) have used CVaR to construct side-constraints while minimizing the expected objective value. In contrast, we consider next the formulation of a stochastic scheduling problem that directly minimizes CVaR as the objective function.

3.1.2 Problem Formulation and a Decomposition Approach

Consider a general scheduling problem whose deterministic version can be formulated as the following Mixed Integer Program (MIP):

$\mathbf{P_1}$: Minimize $\mathbf{h}^{\mathrm{T}}\mathbf{y}$

subject to: $\mathbf{x} \in \Gamma, \mathbf{A}\mathbf{x} \geq \mathbf{b}$

$\mathbf{E}\mathbf{x} + \mathbf{R}\mathbf{y} \geq \mathbf{g}$

$\mathbf{y} \geq 0,$

where the vectors $\mathbf{x}$ and $\mathbf{y}$ are the decision variables, and the set Γ denotes bound values and possibly certain integrality restrictions. In the stochastic setting considered, the parameters $\mathbf{E}$ and $\mathbf{g}$ are subject to random variations once $\mathbf{x}$ is determined due to the stochastic nature of the problem or due to lack of information. We assume that such randomness is captured by a finite set of scenarios, S, with corresponding parameter values $\mathbf{E}_s$ and $\mathbf{g}_s$, $\forall\, s \in S$. These scenarios are derived either from some discrete approximation of the underlying distributions of the problem parameters, or from some scenario generation procedure, with the probability value π_s being associated with a scenario s, $\forall\, s \in S$. The stochastic version of Problem $\mathbf{P_1}$ can be formulated as a two-stage stochastic program as follows:

$\mathbf{P_2}$: Minimize $\sum_{s \in S} \pi_s \mathbf{h}^{\mathrm{T}}\mathbf{y}_s$

subject to: $\mathbf{x} \in \Gamma, \mathbf{A}\mathbf{x} \geq \mathbf{b}$

$\mathbf{E}_s\mathbf{x} + \mathbf{R}\mathbf{y}_s \geq \mathbf{g}_s, \qquad \forall\, s \in S$

$\mathbf{y}_s \geq 0, \qquad \forall\, s \in S,$

where the objective function considers the traditional minimization of expected value. In this context, $\mathbf{x}$ is the vector of first stage decision variables, which need to be determined before the values of the random parameters are observed as alluded above, and $\mathbf{y}$ is a second stage variable with response $\mathbf{y}_s$ under scenario s, $\forall s \in S$. Furthermore, note that we assume a fixed recourse situation, so that the matrix $\mathbf{R}$ is constant across all scenarios.

Instead of the commonly used expectation objective as in Problem $\mathbf{P_2}$, we choose CVaR as the distributional property to minimize. Adopting the derivation (1) of Rockafellar and

Uryasev (2000), it can be shown that the following MIP provides an equivalent formulation for the minimization of CVaR:

$$\textbf{MP:}\quad \text{Minimize} \qquad \phi = \eta + \frac{1}{1-\alpha}\sum_{s\in S}\pi_s\mu_s$$

$$\text{subject to:} \qquad \mathbf{x} \in \Gamma, \mathbf{Ax} \geq \mathbf{b} \tag{3.2}$$

$$\eta + \mu_s \geq \mathbf{h}^{\mathrm{T}}\mathbf{y}_s, \qquad \forall\, s \in S \tag{3.3}$$

$$\mathbf{E}_s\mathbf{x} + \mathbf{R}\mathbf{y}_s \geq \mathbf{g}_s, \qquad \forall\, s \in S \tag{3.4}$$

$$\mu_s \geq 0 \text{ and } \mathbf{y}_s \geq 0, \qquad \forall\, s \in S \tag{3.5}$$

Because of the size and structure of Problem **MP**, particularly for a large number of scenarios, it is attractive to consider Benders decomposition (Benders 1962) to solve this problem. For each scenario $s \in S$, we define the second stage problem as follows:

$$\mathbf{MP_{II}}(s):\ Q_s(\mathbf{x}) \equiv \min_{\mathbf{y}_s \geq 0}\{\mathbf{h}^{\mathrm{T}}\mathbf{y}_s : \mathbf{R}\mathbf{y}_s \geq \mathbf{g}_s - \mathbf{E}_s\mathbf{x}\}.$$

We further assume relatively complete recourse, so that Problem $\mathbf{MP_{II}}(s)$ is feasible for all relevant values of $\mathbf{x}$ feasible to (3.2), and that $\mathbf{MP_{II}}(s)$ is bounded. Observe that, consequently, the first stage problem seeks to minimize ϕ subject to (3.2), $\mu_s \geq 0$, $\forall s \in S$, and that $\eta + \mu_s \geq Q_s(\mathbf{x})$, $\forall s \in S$. Hence, the first stage problem can be written as follows:

$$\mathbf{MP_I:}\quad \text{Minimize} \qquad \phi = \eta + \frac{1}{1-\alpha}\sum_{s\in S}\pi_s\mu_s$$

$$\text{subject to:} \qquad \mathbf{x} \in \Gamma, \mathbf{Ax} \geq \mathbf{b}$$

$$\eta + \mu_s \geq (\mathbf{g}_s - \mathbf{E}_s\mathbf{x})^{\mathrm{T}}\psi, \qquad \forall\, s \in S, \psi \in \Psi \tag{3.6}$$

$$\mu_s \geq 0, \qquad \forall\, s \in S,$$

where Ψ denotes the set of extreme point feasible solutions to the dual of $\mathbf{MP_{II}}(s)$ for any

$s \in S$, and the latter is given as follows:

$$Q_s(\mathbf{x}) \equiv \max_{\psi \geq 0}\{(\mathbf{g}_s - \mathbf{E}_s\mathbf{x})^{\mathrm{T}}\psi : \mathbf{R}^{\mathrm{T}}\psi \leq \mathbf{h}\}.$$

Since the set Ψ is generally exponentially sized, we solve Problem $\mathbf{MP_I}$ via a constraint relaxation and cut generation procedure. Specifically, given a solution $\mathbf{x}$ to some relaxation of $\mathbf{MP_I}$, that has only a subset of restrictions in (3.6) for some selected $\psi \in \Psi_s \subset \Psi$, $\forall s \in S$, we compute $Q_s(\mathbf{x})$ by solving $\mathbf{MP_{II}}(s)$, $\forall\, s \in S$. In case that the inequality $\eta + \mu_s \geq Q_s(\mathbf{x})$ is violated for some $s \in S$, we generate an optimality cut of the form $\eta + \mu_s \geq (\mathbf{g}_s - \mathbf{E}_s\mathbf{x})^{\mathrm{T}}\psi^*$ where ψ^* denotes an optimal dual solution to $\mathbf{MP_{II}}(s)$, and we augment $\Psi_s \leftarrow \Psi_s \cup \{\psi^*\}$. Also note that, as described by Sherali and Lunday (2010), we can generate maximally nondominated Benders cuts by perturbing the right-hand side of $\mathbf{MP_{II}}(s)$ to $\mathbf{g}_s(1+\lambda) - \mathbf{E}_s\mathbf{x}(1+\lambda\mathbf{e})$, where $\mathbf{e}$ is a conformable vector of ones and λ is a suitable (small) perturbation parameter. Adding such cuts for all $s \in S$ as necessary to the relaxed $\mathbf{MP_I}$, we re-solve the augmented first stage problem and repeat the above procedure until we obtain a first stage solution for which $\eta + \mu_s \geq Q_s(\mathbf{x})$, $\forall s \in S$. To reduce the number of added cuts, we aggregate, using equal weights, the optimality cuts that are generated within the same iteration. We also embed this cut generation procedure in a branch-and-bound (B&B) framework whenever Γ contains integrality restrictions, so that the B&B tree is explored only once as described by Geoffrion and Graves (1974). The resulting algorithm is a specialized implementation of the Integer L-shaped method (Birge and Louveaux (1997), Ch. 8). Additionally, the following valid inequality further improves the (initial) relaxation of $\mathbf{MP_I}$.

Proposition 3.1. *For any given set $S' \subset S$ such that $\pi_{S'} \triangleq \sum_{s\in S'} \pi_s \geq 1-\alpha$, the following inequality is valid for Problem* **MP**.

$$\phi \geq \frac{1}{\pi_{S'}} \sum_{s \in S'} \pi_s \mathbf{h}^{\mathrm{T}}\mathbf{y}_s \tag{3.7}$$

Proof: By aggregating inequalities (3.3) using weights of π_s, $\forall\, s \in S'$, we have

$$\sum_{s\in S'}\pi_s\eta+\sum_{s\in S'}\pi_s\mu_s\geq\sum_{s\in S'}\pi_s\mathbf{h}^{\mathrm{T}}\mathbf{y}_s$$

$$\Rightarrow\eta+\frac{1}{\pi_{S'}}\sum_{s\in S'}\pi_s\mu_s\geq\frac{1}{\pi_{S'}}\sum_{s\in S'}\pi_s\mathbf{h}^{\mathrm{T}}\mathbf{y}_s. \tag{3.8}$$

Since $\pi_{S'}\geq 1-\alpha$, we further have

$$\phi=\eta+\frac{1}{1-\alpha}\sum_{s\in S}\pi_s\mu_s\geq\eta+\frac{1}{\pi_{S'}}\sum_{s\in S}\pi_s\mu_s\geq\eta+\frac{1}{\pi_{S'}}\sum_{s\in S'}\pi_s\mu_s. \tag{3.9}$$

Using (3.8) in (3.9) thus yields $\phi\geq\frac{1}{\pi_{S'}}\sum_{s\in S'}\pi_s\mathbf{h}^{\mathrm{T}}\mathbf{y}_s$. ■

The idea here is to select some reasonable set of scenarios $S'\subset S$ that satisfy the condition of Proposition 3.1, and then, to explicitly incorporate the restrictions (3.7) along with the constraints (3.4) and (3.5) pertaining to $s\in S'$ directly within $\mathbf{MP_I}$. We denote these restrictions as:

$$(\phi,\mathbf{x},\mathbf{y})\in\mathcal{Z}. \tag{3.10}$$

In particular, this set $\mathcal{Z}$ provides a lower bound for CVaR based on the outcomes of a selected set of scenarios, and induces a tighter relaxation for the first stage problem $\mathbf{MP_I}$ without complicating its formulation too much. Ideally, the selected scenarios for S' should yield relatively large values of $\mathbf{h}^{\mathrm{T}}\mathbf{y}_s$ at optimality. Note that with this modification, ϕ is regarded both as a decision variable and as the objective function of $\mathbf{MP_I}$, where we accommodate, within the constraints, the additional inequality:

$$\phi\geq\eta+\frac{1}{1-\alpha}\sum_{s\in S}\pi_s\mu_s.$$

Alternatively, we can incorporate the constraints (3.3), (3.4), and (3.5) pertaining to $s\in S'$ directly within $\mathbf{MP_I}$. We denote these restrictions as:

$$(\eta,\boldsymbol{\mu},\mathbf{x},\mathbf{y})\in\mathcal{Z}', \tag{3.11}$$

which can be regarded as a disaggregated version of the restrictions in (3.10).

3.2 Application to a Single-Machine Scheduling Problem

To demonstrate the viability and benefit of optimizing CVaR, we first apply the above methodology to the single-machine total weighted tardiness (TWT) problem. We assume the processing times to be the only random elements in this problem. Consider the following notation:

Parameters:

J: Set of jobs;

w_j: Weight of job j, $\forall\, j \in J$;

d_j: Due date of job j, $\forall\, j \in J$;

p_j^s: Processing time of job j under scenario s, $\forall\, j \in J, s \in S$.

Decision variables:

c_j^s: Completion time of job j under scenario s, $\forall\, j \in J, s \in S$;

t_j^s: Tardiness of job j under scenario s, $\forall\, j \in J, s \in S$;

η: A threshold value (equal to the Value-at-Risk when an optimal solution is obtained);

μ_s: Amount of TWT exceeding the threshold value η under scenario s, $\forall\, s \in S$;

$z_{i,j}$: Job precedence binary variable, where, $z_{i,j} = 1$ if job i is processed before job j, and $z_{i,j} = 0$ otherwise, $\forall i \neq j \in J$.

The model formulation of the single-machine total weighted tardiness problem (SM-TWTP) is as follows:

SM-TWTP:

Minimize $$\phi = \eta + \frac{1}{1-\alpha}\sum_{s\in S}\pi_s\mu_s$$

subject to:

$$\eta + \mu_s \geq \sum_{j\in J} w_j t_j^s, \qquad \forall\, s \in S \tag{3.12}$$

$$\sum_{i\in J\setminus\{j\}} p_i^s z_{i,j} + p_j^s \leq c_j^s, \qquad \forall\, s \in S, j \in J \tag{3.13}$$

$$t_j^s + d_j \geq c_j^s, \qquad \forall\, s \in S, j \in J \tag{3.14}$$

$$z_{i,j} + z_{j,i} = 1, \qquad \forall\, i \neq j \in J \tag{3.15}$$

$$z_{i,j} + z_{j,k} + z_{k,i} \leq 2, \qquad \forall\, i \neq j \neq k \in J \tag{3.16}$$

$$c_j^s \geq 0, t_j^s \geq 0, \mu_s \geq 0, \qquad \forall\, s \in S, j \in J$$

$$z_{i,j} \in \{0,1\}, \qquad \forall\, i \neq j \in J.$$

For each scenario $s \in S$, Constraint (3.12) determines μ_s as the amount of TWT exceeding the threshold value of η. Constraint (3.13) bounds the job completion times according to job sequencing relationships. Constraint (3.14) determines the tardiness of jobs. Constraints (3.15) and (3.16) ensure feasibility of the job sequence by eliminating cyclic sequences (see Sarin, Sherali, and Bhootra (2005)). Note that the variables $\{z_{i,j}\}$ and $\{c_j^s, t_j^s\}$, respectively, correspond to the variables **x** and **y** in the general formulation **MP**.

As an illustration, consider an 8-job problem with 100 scenarios. The job processing times are assumed to follow left-truncated normal distributions (truncated at zero to ensure nonnegativity). The job parameter values are summarized in Table 3.1. Scenario-wise values of p_j are generated via Monte Carlo sampling. This problem was solved to optimality for each of the criteria of minimizing the expected value as well as minimizing CVaR (with $\alpha = 0.8$), which resulted in different optimal sequences: sequence $\delta_E = \langle 1,6,8,4,5,7,2,3 \rangle$ for the expected value criterion, and sequence $\delta_C = \langle 6,8,4,7,5,1,2,3 \rangle$ for the CVaR criterion. The empirical cumulative distribution functions of total weighted tardiness under both sequences are plotted in Figure 3.1. Note that δ_C, the sequence that minimizes CVaR, does not achieve the minimal expected value given by the optimal sequence δ_E. However, for a slight increment in expected value (about 7%), δ_C results in almost a 50% reduction in the value of variance. As a result, the cumulative distribution function of TWT corresponding to δ_C reaches probability 1 at the TWT value of 1179.5, while the TWT corresponding to δ_E has a substantial associated probability of exceeding this value. This example illustrates the risk-averse nature of CVaR and its effectiveness in reducing variability.

To further illustrate the above observation, we randomly generated 30 sample problems

with 8 jobs and with uniformly distributed processing times. The processing time distribution for job j was taken as $\mathcal{U}\left(\rho_j(1-r_j), \rho_j(1+r_j)\right)$, where the values of ρ_j and r_j were generated from uniform distributions $\mathcal{U}(20,100)$ and $\mathcal{U}(0.1,0.25)$, respectively. The job due dates were generated according to

$$\mathcal{U}\left(1-\mathrm{TF}-\frac{\mathrm{RDD}}{2}, 1-\mathrm{TF}+\frac{\mathrm{RDD}}{2}\right)\cdot\sum_{j\in J}E[P_j],$$

with a tardiness factor, TF, of 0.6, and a relative due date, RDD, of 0.4. One hundred scenarios were generated for each sample problem. After the optimal sequences (δ_E and δ_C) were obtained, the expectation and variance of the total weighted tardiness were evaluated for both δ_E and δ_C. The results obtained are plotted in Figure 3.2. The horizontal axis represents the ratio of expectations corresponding to the optimal sequences, δ_C and δ_E, which indicates the relative increment in expectation due to choosing δ_C over δ_E. The vertical axis corresponds to the ratio of the corresponding variances. The smaller the ratio, the greater is the reduction in variability of the TWT value. Note that for 15 out of the 30 problem instances used, the resulting distributions of TWT are different for δ_E and δ_C. In these cases, choosing δ_C over δ_E leads to a significant reduction in the variability of TWT, with only a slight increment in its expected value.

Table 3.1: Parameters of an example problem

Job j	1	2	3	4	5	6	7	8
$E[P_j]$	64	98	93	21	12	55	14	19
$Var[P_j]$	42	7	8	36	20	400	5	100
w_j	4	3	1	3	2	5	2	4
d_j	137	157	147	150	168	141	162	142

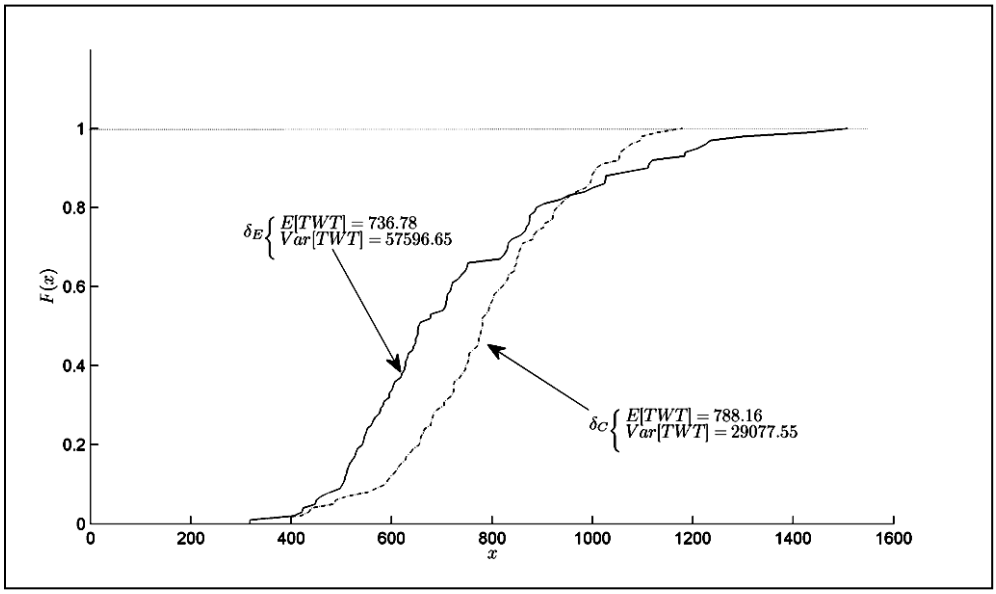


Figure 3.1: Cumulative distribution functions of TWT for the expected value and CVaR criteria

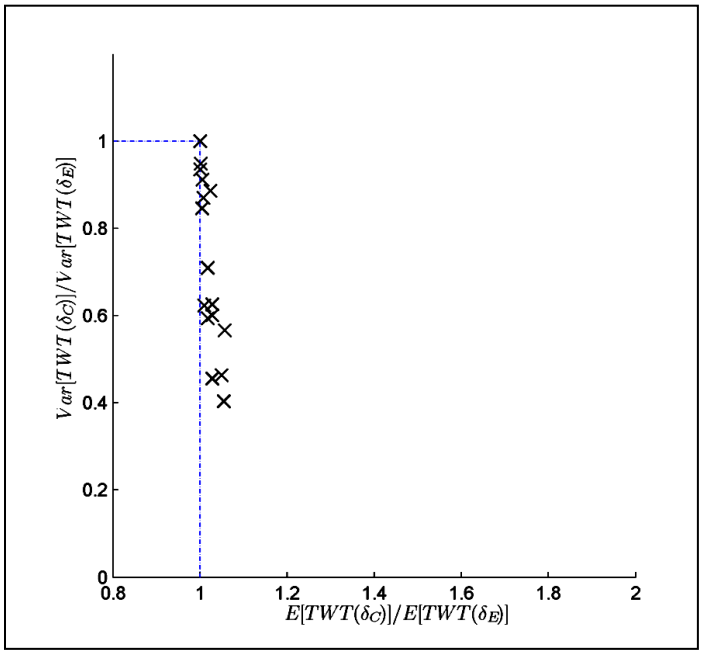


Figure 3.2: Expectation-variance comparison for the sequences δ_C and δ_E for the 8-job single-machine problems.

Impact of Different Probability Levels

In the above discussion, we have assumed the probability level of $\alpha = 0.8$. In order to experimentally investigate the impact of different probability levels on the optimal solution to the SM-TWTP, we reused the problem instance presented in Table 3.1 (for 100 scenarios). We systematically varied the probability level, α, from 0 to 0.99 in steps of 0.01, and solved the SM-TWTP to optimality for each value of α. Two different optimal job sequences were obtained, corresponding to α lying in the ranges of [0,0.5] and [0.51,0.99], respectively. The expectation and variance corresponding to these two sequences are depicted in Figure 3.3, along with the Extreme Expectation-Variance (XEV-) efficient sequences (De et al., 1992), which yield the minimum of $\lambda E[\text{TWT}] + (1-\lambda)E[\text{TWT}]$ for different values of $\lambda \in [0,1]$. Note that two CVaR optimal sequences overlap with two XEV-efficient sequences. A larger α value induces a smaller variance. The XEV-efficient sequences provide more options on the efficient frontier of simultaneously minimizing expectation and variance. However, determination of XEV-efficient sequences is a more difficult problem for job tardiness.

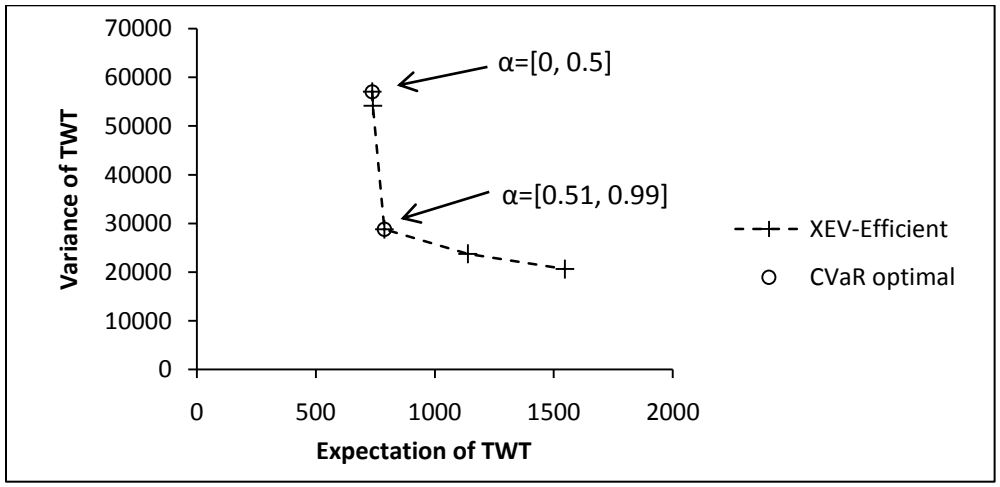


Figure 3.3: Expectation-variance comparison of different sequences for the 8-job single-machine problems.

3.2.1 Solution Approach for SM-TWTP

To solve the SM-TWTP model more efficiently, we implemented the Integer L-shaped Method after enhancing its first stage model with the valid inequalities of (3.10) and (3.11). In particular, the selected set of scenarios, S', was determined according to the total weighted processing times. A scenario having a higher value of the total weighted processing time tends to have larger tardiness values as well. Hence, to induce a tight bound, we constructed the set S' incrementally, starting with an empty set, and then, adding scenarios in nonincreasing order of the total weighted processing time, until $\pi_{S'}$ reached $1-\alpha$. In addition, since a job sequence (given by $\{z_{i,j}\}$) is fixed by the first stage problem $\mathbf{MP_I}$, the subproblem has the following simple structure, which leads to its straightforward solution in closed-form, where $\mathcal{C}_j^s \equiv \sum_{i\in J\setminus\{j\}} p_i^s\, z_{i,j} + p_j^s$, $\forall j \in J, s \in S$:

$\mathbf{MP_{II}}(s)$**:** Minimize $\sum_{j\in J} w_j t_j^s$

subject to:

$$c_j^s \geq \mathcal{C}_j^s, \qquad \forall j \in J$$
$$t_j^s - c_j^s \geq -d_j, \qquad \forall j \in J$$
$$c_j^s \geq 0, t_j^s \geq 0, \qquad \forall j \in J.$$

Dual of $\mathbf{MP_{II}}(s)$**:** Maximize $\sum_{j\in J}\left(\mathcal{C}_j^s \kappa_j - d_j \nu_j\right)$

subject to:

$$\kappa_j - \nu_j \leq 0, \qquad \forall j \in J$$
$$\nu_j \leq w_j, \qquad \forall j \in J$$
$$\kappa_j \geq 0, \nu_j \geq 0, \qquad \forall j \in J.$$

Note that an optimal dual solution is given by $\kappa_j = \nu_j = w_j$, if $\mathcal{C}_j^s \geq d_j$; and $\kappa_j = \nu_j = 0$ otherwise. It is easy to verify that this produces maximally nondominated Benders cuts (Sherali and Lunday, 2010).

To evaluate the performance of our decomposition approach, we benchmarked it against a method that directly uses CPLEX 10.1 to solve the formulation **MP**. To further reduce the size of the problem formulation, we applied the following substitution for both methods, which is implied by Constraint (3.15):

$$z_{i,j} = 1 - z_{j,i}, \qquad \forall i > j.$$

Experiments were conducted on a workstation with a 3.6 GHz CPU and 3 GByte memory. We used $\alpha = 0.8$. In addition to the 8-job problems previously considered, we further generated 30 sample problems having 15 jobs and 400 scenarios. A time limit of 3,600 seconds is applied to avoid long CPU time. To begin with, we applied the Integer L-shaped method with and without the addition of the inequalities defined by $\mathcal{Z}$ (3.10) and $\mathcal{Z}'$ (3.11), in order to determine the effectiveness of these alternative formulations. The results are summarized in Table 3.2. Note that the inclusion of the inequalities of $\mathcal{Z}$ or $\mathcal{Z}'$ greatly enhances the performance of the L-shaped method by inducing a good lower bound for the first stage problem. Between $\mathcal{Z}$ and $\mathcal{Z}'$, the set $\mathcal{Z}'$ generated fewer nodes, and hence, required lesser CPU time. However, both sets of restrictions generated identical initial lower bound values. We also experimented with McDaniel and Devine's (1977) approach of generating an initial set of optimality cuts based on the LP relaxation of $\mathbf{MP_I}$. However, it was not found to improve the computational efficiency of the integer L-shaped method.

Table 3.2: Results on the use of L-shaped method with and without the set $\mathcal{Z}$ and $\mathcal{Z}'$ inequalities

No. of Jobs	No. of Scenarios	Valid Inequalities	Opt Gap of LP Bound	No. of Nodes	CPU Times (seconds)
8	100	N/A	100%	2,400.57	6.49
		$\mathcal{Z}$	21.18%	140.90	0.92
		$\mathcal{Z}'$	21.18%	86.00	0.5
15	400	N/A	100%	110,310.00	3,600.00
		$\mathcal{Z}$	27.99%	10,077.20	571.71
		$\mathcal{Z}'$	27.99%	7,793.90	362.37

With the addition of these inequalities to $\mathbf{MP_I}$, we further compared the performance of the L-shaped method with the direct solution of Problem **MP** by CPLEX. The results are summarized in Table 3.3. Although the L-shaped method generated more B&B nodes, since its first-stage problem is smaller in size than the original problem **MP**, it consumed significantly lesser CPU time, reducing the average computational effort required for the 8-job and 15-job instances by 61.6% and 81.0%, respectively.

Table 3.3: Comparison of the results using the proposed L-shaped method and direct solution by CPLEX

No. of Jobs	No. of Scenarios	Approach	Opt Gap of LP Bound	No. of Nodes	CPU Times (seconds)
8	100	Direct solution by CPLEX	20.53%	64.63	1.46
		L-shaped method	21.18%	86.00	0.56
15	400	Direct solution by CPLEX	27.54%	5,910.70	2,012.37
		L-shaped method	27.99%	7,793.90	362.37

3.2.2 Solution of Large-sized Problems

Although the L-shaped method provides a good alternative approach than solving Problem **MP** directly by CPLEX, yet, to handle large-sized problem instances, we adapted the deterministic Dynasearch method proposed by Congram, Potts, and van de Velde (2002) and further extended by Grosso, Della Croce, and Tadei (2004) in order to address the stochasticity present in our problem. Dynasearch is a local search heuristic method. Given a complete job sequence, δ, as the current solution, Dynasearch searches all solutions that can be reached from δ by non-overlapping pairwise interchanges of jobs. If a neighboring solution is found to be better than δ, it is designated as the starting point for the next iteration of the local search process. Although the local search neighborhood has a size that

grows exponentially with the number of jobs, Dynasearch affords the efficient determination of a local optimal solution by using a dynamic programming approach. For this approach, the stages are defined by the number of jobs already fixed in the leading part of a job sequence. At each stage j, Dynasearch considers the application of non-overlapping pairwise interchanges over the first j jobs of δ. The minimum TWT of these j jobs is regarded as the value function $v(j)$. The value of $v(j+1)$ can then be determined recursively by considering the following two options: keeping the $(j+1)^{\text{th}}$ job at its original position in δ, or exchanging it with the i^{th} job, $\forall\, i \leq j$. Dynasearch has been shown to be quite competitive in solving single-machine TWT problems. Grosso et al. (2004) generalized the idea of pairwise interchanges to also include forward-shifted reinsertions and backward-shifted reinsertions, calling this procedure a Generalized Pairwise Interchanges (GPI) scheme. Our method is built upon this GPI-version of Dynasearch to exploit its full strength.

Congram et al. (2002) and Grosso et al. (2004) considered the deterministic version of the single-machine TWT problem. In our stochastic setting, multiple scenarios introduce additional complications. First, the value function of the dynamic programming needs to be modified to account for scenario-wise outcomes. Second, the CVaR objective function precludes the separability required by dynamic programming. In particular, given a partial sequence $\langle \xi, \zeta \rangle$ that minimizes CVaR for these $|\xi| + |\zeta|$ jobs, it is not necessarily true that ξ is optimal with respect to CVaR for the jobs in ξ. We show this by a counterexample: Let $|S| = 5, \alpha = 0.8, |\xi| = 2$, and $|\zeta| = 1$. For the two jobs in ξ, there exist two alternative sequences: ξ and $\xi' \equiv \text{reverse}(\xi)$. Suppose that the scenario-wise TWT values of these two sequences are $\{1, 2, 2, 5, 2\}$ for ξ, and $\{1, 2, 3, 3, 4\}$ for ξ'. The corresponding CVaR values are 5 and 4, respectively. If the weighted tardiness values of the third job, ζ, are $\{4, 3, 2, 1, 3\}$, then the total weighted tardiness values of the three jobs are $\{5, 5, 4, 6, 5\}$ for the sequence $\{\xi, \zeta\}$, and $\{5, 5, 5, 4, 7\}$ for the sequence $\{\xi', \zeta\}$. Note that, although $\{\xi, \zeta\}$ yields a smaller CVaR value (= 6) for the three job problem, ξ' gives a better value of CVaR for the first two jobs

than ξ does. This outcome follows by the fact that if $\max\{a_1, b_1\} \geq \max\{a_2, b_2\}$, then $\max\{a_1 + e, b_1 + f\} \geq \max\{a_2 + e, b_2 + f\}$ is not necessarily true for all real e, f.

To overcome these difficulties, our proposed method comprises two types of steps: *Selection step* (S-step) and *Optimization step* (O-step). Hence, we call our procedure *SO-Dynasearch*. The Selection step chooses a set of worst scenarios, $\hat{S}$, which yield the highest TWT-values for the current solution. This set is defined such that the cumulative probability value, $\pi_{\hat{S}} \equiv \sum_{s \in \hat{S}} \pi_s$, is no less than $1 - \alpha$, but removing any scenario from it would violate this condition. The Optimization step then uses dynamic programming to search for the sequence that optimizes the average value (weighted by probabilities) of the total weighted tardiness over the scenarios in $\hat{S}$. Note that the weighted average over $\hat{S}$ gives a lower bound on CVaR by Proposition 3.1. The definition of stages is the same as that in the Dynasearch method. At each stage j, we consider all non-overlapping combinations of GPI operations over the jobs $\delta_1, \ldots, \delta_j$. The optimal value of $\frac{1}{\pi_{\hat{S}}} \sum_{s \in \hat{S}} \pi_s \sum_{k=1}^{j} w_{[k]} t_{[k]}^s$ [†] is regarded as the value function $v(j)$, and is computed according to the following recursive equation:

$$v(j) = \min\left\{ v(j-1) + \frac{1}{\pi_{\hat{S}}} \sum_{s \in \hat{S}} \pi_s \, w_{[j]} t_{[j]}^s \, , \min_{1 \leq i \leq j-1; \gamma} \left\{ v(i-1) + \frac{1}{\pi_{\hat{S}}} \sum_{s \in \hat{S}} \pi_s \, I_s^{\gamma}(i,j) \right\} \right\},$$

where $I_s^{\gamma}(i,j)$ is the total weighted tardiness of jobs $\delta_i, \ldots, \delta_j$ in scenario s, subject to the application of some GPI operator $\gamma(i,j)$. In view of these two types of steps, our SO-Dynasearch method can be described as follows:

Control Parameters:

N_0: Maximum number of S-steps after each random start;

N_1: Total number of random starts.

Variables:

ϕ^*: The smallest value of CVaR found so far;

δ: The best incumbent solution (job sequence) found so far;

[†] $w_{[k]}$ and $t_{[k]}^s$ denote the weight and tardiness of the job in the k^{th} position, respectively.

i_s: Number of S-steps allowed before the next random start;

i_r: Remaining number of random starts.

Functions:

TWT(δ, s): Total weighted tardiness corresponding to solution δ under scenario s.

SO-Dynasearch Method:

Step 1. Let $\phi^* = \infty$, $i_s = N_0$, and $i_r = N_1$.

Randomly generate a job sequence and denote it by δ.

Step 2. Use δ to calculate the values of total weighted tardiness under all scenarios in S.

Determine the set of worst scenarios $\hat{S}$ according to their TWT values (S-step).

Step 3. Perform iterated local search starting from δ (O-step):

a) Use dynamic programming to find a sequence δ', in the GPI neighborhood of δ, which minimizes the average value of TWT across all scenarios in $\hat{S}$.

b) If δ' is better than δ, let $\delta = \delta'$, and go to Step 3a; otherwise, continue.

Step 4. Use δ to calculate total weighted tardiness values under all scenarios in S.

Determine ϕ as the resulting CVaR value. If $\phi < \phi^*$, let $\phi^* = \phi$ and $\delta^* = \delta$.

Step 5. If $i_s > 0$ and $\phi > (1/\pi_{\hat{S}}) \cdot \sum_{s \in \hat{S}} \pi_s TWT(\delta, s)$, let $i_s \leftarrow i_s - 1$, and go to Step 2; otherwise, continue.

Step 6. If $i_r = 0$, stop; otherwise, let $i_r \leftarrow i_r - 1$ and $i_s = N_0$.

Step 7. Permute δ^* by using random pairwise interchanges, and denote the resulting sequence as δ. Go to Step 2.

Note that in Step 5, the estimated objective value $(1/\pi_{\hat{S}}) \cdot \sum_{s \in \hat{S}} \pi_s TWT(\delta, s)$ is compared with the true corresponding value of CVaR. If these two values are not the same, the

current set of $\hat{S}$ does not constitute the worst scenarios under sequence δ. Therefore, the S-step and O-step are repeated to seek a further improvement. The parameter N_0 is used to control the number of repetitions and to avoid an infinite loop. Also, note that SO-Dynasearch can be used to optimize E[TWT] by letting $\alpha = 0$, in which case, the O-step would be executed only once after each random start.

To evaluate the quality of solutions generated by SO-Dynasearch and its computational performance, we implemented it for the same data set used to produce the results of Table 3.3, for both types of objectives ($CVaR \equiv \phi_{\mathrm{TWT}}(0.8)$ and E[TWT]). The resulting solutions were found to be optimal for all the problem instances considered. Moreover, the average CPU time required by SO-Dynasearch for the 15-job instances, in particular, was within 25 seconds, which yields more than a 14-fold and 80-fold improvement over the L-shaped method and the direct solution of Problem **MP** by CPLEX, respectively.

Next, we applied SO-Dynasearch to a sample set of 30 larger problems with 35 jobs and 800 scenarios, which were generated using the same procedure as previously described. SO-Dynasearch was able to solve these problems with an average CPU time of 297.2 seconds for minimizing $CVaR \equiv \phi_{\mathrm{TWT}}(0.8)$ and 1032.8 seconds for minimizing E[TWT]. Based on the solutions obtained, the distribution of TWT was determined for each of the 30 problems. The resulting statistics are plotted in Figure 3.4. Note that the optimization of CVaR tends to reduce variance while only slightly increasing the expected value of TWT. This phenomenon is evident in 28 out of the 30 problems tested.

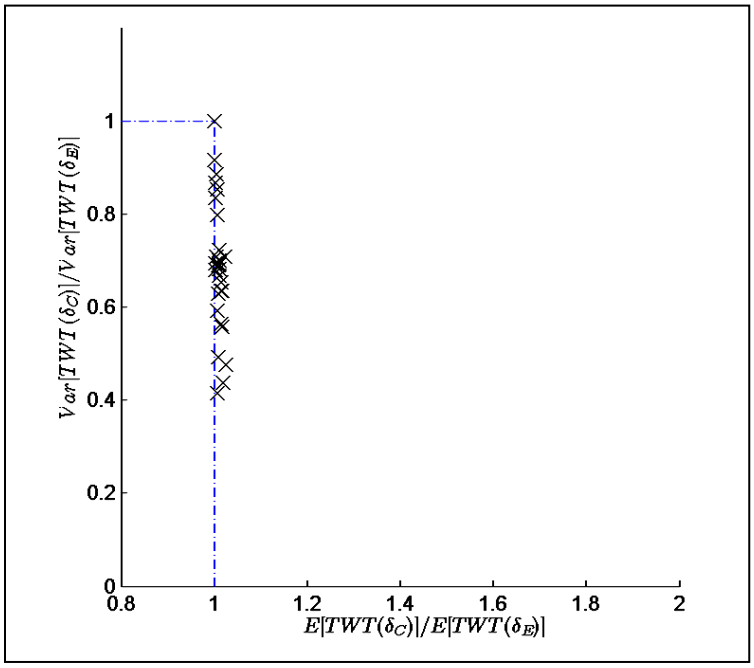


Figure 3.4: Expectation-variance comparison for the sequences δ_C and δ_E for the 35-job single-machine problems

3.3 Application to a Parallel-Machine Scheduling Problem

We extend the above discussion to the problem of minimizing the CVaR related to the total weighted tardiness in an identical parallel-machine environment. As in the single-machine case, we assume that the only uncertain elements are the random processing times. Consider the following notation:

Parameters:

J: Set of jobs;

M: Set of machines;

w_j: Weight of job j, $\forall\, j \in J$;

d_j: Due date of job j, $\forall\, j \in J$;

p_j^s: Processing time of job j under scenario s, $\forall\, j \in J, s \in S$.

Decision variables:

t_j^s: Tardiness of job j under scenario s, $\forall\, j \in J, s \in S$.

η: A threshold value (equal to the Value-at-Risk when an optimal solution is obtained);

μ_s: Amount of TWT exceeding the threshold value η under scenario s, $\forall s \in S$;

$x_{i,j}$: Job assignment; $x_{i,j} = 1$, if job j is assigned to machine i, and $x_{i,j} = 0$, otherwise, $\forall\, i \in M, j \in J$.

$z_{j,k}$: Sequencing decision when job j and job k are assigned to the same machine; $z_{j,k} = 1$, if job j precedes job k, and $z_{j,k} = 0$, otherwise, $\forall\ j \neq k \in J$.

$\omega_{k,j}^i$: Linearization variable defined as $\omega_{k,j}^i \equiv z_{k,j} x_{i,k}$.

PM − TWTP:

Minimize $$\phi = \eta + \frac{1}{1-\alpha}\sum_{s\in S} \pi_s \mu_s$$

subject to:

$$\sum_{i\in M} x_{i,j} = 1, \qquad \forall\, j \in J \tag{3.17}$$

$$\sum_{j\in J} j x_{(i-1),j} \geq \sum_{j\in J} j x_{i,j}, \qquad \forall\, i \in M, i > 1 \tag{3.18}$$

$$z_{k,j} + z_{j,k} = 1, \qquad \forall\, j \neq k \in J \tag{3.19}$$

$$z_{j,k} + z_{k,l} + z_{l,j} \leq 2, \qquad \forall\, j \neq k \neq l \in J \tag{3.20}$$

$$\sum_{i\in M} i x_{i,j} \geq \sum_{i\in M} i x_{i,k} - (|M| - 1) z_{j,k}, \qquad \forall\, j \neq k \in J \tag{3.21}$$

$$\omega_{k,j}^i + 1 \geq z_{k,j} + x_{i,k}, \qquad \forall\, i \in M, j \neq k \in J \tag{3.22}$$

$$\eta + \mu_s \geq \sum_{j\in J} w_j t_j^s, \qquad \forall\, s \in S \tag{3.23}$$

$$\sum_{k\in J\setminus\{j\}} p_k^s \omega_{k,j}^i + p_j^s x_{i,j} \leq d_j + t_j^s + \big(\Lambda - d_j\big)\big(1 - x_{i,j}\big), \tag{3.24}$$

$$\forall\, s \in S, i \in M, j \in J$$

$$x_{i,j} \in \{0,1\}, z_{j,k} \in \{0,1\}, \omega_{k,j}^i \in [0,1], \qquad \forall\, i \in M, j \neq k \in J$$

$$t_j^s \geq 0, \mu_s \geq 0, \qquad \forall\, s \in S, j \in J.$$

Constraint (3.17) assigns each job j to a machine in M. Since all machines are identical, Constraint (3.18) serves as symmetry-breaking constraints, requiring the summation of job indices on a lower-numbered machine to be no less than that on a higher-numbered machine. Constraints (3.19) and (3.20) are, respectively, two-city and three-city subtour elimination constraints for the sequencing priority variables (see Sarin, Sherali, and Bhootra (2005)). Constraint (3.21) reduces the symmetry caused by the inconsequence of the variables $z_{j,k}$ when job j and job k are assigned to different machines, by requiring a job assigned to a lower-indexed machine to precede the other job (in light of (3.19)). Constraints (3.22) and (3.23), respectively, enforce the definitional roles for $\omega_{k,j}^{i}$ and μ_s. Constraint (3.24) determines the tardiness of job j when it is assigned to machine i. Here, the parameter Λ is a sufficient large number to ensure the validity of Constraint (3.24) when $x_{i,j} = 0$. In our implementation, we took Λ as a pre-calculated upper bound value on the makespan.

We adapted the 30 instances of the 8-job single machine problem used for the experimental results presented in Table 3.2, in order to conduct a computational study for the parallel machine case. We assumed two identical machines in parallel, where the job due dates were reduced by half accordingly. For the sake of illustrating the nature of solutions obtained, we used CPLEX 10.1 to obtain optimal solutions for minimizing $\phi_{\mathrm{TWT}}(0.8)$ and $E[\mathrm{TWT}]$. The expectation and variance of TWT were calculated for both resulting solutions, and the related statistics are depicted in Figure 3.5. Note that our previous observations with respect to minimizing CVaR versus $E[\mathrm{TWT}]$ carry over to the parallel machine case as well; optimizing CVaR has the potential to reduce variability significantly, with only a slight increase in the expected value of TWT.

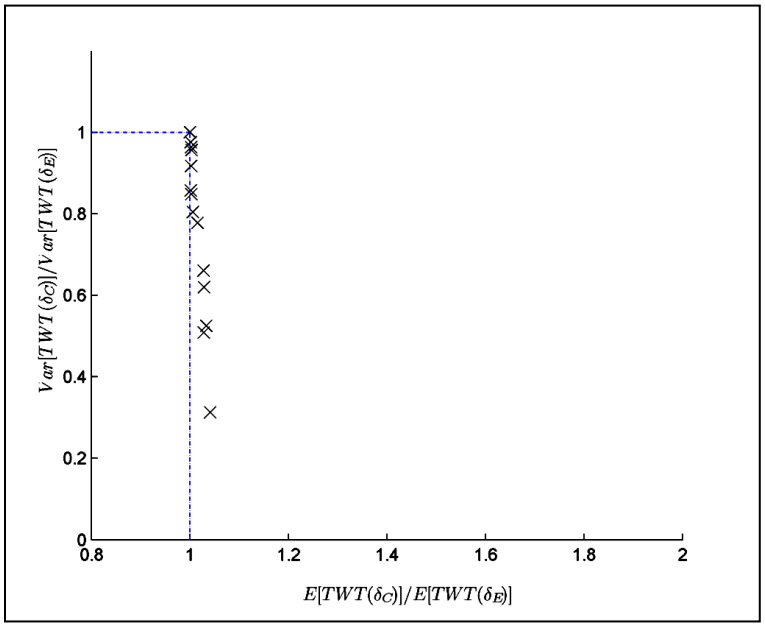


Figure 3.5: Expectation-variance comparison for the sequences δ_C and δ_E for 8-job parallel-machine problems

Chapter 4

Scenario Generation for Lower Bounding in Stochastic Scheduling Problems

Traditionally, stochastic programs are regarded as very difficult to solve, except for some special problem categories. One principal difficulty comes from the non-trivial effort needed to determine the objective value for a given solution. Since parameters are random variables, the determination of the objective value usually amounts to a multi-dimensional integration. In Chapter 2, we have developed approximation methods for the expectation and variance of various performance measures. These methods are based on the assumption that a schedule is given for the problem on hand. They do not directly determine an optimal solution. To evaluate a scheduling performance measure within a stochastic programming model, the most commonly adopted approach is to approximate the original distribution of problem parameters by a discrete distribution with a finite support. The original problem is thus transformed to a scenario-based model. Each scenario corresponds to one possible realization of problem parameters for which the value of the performance measure can be determined accordingly. The goal is to optimize some distributional property of the performance measure, which can be calculated approximately

from its scenario-wise outcomes. This is the approach we have adopted to minimize CVaR in Chapter 3. Advancements in computational capability have made it affordable to solve optimization models with a reasonable number of scenarios. As pointed out by Kaut and Wallace (2003), an appropriate scenario generation approach holds the key to a high quality approximation, besides providing the facility to reasonably generate an increased number of scenarios. Our focus in this chapter is somewhat different from that of Kaut and Wallace (2003) in that we are more concerned here with the quality of a *lower bound* derived from a scenario-based model.

4.1 Introduction

One possible approach for generating scenarios for a stochastic program is to fix the parameter values based on Monte-Carlo sampling (Law and Kelton 1999). For example, the Sample Average Approximation method (Kleywegt, Shapiro, and Homem-de-Mello 2002) first generates a set of scenarios using a random number generator. Then, a scenario-based model is solved with respect to this fixed set of scenarios. Once a feasible solution is obtained, the objective value of its outcome is estimated using a large number of scenarios. This provides an upper bound on the optimal objective value for the original stochastic problem. To evaluate the quality of this solution in terms of its optimality gap, we also need to establish a lower bound on the objective value for the original stochastic program. Mak, Morton, and Wood (1999) considered the following bounding technique. Suppose that the goal is to minimize the expectation of a performance measure, $G(x,\theta)$, where x is the decision variable and θ denotes the random parameter. We can, instead, minimize the average performance, $\frac{1}{K}\sum_{s=1}^{K} G(x,\theta^s)$, over K randomly generated scenarios $\{\theta^s, s = 1,\dots,K\}$. Let the optimal solution value of this scenario-based model be

$$G_K^* \equiv \min_{x\in X} \frac{1}{K}\sum_{s=1}^{K} G(x,\theta^s),$$

where the set X is the solution space constraining x. Since the scenario-wise parameter θ^s is randomly generated, the corresponding optimal objective, G_K^*, is also a random variable. Mak et al. (1999) showed that its expectation, $E[G_K^*]$, is a valid lower bound for the real optimal value, $G^* \equiv \min_{x \in X} E[G(x, \xi)]$. Therefore, a probabilistic lower bound for G^* can be determined from a statistical confidence interval of $E[G_K^*]$, which is established based on a repeated calculation of G_K^* with independent sets of scenarios. In other words, the scenario-based model needs to be solved to optimality multiple times with different sets of random scenarios to generate samples of G_K^*. These sample values could then be used to establish a confidence interval of $E[G_K^*]$, the lower end-point of which yields a probabilistic lower bound on G^*.

On the other hand, a Discrete Bounding Approximation can be used to generate exact lower bounds for G^* (Birge and Louveaux (1997), Section 9.2). This approach is based on an extension of Jensen's inequality (Ross and Peköz (2007), Section 4.1). The support of a random parameter is partitioned into small regions. In each region, the conditional expectation of this parameter is calculated and regarded as the outcome of a discrete distribution, which is then used to replace the original distribution of this parameter. This approach has been discussed by Birge and Wets (1986), and Frauendorfer and Kall (1988), and was later improved by Edirisinghe and Ziemba (1994). The developed methodology has been applied to optimize the expectation objective in a two-stage stochastic linear program, where subgradient information of the second stage value function is used to determine partitions and to compute bound values.

In this Chapter, we develop a scenario generation method using the approach of Discrete Bounding Approximation. Our new method is based on a recursive partitioning of the sample space of random parameters, and it generates exact lower bounds for stochastic scheduling problems with relatively less computational effort. The proposed method can be incorporated into an optimization framework to establish an optimality gap for the solution obtained.

The remainder of this chapter is organized as follows. In Section 4.2, we introduce the

theoretical basis of Discrete Bounding Approximation for a general category of performance measures. Section 4.3 develops an alternative scenario generation procedure for Discrete Bounding Approximation, assuming the independence of problem parameters. In Section 4.4, we consider the single machine total weighted tardiness problem and present results of our numerical experimentation to compare the performances of different lower bounding approaches.

4.2 Lower Bounding Based on Stochastic Dominance

We assume that the structure of a stochastic scheduling problem allows it to be formulated as the following two-stage program with fixed recourse:

$$
\begin{aligned}
\textbf{SP}: \quad \text{Minimize} \quad & \mathcal{H}[\mathbf{q}(\boldsymbol{\theta})^{T}\mathbf{x} + \mathbf{h}^{T}\mathbf{y}_{\boldsymbol{\theta}}] & & \\
\text{subject to:} \quad & \mathbf{x} \in \Gamma, \mathbf{A}\mathbf{x} = \mathbf{b} & & \\
& \mathbf{E}\big(\boldsymbol{\theta}(\omega)\big)\mathbf{x} + \mathbf{R}\mathbf{y}_{\boldsymbol{\theta}(\omega)} = \mathbf{g}\big(\boldsymbol{\theta}(\omega)\big), & & \forall \boldsymbol{\theta}(\omega) \in \Theta \\
& \mathbf{y}_{\boldsymbol{\theta}(\omega)} \in \mathbb{R}_{+}^{n_2}, & & \forall \boldsymbol{\theta}(\omega) \in \Theta.
\end{aligned}
$$

The random vector $\boldsymbol{\theta}$ denotes the problem parameters defined over the associated probability space $(\Omega, \mathcal{F}, P)$. We use $\boldsymbol{\theta}(\omega)$, $\forall \omega \in \Omega$, to indicate a particular realization of $\boldsymbol{\theta}$, while the set Θ is the corresponding support. The function $\mathcal{H}[\cdot]$ denotes a nondecreasing distributional property (scalar) of the random variable $[\cdot]$. The vector $\mathbf{x} \in \mathbb{R}^{n_1}$ consists of the first stage decision variables. Due to nonanticipativity, $\mathbf{x}$ is assumed to be invariant of the realizations of $\boldsymbol{\theta}$. On the other hand, the vector $\mathbf{y}_{\boldsymbol{\theta}(\omega)} \in \mathbb{R}_{+}^{n_2}$, which denotes the second stage variables, takes on values that are determined after a realization $\boldsymbol{\theta}(\omega)$ of $\boldsymbol{\theta}$ is observed. The coefficient vectors/matrices $\mathbf{q}(\boldsymbol{\theta})$, $\mathbf{g}(\boldsymbol{\theta})$, and $\mathbf{E}(\boldsymbol{\theta})$ are assumed to be affine functions of $\boldsymbol{\theta}$. Possible integrality restrictions on $\mathbf{x}$ are included in the description of the set $\Gamma \subseteq \mathbb{R}^{n_1}$ (for example, $\Gamma \equiv \{0,1\}^{n_1}$, if all the $\mathbf{x}$-variables are required to be binary-valued). Without loss of

generality, the above formulation can be simplified to the following (see Appendix B):

$$
\begin{aligned}
\mathbf{SP_\theta}: \quad & \text{Minimize} && \mathcal{H}[\mathbf{h}^T\mathbf{y}_{\boldsymbol{\theta}}] && \\
& \text{subject to:} && \mathbf{x} \in \Gamma, \mathbf{A}\mathbf{x} = \mathbf{b} && \\
& && \mathbf{E}\big(\boldsymbol{\theta}(\omega)\big)\mathbf{x} + \mathbf{R}\mathbf{y}_{\boldsymbol{\theta}(\omega)} = \mathbf{g}\big(\boldsymbol{\theta}(\omega)\big), && \forall \boldsymbol{\theta}(\omega) \in \Theta \\
& && \mathbf{y}_{\boldsymbol{\theta}(\omega)} \in \mathbb{R}_+^{n_2}, && \forall \boldsymbol{\theta}(\omega) \in \Theta.
\end{aligned}
$$

Note that the recourse matrix $\mathbf{R}$ is assumed to be deterministic (hence, the terminology *fixed recourse*). In addition, we assume *relative complete recourse*, i.e., for any given $\mathbf{x} \in \{\mathbf{x} \in \Gamma: \mathbf{A}\mathbf{x} = \mathbf{b}\}$ and any realization of $\boldsymbol{\theta}$, there exists a feasible solution $\mathbf{y}$ to the *second stage recourse problem* (noting that $\mathcal{H}[.]$ is nondecreasing):

$$
\begin{aligned}
Q(\mathbf{x}, \boldsymbol{\theta}) \equiv \quad & \text{Minimize} && \mathbf{h}^T\mathbf{y} \\
& \text{subject to:} && \mathbf{R}\mathbf{y} = \mathbf{g}(\boldsymbol{\theta}) - \mathbf{E}(\boldsymbol{\theta})\mathbf{x} \\
& && \mathbf{y} \in \mathbb{R}_+^{n_2}.
\end{aligned} \tag{4.1}
$$

Based on the above assumption, the original problem can be re-written as follows:

$$
\mathbf{SP_\theta}: \quad \text{Minimize} \quad \{\mathcal{H}[Q(\mathbf{x}, \boldsymbol{\theta})] : \mathbf{A}\mathbf{x} = \mathbf{b};\ \mathbf{x} \in \Gamma\}.
$$

Proposition 4.1: *For any given first stage variable* $\mathbf{x}$*, the value function* $Q(\mathbf{x}, \boldsymbol{\theta})$ *is convex w.r.t.* $\boldsymbol{\theta}$.

Proof: Follows directly from linear programming duality, noting that $\mathbf{g}(\boldsymbol{\theta})$ and $\mathbf{E}(\boldsymbol{\theta})$ are affine functions of $\boldsymbol{\theta}$. ■

Note that a result similar to the above proposition is given in Section 3.1.c of Birge and Louveaux (1997), but our problem setup is slightly different in that the problem parameter $\boldsymbol{\theta}$ is explicitly considered. In the context of stochastic scheduling, the first stage variables usually represent assignment and sequencing decisions. Proposition 4.1 establishes the convexity of the second stage value function with respect to the values of the random parameters.

Next, we establish bound values for $\mathcal{H}[Q(\mathbf{x},\boldsymbol{\theta})]$. For the case of expected value (i.e., $\mathcal{H}[\cdot] \equiv E[\cdot]$), we consider an alternative distribution $\boldsymbol{\xi}$ of problem parameters, such that $\boldsymbol{\xi} \leq_{cx} \boldsymbol{\theta}$, where the relation $\leq_{cx}$ denotes the following *convex order* between probability distributions:

$$\boldsymbol{\xi} \leq_{cx} \boldsymbol{\theta} \Leftrightarrow E[u(\boldsymbol{\xi})] \leq E[u(\boldsymbol{\theta})], \forall \text{ convex function } u(.).$$

Consequently, noting Proposition 4.1, we have

$$\mathcal{H}[Q(\mathbf{x},\boldsymbol{\xi})] = E[Q(\mathbf{x},\boldsymbol{\xi})] \leq E[Q(\mathbf{x},\boldsymbol{\theta})] = \mathcal{H}[Q(\mathbf{x},\boldsymbol{\theta})].$$

Hence, substituting $\boldsymbol{\xi}$ for the original distribution of $\boldsymbol{\theta}$ provides a valid lower bound. For the case of conditional-value-at-risk (i.e., $\mathcal{H}[\cdot] \equiv CVaR[\cdot]$; see Section 3.1), we consider an alternative random vector $\boldsymbol{\chi}$ of problem parameters that satisfies $\boldsymbol{\chi} \leq_{icx} \boldsymbol{\theta}$, where the relation $\leq_{icx}$ denotes the following *increasing* (nondecreasing) *convex order*:

$$\boldsymbol{\chi} \leq_{icx} \boldsymbol{\theta} \Leftrightarrow E[u(\boldsymbol{\chi})] \leq E[u(\boldsymbol{\theta})], \forall \text{ nondecreasing convex function } u(.).$$

To derive a valid lower bound for CVaR, we restrict our discussion to the special case that $Q(\mathbf{x},\boldsymbol{\theta})$ is a nondecreasing function of $\boldsymbol{\theta}$, as for example, when an increment in $\boldsymbol{\theta}$ makes the problem that evaluates Q more restricted. Note that this assumption generally holds true for the regular performance measures of scheduling problems.

Proposition 4.2: *If $Q(\boldsymbol{x},\boldsymbol{\theta})$ is a nondecreasing function of $\boldsymbol{\theta}$ and $\boldsymbol{\chi} \leq_{icx} \boldsymbol{\theta}$, then $Q(\boldsymbol{x},\boldsymbol{\eta}) \leq_{icx} Q(\boldsymbol{x},\boldsymbol{\theta})$.*

Proof: Given any nondecreasing convex function $u(.)$, since $Q(\mathbf{x}, \boldsymbol{\theta})$ is nondecreasing and convex, the composite function $u(Q(\mathbf{x},\boldsymbol{\theta}))$ is also nondecreasing and convex w.r.t. $\boldsymbol{\theta}$ (Bazaraa, Sherali, and Shetty, 2006). Since $\boldsymbol{\chi} \leq_{icx} \boldsymbol{\theta}$, we therefore have

$$E\big[u\big(Q(\boldsymbol{\chi},\boldsymbol{\eta})\big)\big] \leq E\big[u\big(Q(\mathbf{x},\boldsymbol{\theta})\big)\big],$$

which yields

$$Q(\mathbf{x},\boldsymbol{\chi}) \leq_{icx} Q(\mathbf{x},\boldsymbol{\theta}). \blacksquare$$

Hürlimann (2002) has shown that CVaR preserves the increasing convex order. Hence, we have the following corollary:

Corollary 4.1: *Under the same assumptions* of *Proposition 4.2*, the fo*llowing inequality holds:*

$$CVaR[Q(\mathbf{x}, \boldsymbol{\chi})] \leq CVaR[Q(\mathbf{x}, \boldsymbol{\theta})].$$

This inequality leads to a lower bounding approach for conditional-value-at-risk (by substituting $\boldsymbol{\chi}$ for the original distribution of $\boldsymbol{\theta}$).

4.3 Discretization with Recursive Partitioning

In this section, we propose a procedure for constructing a discrete distribution $\boldsymbol{\xi}$ such that $\boldsymbol{\xi} \leq_{cx} \boldsymbol{\theta}$. Note that the same method also applies to the construction of $\boldsymbol{\chi}$ such that $\boldsymbol{\chi} \leq_{icx} \boldsymbol{\theta}$, since $\boldsymbol{\chi} \leq_{cx} \boldsymbol{\theta}$ implies $\boldsymbol{\chi} \leq_{icx} \boldsymbol{\theta}$.

Consider the probability space Ω associated with ω. A set $\{\Omega_1, \Omega_2, \dots, \Omega_K\}$ is called a *partition* of Ω if $\bigcup_{s=1}^{K} \Omega_s = \Omega$ and $\Omega_i \cap \Omega_j = \emptyset, \forall i \neq j$. Given a valid partition of Ω, we can define a discrete distribution with K possible outcomes such that $P\{\boldsymbol{\xi} = \boldsymbol{\xi}^s\} = P(\Omega_s)$, where $\boldsymbol{\xi}^s \equiv E[\boldsymbol{\theta}(\omega) | \omega \in \Omega_s]$, $\forall s = 1, \cdots, K$. In other words, $\boldsymbol{\xi}$ takes the conditional expectation of $\boldsymbol{\theta}$, with the probability that ω falls in a particular region.

Note that for any convex function $u(.)$, we have

$$E[u(\boldsymbol{\theta})] = \sum_{s=1}^{K} E\big[u\big(\boldsymbol{\theta}(\omega)\big) | \omega \in \Omega_s\big] P(\Omega_s)$$

$$\geq \sum_{s=1}^{K} u(E[\boldsymbol{\theta}(\omega) | \omega \in \Omega_s])\, P(\Omega_s) = \sum_{s=1}^{K} u(\boldsymbol{\xi}^s)\, P\{\boldsymbol{\xi} = \boldsymbol{\xi}^s\} = E[u(\boldsymbol{\xi})].$$

The above inequality holds by applying Jensen's inequality on the conditional expectation $E[\cdot\,|\omega \in \Omega_s]$. Hence, we have $\boldsymbol{\xi} \leq_{cx} \boldsymbol{\theta}$.

In case that the elements of the random vector $\boldsymbol{\theta}$ are stochastically independent, it is convenient to derive the partition of Ω as follows. Let m be the dimension of the random vector $\boldsymbol{\theta}$. For the o^{th} element, θ_o, we denote its probability density function by $f_o(\boldsymbol{\theta})$, its cumulative distribution function (CDF) by $F_o(\boldsymbol{\theta})$, and its inverse CDF by $F_o^{-1}(\mathbf{v})$, $\forall o = 1, \dots, m$. Suppose that the m-dimensional unit hypercube, $[0,1]^m$, is partitioned into K hyper-rectangular regions, each of which can be written as:

$$R_s \equiv \{\mathbf{v} = (v_1, \dots, v_m)^T : v_o \in I_o^s, \forall o = 1, \dots, m\}, \forall s = 1, \dots, K,$$

where $I_o^s = \begin{cases} [l_o^s, u_o^s], \text{if } l_o^s = 0 \\ (l_o^s, u_o^s], \text{if } l_o^s > 0 \end{cases} \forall s = 1, \dots, K; o = 1, \dots, m.$

Note that l_o^s and u_o^s are, respectively, the lower bound and upper bound of R_s along the o^{th} dimension. Consequently, we induce the following partition of the probability space Ω:

$$\Omega_s \equiv \{\omega : F_o(\theta_o(\omega)) \in I_o^s, \forall o = 1, \dots, m\}, \forall s = 1, \dots, K.$$

According to the assumption of mutual independence, we have $P(\Omega_s) = \prod_{o=1}^m P\{F_o(\theta_o) \in I_o^s\} = \prod_{o=1}^m (u_o^s - l_o^s)$. We also have

$$E[\theta_o(\omega) | \omega \in \Omega_s] = E[\theta_o | F_o(\theta_o) \in I_o^s]$$
$$= \frac{1}{u_o^s - l_o^s} \int_{F_o^{-1}(l_o^s)}^{F_o^{-1}(u_o^s)} z f_o(z) \mathrm{d}z = \frac{1}{u_o^s - l_o^s} \int_{l_o^s}^{u_o^s} F_o^{-1}(v) \mathrm{d}v.$$

Hence, the discrete distribution can be summarized as follows:

The random vector $\boldsymbol{\xi}$ has K possible outcomes. For each outcome $\boldsymbol{\xi}^s$, its o^{th} element is given by $\boldsymbol{\xi}_o^s = E[\theta_o(\omega) | \omega \in \Omega_s] = \frac{1}{u_o^s - l_o^s} \int_{l_o^s}^{u_o^s} F_o^{-1}(v) \mathrm{d}v, \forall s = 1, \dots, K, o = 1, \dots, m.$

The associated probability is $\pi_s \equiv P\{\boldsymbol{\xi} = \boldsymbol{\xi}^s\} = P(\Omega_s) = \prod_{o=1}^m (u_o^s - l_o^s)$.

In the above discussion, we assumed that a suitable partition of the unit cube, $[0,1]^m$, is available. This partition can be obtained recursively using the following algorithm:

Recursive-Partition Algorithm

Recursive Function:

Input:

$\widehat{\Omega} = [l_1, u_1] \times [l_2, u_2] \times \cdots \times [l_m, u_m]$: A hyper-rectangular region within $[0,1]^m$.

s: Number of sub-regions for partitioning the given hyper-rectangular region.

$\tilde{\mathbf{x}}$: A first stage incumbent solution of $\mathbf{SP_\theta}$†.

Function Procedure:

Step 1. If $s = 1$, record $\widehat{\Omega}$ as a region of the final partition and return.

---

† Such a solution can be obtained from a deterministic problem using the expected values of random parameters.

Step 2. For each $o = 1 \ldots m$,

a) Let $\tau_o = \frac{l_o + u_o}{2}$. Calculate the conditional expectation of problem parameters in the following two regions:

$R_o^L \equiv [l_1, u_1] \times \cdots \times [l_o, \tau_o] \times \cdots \times [l_m, u_m]$

and $R_o^U \equiv [l_1, u_1] \times \cdots \times [\tau_o, u_o] \times \cdots \times [l_m, u_m]$.

Record the results as $\boldsymbol{\xi}^L$ and $\boldsymbol{\xi}^U$, respectively.

b) Calculate the second stage value function according to the parameters calculated in Step 2a, and denote the results as $Q^L \equiv Q(\tilde{\mathbf{x}}, \boldsymbol{\xi}^L)$ and $Q^U \equiv Q(\tilde{\mathbf{x}}, \boldsymbol{\xi}^U)$. Record the absolute difference $\Delta_o \equiv |Q^L - Q^U|$.

Step 3. Choose $o^* \in \text{argmax}_{o=1\ldots m}\{\Delta_o\}$ as the partitioning dimension, breaking any ties in favor of the largest interval width, $(u_o - l_o)$.

Step 4. Decide the allocation of sub-regions to the lower and upper partitions:

Use Monte-Carlo method to generate 30 sample values of $\boldsymbol{\theta}$ in the lower partition, $R_{o^*}^L$, and the upper partition, $R_{o^*}^U$. Calculate the sample variance of $Q(\tilde{\mathbf{x}}, \boldsymbol{\theta})$ in these two partitions as σ_L^2 and σ_U^2, respectively. The numbers of sub-regions allocated to the lower partition (s_L) and the upper partition (s_U) are determined according to the following equations (taking the closest integer values): $s_L = \frac{\sigma_L}{\sigma_L + \sigma_U} s$; $s_U = \frac{\sigma_U}{\sigma_L + \sigma_U} s$.

Step 5. Call the recursive function with $\widehat{\Omega} = R_{d^*}^L$ and $s = s_L$ as the input arguments and also with $\widehat{\Omega} = R_{d^*}^U$ and $s = s_U$ as the input arguments.

Main Procedure:

Call the recursive function with $\widehat{\Omega} = [0,1]^m$ and $s = K$ as the input arguments.

Note that our recursive partitioning algorithm is similar to the Recursive Stratified Sampling (RSS) procedure for Monte Carlo Integration. It always bisects the volume of the current region (in terms of cumulative probability). Furthermore, the number of sub-regions is allocated according to the (estimated) standard deviation of objective values. Unlike the RSS, we choose the partitioning dimension based on the maximum difference of the Q^L- and Q^U-values resulting from the lower and upper partitions.

A similar discrete bounding approach has been proposed by Birge and Wets (1986), where they considered the case when the technology matrix (i.e., $\mathbf{E}$ in our formulation) is deterministic. Their partitioning procedure is different from ours in that the number of regions is increased progressively; that is, in each step, a region of the current partition is selected and divided into two regions. The selection of this target region for further partitioning is based on the optimal solution ($\hat{\mathbf{x}}$) to the scenario-based model derived in the previous step. Birge and Wallace (1986) extended the progressive partitioning procedure to deal with the case that the technology matrix is stochastic. They solved a linear program to minimize the Manhattan distance between the vectors $\mathbf{h}$ and $\mathbf{T}\hat{\mathbf{x}}$ (i.e., $\mathbf{g}$ and $\mathbf{E}\hat{\mathbf{x}}$, respectively, in our formulation), and the region that contains an optimal solution to this linear program was chosen as the target region for partitioning. To choose a partitioning direction, a directional subgradient of the value function $Q(\mathbf{x}, \boldsymbol{\theta})$ was evaluated at the two end points of all edges of the target region. The edge whose two end points have the largest difference in subgradient values indicates the nonlinearity of $Q(\mathbf{x}, \boldsymbol{\theta})$, and hence, the boundary of sub-regions was drawn perpendicular to this edge. Note that if there are m independent random parameters, the above procedure needs to compare $m \times 2^{m-1}$ edges to choose the partitioning direction. Hence, it is only practical for lower dimensional cases. Instead, we can detect the nonlinearity of $Q(\mathbf{x}, \boldsymbol{\theta})$ by comparing the directional subgradient on the line segments that pass through the center of the region along the m coordinate directions.

Furthermore, the distributions of random parameters were not explicitly considered in the partitioning procedures of Birge and Wets (1986) and Birge and Wallace (1986).

Using the discrete distribution $\boldsymbol{\xi}$ obtained from the Recursive-Partition procedure, we can reformulate Problem $\mathbf{SP_\theta}$ as the following scenario-based model:

$$
\begin{aligned}
\mathbf{SP_\xi}: \quad \text{Minimize} \quad & \mathcal{H}[\mathbf{h}^T\mathbf{y}_s] \\
\text{subject to:} \quad & \mathbf{x} \in \Gamma, \mathbf{Ax} = \mathbf{b} \\
& \mathbf{E}(\boldsymbol{\xi}^s)\mathbf{x} + \mathbf{Ry}_s = \mathbf{g}(\boldsymbol{\xi}^s), && \forall s = 1, \dots, K \\
& \mathbf{y}_s \in \mathbb{R}_+^{n_2}, && \forall s = 1, \dots, K.
\end{aligned}
$$

Note that, Problem $\mathbf{SP_\xi}$ provides a lower bound for Problem $\mathbf{SP_\theta}$, since the inequality $F[Q(\mathbf{x}, \boldsymbol{\xi})] \leq F[Q(\mathbf{x}, \boldsymbol{\theta})]$ holds for every feasible solution $\mathbf{x}$.

For the case of expectation (i.e., $\mathcal{H}[\cdot] \equiv E[\cdot]$), this problem can be further written as follows:

$$
\begin{aligned}
\mathbf{SP_\xi}: \quad \text{Minimize} \quad & \sum_{s=1}^{K} \pi_s \mathbf{h}^T\mathbf{y}_s \\
\text{subject to:} \quad & \mathbf{x} \in \Gamma, \mathbf{Ax} = \mathbf{b} \\
& \mathbf{E}(\boldsymbol{\xi}^s)\mathbf{x} + \mathbf{Ry}_s = \mathbf{g}(\boldsymbol{\xi}^s), && \forall s = 1, \dots, K \\
& \mathbf{y}_s \in \mathbb{R}_+^{n_2}, && \forall s = 1, \dots, K.
\end{aligned}
$$

For the case of conditional-value-at-risk (i.e., $\mathcal{H}[\cdot] \equiv CVaR[\cdot]$), the above problem can be formulated as follows:

$$
\begin{aligned}
\mathbf{SP_\xi}: \quad \text{Minimize} \quad & \eta + \frac{1}{1-\alpha}\sum_{s=1}^{K} \pi_s \mu_s \\
\text{subject to:} \quad & \mathbf{x} \in \Gamma, \mathbf{Ax} = \mathbf{b} \\
& \mathbf{E}(\boldsymbol{\xi}^s)\mathbf{x} + \mathbf{Ry}_s = \mathbf{g}(\boldsymbol{\xi}^s), && \forall s = 1, \dots, K \\
& \eta + \mu_s \geq \mathbf{h}^T\mathbf{y}_s, && \forall s = 1, \dots, K
\end{aligned}
$$

$$\mu_s \geq 0, \mathbf{y}_s \in \mathbb{R}_+^{n_2}, \qquad \forall s = 1, \dots, K.$$

Note that Problem $\mathbf{SP_\xi}$ is in the same form as Problem **MP** in Section 3.1.2. The only difference is how parameter values are determined for each scenario. We designate this Discrete Bounding Approximation method as Recursive DBA, in contrast with the Progressive DBA method proposed by Birge and Wallace (1986).

4.4 Lower Bounds for a Single-Machine Scheduling Problem

In this section, we consider the single machine total weighted tardiness problem and compare lower bounds generated by the Sample Average Approximation (SAA) method and our Recursive DBA procedure. We consider the case of the expectation objective function. The problem formulation is given as follows:

Parameters:

J: Set of jobs; $J = \{1, \dots, n\}$;

S: Set of scenarios; $S = \{1, \dots, K\}$;

p_j^s: Processing time of job j in scenario s, $\forall j \in J, s \in S$;

d_j: Due date of job j, $\forall j \in J$;

w_j: Weight (importance factor) of job j, $\forall j \in J$;

Decision variables:

$z_{i,j}$: Indirect job precedence; $z_{i,j} = 1$ if job i starts before job j, and 0 otherwise.

c_j^s: Completion time of job j in scenario s, $\forall j \in J, s \in S$.

t_j^s: Tardiness of job j in scenario s, $\forall j \in J, s \in S$.

Minimize $$G_K = \sum_{s \in S} \pi_s \sum_{j \in J} w_j t_j^s$$

subject to: $$\sum_{i \in J \setminus \{j\}} p_i^s z_{i,j} + p_j^s \leq c_j^s, \qquad \forall\, s \in\ S, j \in J$$

$$
\begin{aligned}
t_j^s + d_j \geq c_j^s, \quad & \forall\, s \in S, j \in J \\
z_{i,j} + z_{j,i} = 1, \quad & \forall\, i \neq j \in J \\
z_{i,j} + z_{j,k} + z_{k,i} \leq 2, \quad & \forall\, i \neq j \neq k \in J \\
c_j^s \geq 0, t_j^s \geq 0, \mu_s \geq 0, \quad & \forall\, s \in S, j \in J \\
z_{i,j} \in \{0,1\}, \quad & \forall\, i \neq j \in J.
\end{aligned}
$$

Note that the above formulation is the same as Problem SM-TWTP in Section 3.2, except that expected value of total weighted tardiness is minimized, instead of CVaR. The processing times $\{p_j\}$ are the only random parameters, which constitute vector $\boldsymbol{\theta}$. The first stage variables are $\{z_{i,j}\}$, and the second stage variables are $\{c_j^s, t_j^s\}$. Since the coefficients of the second stage variables do not depend on $\boldsymbol{\theta}$, we have a fixed recourse problem.

Next, we compare lower bounds generated by the SAA method and our Recursive DBA procedure over randomly generated problem instances with different processing time distributions. Each problem instance consists of 15 jobs. For any job j, its expected processing time $\mu_j \equiv E[p_j]$ was generated from a continuous uniform distribution: $\mathcal{U}(3,20)$. Its due date d_j was generated from

$$
\mathcal{U}\left(1 - \mathrm{TF} - \frac{\mathrm{RDD}}{2}, 1 - TF + \frac{\mathrm{RDD}}{2}\right) \cdot \sum_{j \in J} \mu_j,
$$

where $\mathrm{TF} = 0.8$ and $\mathrm{RDD} = 0.4$ are respectively the Tardiness Factor and the Range of Due Dates parameters (Koulamas, 1996). The weights of jobs $\{w_j\}$ were selected according to a discrete uniform distribution on $\{1, \ldots, 5\}$. With μ_j fixed as described above, two different distributions of p_j were considered:

1) **Uniform**: $p_j \sim \mathcal{U}(a_j, b_j)$

   Generate parameter r_j from $\mathcal{U}(0,1)$;

   Let $a_j = r_j \mu_j$ and $b_j = 2\mu_j - a_j$. Note that $E[p_j] = \frac{a_j + b_j}{2} = \mu_j$.

2) **Truncated Normal**: $p_j \sim \max\left(\mathcal{N}\left(\hat{\mu}_j, \left(\hat{\sigma}_j\right)^2\right), 0\right)$

Generate parameter r_j from $\mathcal{U}(0.2,0.5)$; Let $\hat{\sigma}_j = r_j\mu_j$;

Determine the value of $\hat{\mu}_j$ such that $E[p_j] = \hat{\mu}_j + \frac{\phi(-\hat{\mu}_j/\hat{\sigma}_j)}{1-\Phi(-\hat{\mu}_j/\hat{\sigma}_j)}\hat{\sigma}_j = \mu_j$, where $\phi(.)$ and $\Phi(.)$ are the PDF and CDF of standard normal distribution, respectively.

For either of these processing time distributions, we generated a set of 10 problem instances. In the SAA method, we used Monte Carlo method to generate K scenarios. Problem TWT was then solved to optimality to determine a realization of G_K^*. By repeating the above scenario generation and solution procedure 30 times, we obtained 30 samples of G_K^*-values for the same problem instance. We then applied the bootstrapping method with 2000 bootstrapping samples to establish a 95% confidence interval, $[lb_{SAA}, ub_{SAA}]$, for $E[G_K^*]$. Since the lower bound value, lb_{SAA}, is subject to random variation induced by the Monte Carlo method, we repeated the overall procedure 10 times to obtain 10 realizations of lb_{SAA} values for the same problem instance. In other words, the procedure of generating scenarios and then solving Problem TWT was repeated for $30 \times 10 = 300$ times. The results were grouped into 10 batches to obtain 10 values of lb_{SAA}. The time required to process each batch was recorded as the corresponding CPU time.

In the Recursive DBA method, we first solved the deterministic version of Problem TWT with expected job processing times. Next, we used the optimal sequence obtained as the incumbent solution $\tilde{\mathbf{x}}$ in the Recursive-Partition algorithm. Using the scenarios generated by the Recursive-Partition algorithm, we solved Problem TWT to obtain a valid lower bound on the optimal objective value of the original stochastic program. We denote this lower bound by lb_{DBA}. Since there are random elements in the Recursive-Partition algorithm, we also repeated the DBA procedure 10 times to obtain 10 values of lb_{DBA} values for the same problem instance.

Both of the above methods were implemented in MATLAB 7.6 and tested on a computer with a 3.6 GHz CPU and 3 GByte memory, where numerical results were obtained for 100 and 400 scenarios. Tables 4.1-4.4 provide results of comparisons between the SAA and Recursive DBA methods. Note that, since both lb_{SAA} and lb_{DBA} are random, their sample

mean and sample variance (over the 10 sample values) are recorded for comparison. Better bound values and shorter CPU times are highlighted in bold. For the cases of uniform distribution, Recursive DBA consistently generated better lower bound values (see Tables 4.1and 4.3); and, on average, it required about 11-fold less CPU time when 100 scenarios were used and about 21-fold less CPU time when 400 scenarios were used, over the corresponding values for SAA. For the case of truncated normal distribution, the results were slightly mixed, but strongly in favor of the Recursive DBA method on average (see Tables 4.2 and 4.4). When 100 scenarios were used, the average lower bound value generated by Recursive DBA was slightly better and the average CPU time required by Recursive DBA was about 7-fold less over the corresponding values for SAA. Further, when 400 scenarios were used, the average lower bound value generated by Recursive DBA was slightly lower than that for SAA, but, on average, it required about 12-fold less CPU time than SAA.

Table 4.1: Comparison between SAA and recursive DBA methods on problems with Uniform distributions and 100 scenarios.

Problem	SAA			Recursive DBA		
	$E[lb_{SAA}]$	$Var[lb_{SAA}]$	Average CPU Time (seconds)	$E[lb_{DBA}]$	$Var[lb_{DBA}]$	Average CPU Time (seconds)
1	922.77	0.75	115.35	**923.87**	0.00	**31.72**
2	886.81	0.49	**28.96**	**887.52**	0.00	29.33
3	1825.63	4.67	64.40	**1830.43**	0.00	**30.88**
4	2174.47	4.85	233.76	**2176.96**	0.00	**37.11**
5	1565.31	1.59	**17.00**	**1568.03**	0.00	28.90
6	1997.50	6.75	274.49	**2001.61**	0.00	**34.35**
7	1583.49	1.09	110.32	**1585.55**	0.00	**31.81**
8	1117.95	1.01	78.32	**1118.91**	0.00	**31.08**
9	1138.47	1.82	**21.76**	**1140.90**	0.00	29.14
10	1459.41	6.25	3316.33	**1461.96**	0.00	**114.79**
Average	1467.18	2.93	426.07	**1469.57**	0.00	**39.91**

Table 4.2: Comparison between SAA and recursive DBA methods on problems with truncated Normal distributions and 100 scenarios.

Problem	SAA			Recursive DBA		
	$E[lb_{SAA}]$	$Var[lb_{SAA}]$	Average CPU Time (seconds)	$E[lb_{DBA}]$	$Var[lb_{DBA}]$	Average CPU Time (seconds)
1	933.83	7.41	108.44	**936.39**	0.00	**65.88**
2	891.89	18.77	**19.56**	**894.92**	0.00	63.70
3	1843.45	66.53	74.85	**1852.49**	0.01	**65.36**
4	2192.31	34.34	211.39	**2195.28**	0.01	**72.12**
5	1572.23	43.92	**20.19**	**1579.29**	0.00	63.24
6	2016.15	56.87	393.05	**2029.45**	0.00	**75.66**
7	**1605.85**	29.95	110.66	1602.22	0.03	**66.34**
8	**1129.54**	8.03	66.27	1128.73	0.00	**65.63**
9	1156.85	26.21	**38.42**	**1159.90**	0.01	64.08
10	1492.66	12.79	3946.95	**1499.06**	0.01	**159.17**
Average	1483.48	30.48	498.98	**1487.77**	0.01	**76.12**

Table 4.3: Comparison between SAA and recursive DBA methods on problems with Uniform distributions and 400 scenarios.

Problem	SAA			Recursive DBA		
	$E[lb_{SAA}]$	$Var[lb_{SAA}]$	Average CPU Time (seconds)	$E[lb_{DBA}]$	$Var[lb_{DBA}]$	Average CPU Time (seconds)
1	923.59	0.70	1506.58	**924.15**	0.00	**149.20**
2	887.69	0.13	329.68	**887.73**	0.00	**124.67**
3	1828.38	2.26	656.72	**1830.56**	0.00	**131.42**
4	2176.34	0.26	2966.34	**2177.19**	0.00	**203.03**
5	1567.87	0.89	209.73	**1568.07**	0.00	**119.58**
6	2000.42	3.43	2970.52	**2001.74**	0.00	**177.16**
7	1585.13	0.96	1490.80	**1585.74**	0.00	**149.21**
8	1118.29	0.23	873.53	**1118.95**	0.00	**139.01**
9	1140.26	0.56	330.82	**1140.97**	0.00	**145.36**
10	1461.25	1.34	35433.80	**1461.99**	0.00	**936.01**
Average	1468.92	1.08	4676.85	**1469.71**	0.00	**227.46**

Table 4.4: Comparison between SAA and recursive DBA methods on problems with truncated Normal distributions and 400 scenarios

Problem	SAA			Recursive DBA		
	$E[lb_{SAA}]$	$Var[lb_{SAA}]$	Average CPU Time (seconds)	$E[lb_{DBA}]$	$Var[lb_{DBA}]$	Average CPU Time (seconds)
1	**938.64**	1.30	1260.91	937.24	0.00	**271.89**
2	895.21	2.58	**161.88**	**895.88**	0.00	243.10
3	1850.87	12.60	716.23	**1855.06**	0.01	**259.54**
4	**2200.79**	5.41	2643.89	2197.66	0.01	**359.66**
5	1578.82	3.59	**281.61**	**1579.85**	0.00	304.74
6	2026.12	12.88	3860.20	**2030.39**	0.00	**395.51**
7	**1607.57**	3.80	1273.49	1604.85	0.00	**329.13**
8	**1134.53**	6.49	821.38	1130.18	0.01	**345.85**
9	**1163.58**	8.59	370.58	1160.82	0.00	**311.47**
10	1499.25	16.97	38081.93	**1500.85**	0.01	**1451.41**
Average	**1489.54**	7.42	4947.21	1489.28	0.00	**427.23**

To provide a more detailed comparison, we focused on problem instance 1 for the truncated normal case, and experimented with different numbers of scenarios (K) for both the SAA and Recursive DBA methods. Values of $K \in \{25, 100, 225, 400, 625, 900, 1225, 1600\}$ were used for Recursive DBA; and values of $K \in \{25, 100, 225, 400, 625\}$ were used for SAA. The corresponding results are plotted in Figure 4.1. Besides the average bound values (depicted as lines with markers), we have also plotted the minimum and the maximum of the 10 bound values obtained for either method. Note that the bound values for Recursive DBA (solid lines) have a very low variation with repeated runs, and hence, all these lines almost overlap. On the other hand, the bound values for SAA tend to have a large variation, which appears to decrease with an increment in K. Note that the bound values for the Recursive DBA method dominate for lower values of K (corresponding to lower CPU times).

So far, we have used the sample size of 30 (G_K^*-values) to calculate lb_{SAA}. To investigate the impact of sample size on the value of lb_{SAA}, we re-grouped the 300 G_K^*-values into

batches of 100 and 300, respectively, and re-calculated the bound values. The results are depicted in Figure 4.2. Note that the batch size impacts both the bound values and the CPU times. However, the Recursive DBA method continues to dominate SAA for lower values of K (corresponding to lower CPU times).

We further compared the performance of our Recursive DBA method with the progressive partition procedure of Birge and Wallace (1986) (designated as Progressive DBA). For Birge and Wallace's method, the number of scenarios was increased iteratively, and Problem TWT was resolved each time a further partitioning of the sample space was obtained. The numerical results are summarized in Tables 4.5 and 4.6. Although the comparison was conducted with a target scenario number of 100 for both methods, the Progressive DBA approach terminated prematurely before 100 scenarios were generated, due to a failure in finding a suitable partitioning point. Hence, this method required a smaller CPU time on average for the case of truncated normal distribution. However, our Recursive DBA method consistently provided tighter lower bound values for all the problem instances tested.

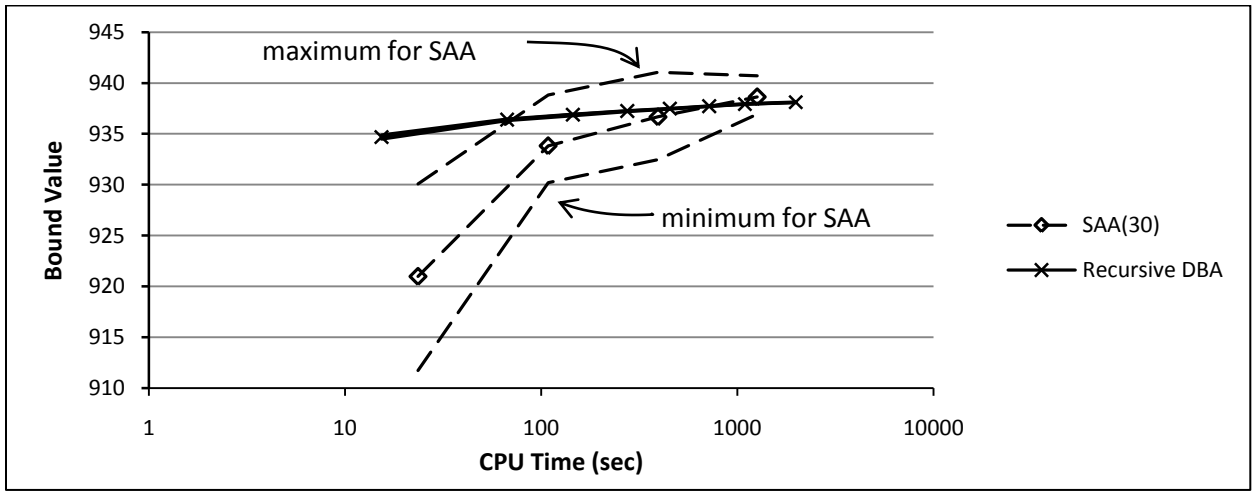


Figure 4.1: Bound values and CPU times required for problem instance 1 under truncated Normal distributions.

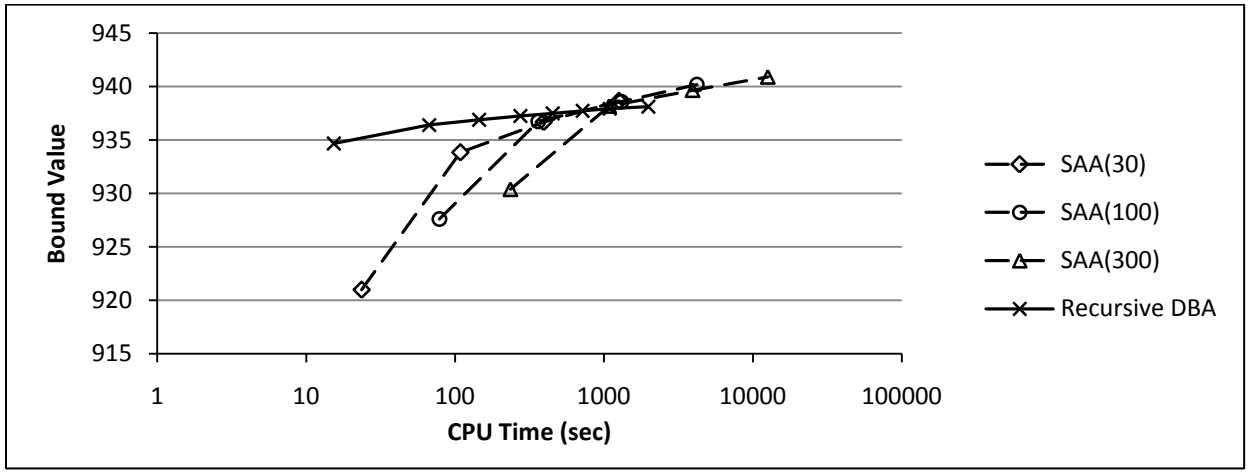


Figure 4.2: Bound values and CPU times required for problem instance 1 under truncated Normal distributions with different batch sizes.

Table 4.5: Comparison of progressive DBA and recursive DBA for problems with Uniform distributions.

Problem	Progressive DBA			Recursive DBA (100 scenarios)	
	Lower Bound	CPU Time (seconds)	Number of Scenarios	$E[lb_{DBA}]$	Average CPU Time (seconds)
1	921.74	25.48	34	**923.87**	31.72
2	886.54	23.75	34	**887.52**	29.33
3	1829.48	21.57	35	**1830.43**	30.88
4	2174.41	40.30	34	**2176.96**	37.11
5	1567.45	11.01	34	**1568.03**	28.90
6	2000.43	44.41	34	**2001.61**	34.35
7	1583.36	23.97	34	**1585.55**	31.81
8	1118.00	22.23	34	**1118.91**	31.08
9	1140.28	17.40	34	**1140.90**	29.14
10	1457.75	256.94	28	**1461.96**	114.79
Average	1467.94	48.71	33.5	**1469.57**	39.91

Table 4.6: Comparison progressive DBA and recursive DBA for problems with truncated Normal distributions.

Problem	Progressive DBA			Recursive DBA (100 scenarios)	
	Lower Bound	CPU Time (seconds)	Number of Scenarios	$E[lb_{DBA}]$	Average CPU Time (seconds)
1	925.79	35.78	47	**936.39**	65.88
2	888.61	4.03	5	**894.92**	63.70
3	1825.39	0.44	1	**1852.49**	65.36
4	2179.77	10.64	12	**2195.28**	72.12
5	1575.94	3.78	12	**1579.29**	63.24
6	1996.27	0.70	1	**2029.45**	75.66
7	1582.56	0.47	1	**1602.22**	66.34
8	1126.70	44.78	60	**1128.73**	65.63
9	1144.73	7.66	8	**1159.90**	64.08
10	1473.35	14.00	2	**1499.06**	159.17
Average	1471.75	12.23	16.3	**1487.77**	76.12

Chapter 5

Primary Pharmaceutical Manufacturing Scheduling Problem

The pharmaceutical supply chain consists of two key manufacturing phases: primary manufacturing and secondary manufacturing. *Primary manufacturing* is associated with the production of active pharmaceutical ingredients (APIs), which are the active substances constituting a drug. *Secondary manufacturing* involves the mixing of APIs with inert materials to make final products. The competitiveness of a pharmaceutical supply chain, to a large extent, depends on the performance of its internal business processes (Shah, 2004) , where the responsiveness of primary manufacturing to the requirements of the secondary phase dictates the overall effectiveness of the supply chain. This responsiveness is directly impacted by the scheduling of production in a primary pharmaceutical manufacturing facility, which is the problem that we address in this chapter, and designate it as the primal pharmaceutical manufacturing scheduling problem (PPMSP).

5.1 Problem Statement

A typical primary manufacturing facility hosts multiple processing bays, each of which

consists of a serial arrangement of four processing stages: reactor, centrifuge, crystallizer, and dryer. These processing stages are equipped with containers of limited capacities. Since there is no additional storage space between these containers, they also act as temporary storage devices for processed material, and the production of a product in a bay has to be carried out in a consecutive manner without interruption. Hence, we can regard the entire bay as one processing device, working in a *batch production mode*. In some cases, the production of an API (called as *end product*) needs multiple steps to complete. Each of these steps occupies a processing bay entirely, and the output is stored in a separate storage device. We call the outputs, which will be consumed later to produce other products, as *intermediate products*. Both the end products and the intermediate products incur inventory costs.

From a primary manufacturing facility's point of view, the requirements specified by the secondary manufacturing phase are regarded as external demands. For the convenience of production planning, the time horizon is divided into multiple periods such that demands only need to be satisfied at the end of each period. Consequently, inventory costs are determined on a period-by-period basis. In other words, the remaining amounts of end products and intermediate products at the end of each period become inventories, and they incur holding cost due to storage expenses and tied-up capital.

Due to the availability of a limited number of processing bays, multiple products may need to share the same bay during production. The definition of a product family has two connotations: firstly, it reflects the Bill-of-Materials (BOM) relation, as an intermediate product always belongs to the family of its end product (see Figure 5.1 for an example); and secondly, a production changeover between different products may necessitate significant time and cost, depending on the product families involved. In particular, production batches of the same family can be scheduled consecutively in a bay without incurring a significant setup time. Hence, we only consider changeover setup time between batches of different families, and define a *production campaign* as a series of production batches that belong to the same product family. In other words, production batches are grouped into campaigns

according to their family affiliations. Changeover setups are required between campaigns of different families. Note that changeover operation between two campaigns depends on the different types of families involved, and is sequence-dependent.

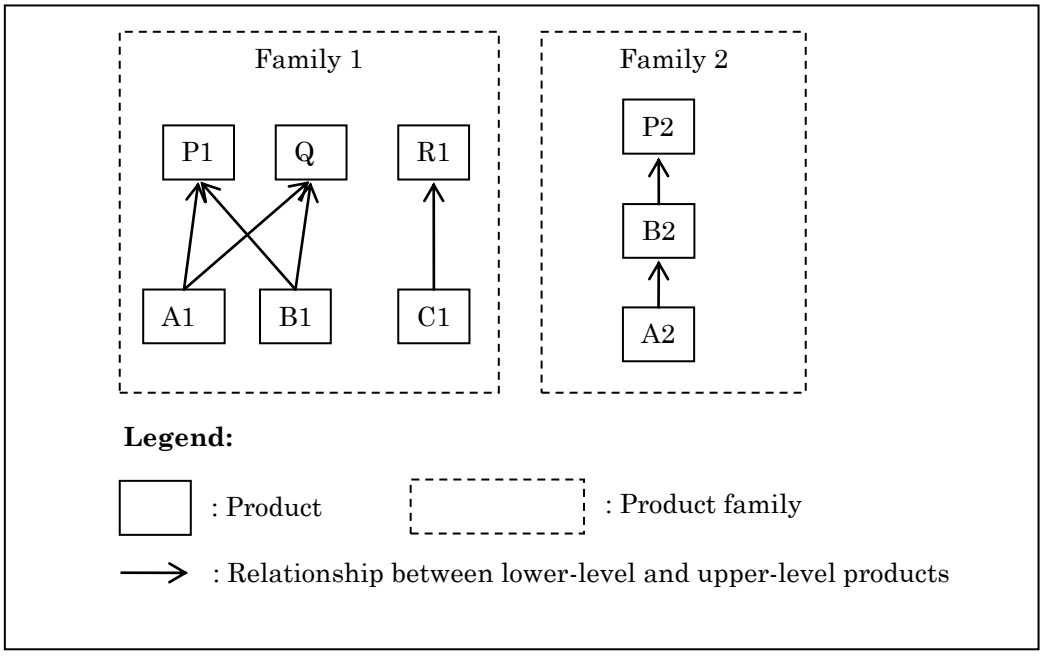


Figure 5.1: Two families with their products and BOM relations

The primal pharmaceutical manufacturing scheduling problem can be formally described as follows:

Production Configuration:

1) There are multiple flexible machines (or bays) operating in parallel.
2) There are multiple levels of products. Low-level products serve as raw materials for high-level products. Products of different levels may share the same bay.
3) Each product can be produced in a limited set of compatible bays. Production is completed in batches, with each batch producing only one type of product. Batch

processing time does not depend on the production amount. But batch processing time and maximum batch size vary by bays and products.

4) The time horizon is divided into multiple periods. Demands must be satisfied at the end of each period. Excessive amount of production is stored as inventory which incurs period-wise holding costs.
5) Products are grouped into families such that Bill of Materials (BOM) relations exist only among the items of a product family
6) Changeover setup times are required between batches of different families. A sequence of batches of products of the same family is called a production campaign.

Objective:

Minimize total cost, which consists of

1) Inventory holding cost, which differs by products;
2) Changeover setup cost, which is proportional to the setup time.

Decisions:

1) Amount of production for each product during each period;
2) Number of batches of each product to produce in each bay during each period;
3) Number of campaigns of each family in each bay during each period;
4) Campaign setup sequences.

Additional Features:

1) To avoid contamination, non-adjacency requirements exist between some families.
2) The length of a campaign (in terms of number of batches) may have a managerial upper bound.
3) Due to the above two features, it is possible to schedule multiple campaigns pertaining to a family in the same bay during a period.
4) Changeover setup times are sequence-dependent;
5) Setups are allowed to straddle two or more periods. These are called *carryover setups* (see Figure 5.2).

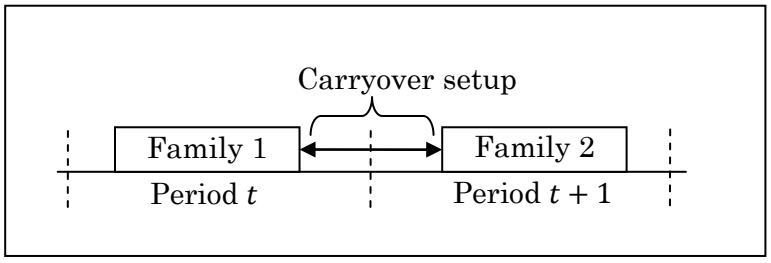


Figure 5.2: An example of carryover setup

5.2 Literature Review

As an integrated lot-sizing and sequencing problem, the PPMSP is closely related to the Capacitated Lot-sizing Problem (CLSP). An important feature of the CLSP is that the production cost in a given period is a concave function of the production amount. In most cases, this concavity is caused by a fixed charge (or setup cost) that is incurred due to the necessary preparation for production. However, setup costs for the PPMSP are dictated by campaign sequences in a complex manner, instead of being constant. Furthermore, the campaign sequencing problem in the PPMSP is the High Multiplicity Asymmetric Traveling Salesman Problem (HMATSP) due to the fact that multiple campaigns of the same product family can occur in the same bay during a period. In this section, we provide a literature review on these two types of problems (i.e., CLSP and HMATSP). We also review references on the integration of lot-sizing and scheduling problems.

5.2.1 Lot-Sizing Problem

The lot-sizing problem is a classical production planning problem. The production activities are divided into multiple periods, which constitute a predefined partition of the planning horizon. The demand varies from one period to another and must be met at the end of each period. The objective function consists of the following components: setup cost that is incurred if production is scheduled in a period, production cost that is proportional to the

amount of production, and inventory cost that is incurred by the extra products stored at the end of each period. If we generalize the production cost to include the setup cost as well, it can be viewed as a concave function of the production amount.

The uncapacitated lot-sizing problem does not consider any restriction on the amount of production in a period, and some of the earliest research work in lot-sizing was devoted to the solution of this problem. The single item version has been successfully solved by Wagner and Whitin (1958) using a dynamic programming approach. Zangwill (1969) and Kalymon (1972) extended this result to include multiple echelons (or facilities) with an arborescence structure†. Their solution approaches were based on network flow-based formulations with concave cost functions. The dynamic programming method of Wagner and Whitin (1958) was later improved by Federgruen and Tzur (1991) and Wagelmans, van Hoesel, and Kolen (1992), reducing the time complexity from $O(|T|^2)$ to $O(|T|\log(|T|))$, where $|T|$ denotes the number of periods. They applied geometric techniques, which were generalized by van Hoesel, Wagelmans, and Moerman (1994) to solve problems with backlog cost and start-up cost that is incurred when the production status is switched from off to on.

As pointed out by Leung, Magnanti, and Vachani (1989), "adding capacity restrictions makes the (lot-sizing) problem much more difficult". Although the single item (or product) problem with a constant capacity was efficiently solved by Florian and Klein (1971) using a dynamic programming algorithm, the multiple-item version (also with constant capacity) has been shown to be NP-complete by Bitran and Yanasse (1982). Bitran and Yanasse (1982) also showed that, with less restrictive assumptions on the capacity of a single processor, various cases of the single item capacitated lot-sizing problem are NP-hard. To solve this problem effectively, various solution approaches have been developed. Barany (1984) studied the polyhedral structure of the single-item, uncapacitated lot-sizing problem,

† Arborescence is a rooted directed tree. In the inventory system considered by Kalymon (1972), it is further postulated that a unit of production at a non-root facility requires a unit of input from its predecessor.

and then, applied the derived inequalities to the multi-item capacitated version that contains it as a subproblem. Leung et al. (1989) investigated the multi-item single-resource problem with constant capacity. They derived valid inequalities from the polyhedral structure of the single item problem with constant capacity. Atamtürk and Muñoz (2004) derived bottleneck cover inequalities for the single-item lot-sizing problem with general capacity specification. Other forms of inequalities for the same problem have been derived by Pochet (1988), Miller, Nemhauser, and Savelsbergh (2000) and Loparic, Marchand, and Wolsey (2003). Besides adding valid inequalities to enhance the problem formulation, heuristic solution methods have also been developed. For example, Hindi (1995) proposed a tabu search method to solve a single item capacitated problem with start-up cost.

To solve the multi-item single-resource version of the lot-sizing problem, Graves (1982) proposed a hybrid decomposition scheme, which considers two subproblems (the aggregate planning subproblem and the detailed scheduling subproblem) and uses inventory consistency as the linking relationship. The values of corresponding Lagrangian multipliers were determined by an iterative feedback procedure. Diaby et al. (1992) used a Lagrangian relaxation-based heuristic procedure to generate near-optimal solutions for very large-scale capacitated lot-sizing problems with setup times and limited overtime. They applied Lagrangian relaxation on the resource requirement constraint and determined Lagrangian multipliers by using sub-gradient optimization. Besides Lagrangian relaxation-based approaches, other model reformulation and decomposition techniques have also been investigated. Eppen and Martin (1987) applied variable redefinition techniques and used a shortest route representation for the uncapacitated subproblem. Armentano, Franca, and de Toledo (1999) considered setup time, and reformulated the problem as a generalized network flow problem, which was solved with a branch-and-bound method. Degraeve and Jans (2007) applied a Dantzig-Wolfe reformulation and solved the problem using branch-and-price.

Additional features of the CLSP include multiple levels of products and explicit consideration of parallel machines. For the single-machine multi-level version, Billington,

McClain, and Thomas (1983) provided a Mixed-Integer Program (MIP) formulation and a problem reduction technique, called Product Structure Compression. Kimms (1997) solved a similar problem by using the demand shuffle technique, which is computationally efficient but yields suboptimal solutions. Stadtler (2003) considered multi-level production on parallel machines and developed a "Plant Location" formulation. Due to its complexity, a suboptimal solution approach was developed based on rolling schedules. Other relevant work on the parallel-machine version exists in the literature. However, since most of them also include sequence-dependent setups, a review of related literature is relegated to Section 5.2.3.

5.2.2 High Multiplicity Asymmetric Traveling Salesman Problem

The Asymmetric Traveling Salesman Problem (ATSP) is a classical combinatorial optimization problem, in which a traveling agent visits every one of a given set of cities exactly once and returns to the starting city after having completed the tour in minimum distance travelled. This problem structure often appears in a scheduling environment involving sequence-dependent setups, where these setups can be regarded as distances between cities. Mathematical programming formulations of the ATSP usually include two types of constraints. The assignment type constraints ensure that each city is visited exactly once, while the subtour elimination constraints (SECs) enforce a connected tour over the cities. Various types of SECs have been developed over the years. These include the Dantzig-Fulkerson-Johnson (DFJ) SECs based on cut-sets enumeration (Dantzig, Fulkerson, and Johnson, 1954), the Miller-Tucker-Zemlin (MTZ) SECs based on the rank-order of visits (Miller, Tucker, and Zemlin, 1960), the single commodity flow-based SECs of Gavish and Graves (1978), the multiple commodity flow-based SECs of Wong (1980), and the precedence-variable-based SECs of Sarin, Sherali, and Bhootra (2005), and Sherali, Sarin, and Tsai (2006). A comprehensive review on various SEC formulations can be found in Öncan, AltInel, and Laporte (2009).

The high multiplicity version of the ATSP allows multiple visits to the same city. This

feature arises when groups of identical jobs need to be scheduled in the presence of sequence-dependent setup times. To solve this problem, Cosmadakis and Papadimitriou (1984) proposed a column generation approach based on optimal Eulerian graphs. Grigoriev and van de Klundert (2006) developed an integer programming formulation with DFJ-type subtour elimination constraints, and investigated asymptotic properties of the problem parameterized by multiplicity values (i.e., numbers of visits). Sarin, Sherali, and Yao (2010) proposed another mathematical programming formulation using multiple commodity flow SECs, and have shown its superiority over the formulation of Grigoriev and van de Klundert (2006).

5.2.3 Integrated Lot-Sizing and Scheduling

When sequence-dependent setups are incorporated into a lot-sizing problem, the combined problem becomes even more difficult to solve. Baker and Muckstadt (1989) presented CHES problems, which pertains to scheduling of jobs on parallel machines involving sequence-dependent setup costs. Fleischmann (1994) considered a single-resource small-bucket model, in which only one product is allowed to be produced in a slot (period). Sequence-dependent setup costs were incorporated in the objective function, and lower bounds were obtained using Lagrangian relaxation. Kang, Malik, and Thomas (1999) solved the multiple-machine case, assuming sequence-dependent setup costs, but no setup time. Their solution approach was based on split-sequences and column generation. Haase and Kimms (2000) dealt with the single-machine case, considering both setup times and setup costs, and used efficient sequences to construct the model formulation. Their solution approach provided optimal solutions for small-sized problems. Meyr (2002) further considered the parallel-machine (heterogeneous) case with sequence-dependent setup times and costs. Their formulation relied on further subdivision of production periods into micro-periods. Heuristic methods were employed to solve the problem. Gupta and Magnusson (2005) developed both a mathematical programming formulation and a heuristic solution method for the single-machine version with setup times and costs. They considered additional

problem features like setup carryover and idle periods, but failed to correctly incorporate subtour elimination constraints into their formulation. Dogan and Grossmann (2006) also considered the single-machine case. Their treatment of setup changeover was based on predefined "slots" in each period. A bi-level decomposition procedure was adopted to solve the planning and scheduling problems in coordination. Fandel and Stammen-Hegene (2006) provided a mathematical programming formulation for a multi-product, multi-level job shop version of the integrated lot-sizing and scheduling problem. However, their nonlinear model is difficult to solve.

We note that the features of the PPMSP have not yet been fully addressed in the existing literature. Although features like parallel machines, multiple product levels and sequence-dependent setups have been previously investigated, yet a comprehensive consideration and the incorporation of high-multiplicity campaigns are new to the integrated lot-sizing and sequencing problem on hand.

5.3 A Basic MIP Formulation for PPMSP

We present a mixed integer linear programming formulation of the PPMSP as follows:

Sets:

- F: Set of real product families;
- $\hat{F}$: Set of (general) product families; $\hat{F} \equiv F \cup \{0\}$, where the dummy family "0" corresponds to the starting city in an HMATSP viewpoint;
- A_f: Set of families that are forbidden to immediately follow family f, $\forall f \in F$;
- N_f: Set of products in family $f, \forall f \in F$;
- M: Set of bays;
- N^k: Set of products that can be produced in bay $k, \forall k \in M$;
- M^i: Set of bays that can produce product i; $M^i \equiv \{k \in M: i \in N^k\}$;
- F^k: Set of real product families that can be produced in bay k, $\forall k \in M$;

$F^k \equiv \{f \in F: N_f \cap N^k \neq \emptyset\}$;

$\hat{F}^k$: Set of (general) product families that can be produced in bay , $\forall k \in M$; $\hat{F}^k \equiv F^k \cup \{0\}$;

T: Set of periods in the current planning horizon; $T = \{1, \dots, |T|\}$;

$\hat{T}$: Set of time periods including period "0", which precedes the first period of T; $\hat{T} = \{0\} \cup T$.

Note that the set of compatible bays M^i is defined for each product i, and hence, $i \in N_f$ and $f \in F^k \nRightarrow i \in N^k$.

Parameters:

σ: Setup cost per unit time;

f_0^k: Initial setup status of bay k (product family just processed at the end of period "0"), $\forall k \in M$;

$I_{f,i}^0$: Initial inventory of product i of family f, $\forall f \in F, i \in N_f$;

$h_{f,i}$: Holding cost per period for a unit amount of product i of family f, $\forall f \in F, i \in N_f$;

$d_{f,i}^t$: External demand for product i of family f at the end of period t,$\forall t \in T, f \in F, i \in N_f$;

$q_{f,i}^k$: Maximum batch size for product i of family f when produced in bay k, $\forall f \in F$, $i \in N_f, k \in M^i$;

$p_{f,i}^k$: Batch processing time for product i of family f when produced in bay k, $\forall f \in F$, $i \in N_f, k \in M^i$;

$s_{f,g}^k$: Changeover (setup) time required for a campaign of family g to follow a campaign of family f in bay k, $\forall k \in M, f, g \in F^k$. Note that $s_{f,f}^k = 0$,$\forall k \in M, f \in F^k$.

$c^{k,t}$: Time capacity of bay k during period t, $\forall k \in M, t \in T$;

$l_f^{k,t}$: Maximum campaign length (number of batches) of family f permitted in bay k during period t, $\forall k \in M, t \in T, f \in F^k$;

$m_f^{k,t}$: Maximum number of batches of family f that can be scheduled in bay k during period t without exceeding the time capacity, $\forall k \in M, t \in T, f \in F^k$;

$m_f^{k,t} = \left\lfloor \frac{c^{k,t}}{\min_{i \in N_f \cap N^k} \{p_{f,i}^k\}} \right\rfloor$;

$b_{f,i,j}$: Amount of product i required to produce one unit of product j in any bay (*BOM ratio*), $\forall f \in F, i \neq j \in N_f$.

Variables:

$I_{f,i}^t$: Inventory level of product i of family f at the end of period t,$\forall t \in T, f \in F, i \in N_f$;

$G_{f,i}^t$: Amount of product i of family f produced during period t, $\forall t \in T, f \in F, i \in N_f$;

$P_{f,i}^{k,t}$: Amount of product i of family f produced in bay k during period t, $\forall t \in T, f \in F$, $i \in N_f, k \in M^i$;

$W_{f,i}^{k,t}$: Number of batches of product i of family f produced in bay k during period t, $\forall t \in T, f \in F, i \in N_f, k \in M^i$;

$R_f^{k,t}$: Number of campaigns of family f scheduled in bay k during period t, $\forall k \in M$, $t \in T, f \in F^k$;

$Z_f^{k,t}$: Indicator for whether any product of family f is produced in bay k during period t, $\forall k \in M, t \in T, f \in F^k$;

$X_{f,g}^{k,t}$: Number of changeover setups from family f to family g in bay k during period t, $\forall k \in M, t \in T, f \neq g \in \hat{F}^k$. Note that self loops are not allowed (i.e.,$f \neq g$) within each period in order to ensure campaign length restrictions. In case that there is no campaign scheduled in bay k during period t, all corresponding X-variables take the value of zero.

$\gamma_{f,g}^{k,t_1,t_2}$: Indicator of a carryover setup for switching from family f at the end of period t_1 to family g at the start of period t_2, $\forall k \in M, t_1 < t_2 \in \hat{T}; f, g \in F^k$. Note that self loops (i.e., $f = g$) are allowed for carryover setups. Hence, any campaign length restriction is only enforced within a period. Also, note that the actual setup time may not fully cover the gap between the two production campaigns (see Figure 5.3).

$\beta^{k,t}$: Fraction of carryover setup time scheduled at the beginning of period t in bay k, $\forall k \in M, t \in T$ (see Figure 5.3);

$\alpha^{k,t}$: Fraction of carryover setup time scheduled at the end of period t in bay k, $\forall k \in M, t \in T$ (see Figure 5.3). Note that there is no carryover setup at the end of the planning horizon. Hence, $\alpha^{k,|T|} \equiv 0, \forall k \in M$.

$\lambda_{f,g}^{k,t,u}$: Amount of flow of commodity u on edge (f, g) in bay k during period t, $\forall k \in M, t \in T, u, g \in F^k, f \in \hat{F}^k; f \neq g$. The flow of commodity u originates from the dummy family "0" and ends at family u, $\forall u \in F^k$. This ensures the connectivity of the setup graph in bay k during period t, $\forall k \in M, t \in T$.

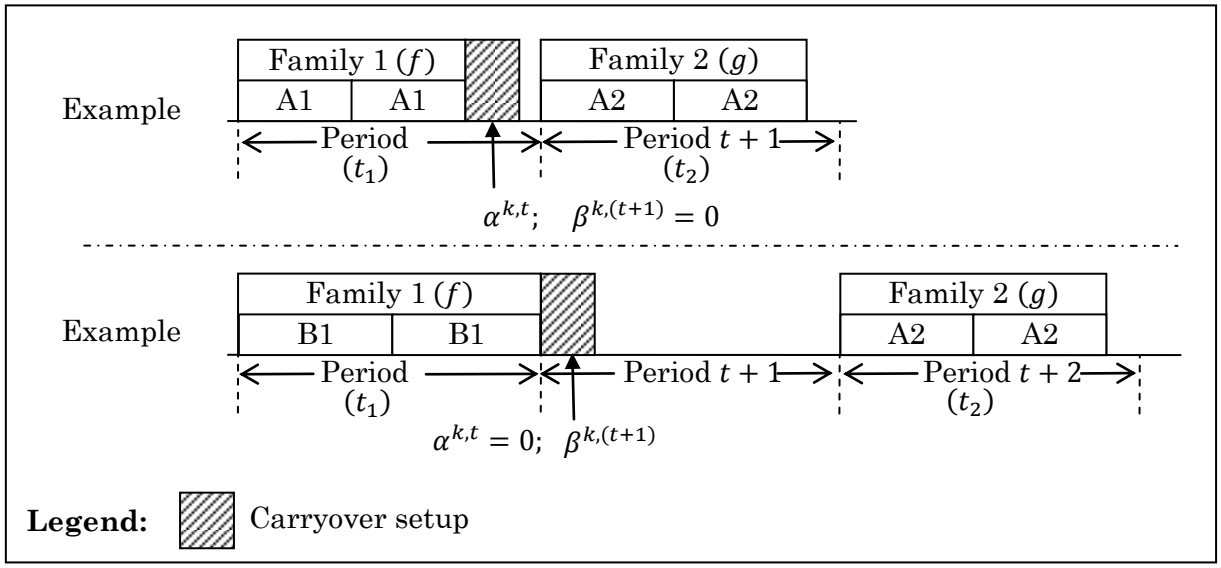


Figure 5.3: Two examples of carryover setups

Model PPMSP:

$$\text{Minimize} \sum_{t \in T} \sum_{f \in F} \sum_{i \in N_f} h_{f,i} I_{f,i}^t + \sigma \sum_{k \in M} \sum_{f \neq g \in F^k} s_{f,g}^k \left(\sum_{t \in T} X_{f,g}^{k,t} + \sum_{t_1 < t_2 \in \hat{T}} \gamma_{f,g}^{k,t_1,t_2} \right)$$

subject to:

- Flow Balance Constraint:

$$I_{f,i}^{t-1} + G_{f,i}^{t} = I_{f,i}^{t} + d_{f,i}^{t} + \sum_{j \in N_f} b_{f,i,j} G_{f,j}^{t},$$

$$\forall t \in T, f \in F, i \in N_f \tag{5.1}$$

- Production Allocation Constraint:

$$G_{f,i}^{t} = \sum_{k \in M^i} P_{f,i}^{k,t},$$

$$\forall t \in T, f \in F, i \in N_f \tag{5.2}$$

- Production Batch Constraint:

$$P_{f,i}^{k,t} \leq q_{f,i}^{k} W_{f,i}^{k,t},$$

$$\forall t \in T, f \in F, i \in N_f, k \in M^i \tag{5.3}$$

- Time Capacity Constraint:

$$\sum_{f \in F^k} \sum_{i \in N_f \cap N^k} p_{f,i}^{k} W_{f,i}^{k,t} + \sum_{f \neq g \in F^k} s_{f,g}^{k} X_{f,g}^{k,t} + \alpha^{k,t} + \beta^{k,t} \leq c^{k,t},$$

$$\forall t \in T, k \in M \tag{5.4}$$

- Sequencing Constraint:

$$\sum_{g \in F^k} \sum_{t_2 \in T} \gamma_{f_0^k,g}^{k,0,t_2} \leq 1, \qquad \forall k \in M \tag{5.5}$$

$$\gamma_{f,g}^{k,0,t_2} = 0,$$

$$\forall k \in M, t_2 \in T, f, g \in F^k; f \neq f_0^k \tag{5.6}$$

$$X_{0,g}^{k,t_2} = \sum_{f \in F^k} \sum_{\substack{t_1 \in T, \\ t_1 < t_2}} \gamma_{f,g}^{k,t_1,t_2},$$

$$\forall k \in M, t_2 \in T, g \in F^k \tag{5.7}$$

$$X_{f,0}^{k,t_1} \geq \sum_{g \in F^k} \sum_{\substack{t_2 \in T, \\ t_2 > t_1}} \gamma_{f,g}^{k,t_1,t_2},$$

$$\forall k \in M, t_1 \in T, f \in F^k \tag{5.8}$$

$$\sum_{f \in \hat{F}^k \setminus \{g\}} X_{f,g}^{k,t} = R_g^{k,t}, \qquad \forall k \in M, t \in T, g \in F^k \tag{5.9}$$

$$\sum_{g \in \hat{F}^k \setminus \{f\}} X_{f,g}^{k,t} = R_f^{k,t}, \qquad \forall k \in M, t \in T, f \in F^k \tag{5.10}$$

- Carryover Setup Constraint:

$$\sum_{f,g \in F^k} s_{f,g}^k \gamma_{f,g}^{k,0,1} = \beta^{k,1},$$

$$\forall k \in M \tag{5.11}$$

$$\sum_{f,g \in F^k} s_{f,g}^k \gamma_{f,g}^{k,t-1,t} = \alpha^{k,t-1} + \beta^{k,t},$$

$$\forall k \in M, t \in T; t > 1 \tag{5.12}$$

- Campaign Production Constraint:

$$\sum_{i \in N_f \cap N^k} W_{f,i}^{k,t} \leq l_f^{k,t} R_f^{k,t},$$

$$\forall k \in M, t \in T, f \in F^k \tag{5.13}$$

$$\sum_{i \in N_f \cap N^k} W_{f,i}^{k,t} \geq R_f^{k,t},$$

$$\forall k \in M, t \in T, f \in F^k \tag{5.14}$$

$$\sum_{i \in N_f \cap N^k} W_{f,i}^{k,t} \leq m_f^{k,t} Z_f^{k,t},$$

$$\forall k \in M, t \in T, f \in F^k \tag{5.15}$$

$$R_f^{k,t} \geq Z_f^{k,t},$$

$$\forall k \in M, t \in T, f \in F^k \tag{5.16}$$

- Setup Prohibition Constraint:

$$X_{f,g}^{k,t} = 0, \quad \forall k \in M, t \in T, f \in F^k, g \in F^k \cap A_f \tag{5.17}$$

$$\gamma_{f,g}^{k,t_1,t_2} = 0, \quad \forall k \in M, t_1 < t_2 \in \hat{T}, f \in F^k, g \in F^k \cap A_f \tag{5.18}$$

- Graph Connectivity Constraint:

$$\lambda_{f,g}^{k,t,u} \leq X_{f,g}^{k,t}, \quad \forall k \in M, t \in T, u, g \in F^k, f \in \hat{F}^k; f \neq g \tag{5.19}$$

$$\sum_{g \in F^k} \lambda_{0,g}^{k,t,u} = Z_u^{k,t}, \quad \forall k \in M, t \in T, u \in F^k \tag{5.20}$$

$$\sum_{f \in \hat{F}^k \setminus \{u\}} \lambda_{f,u}^{k,t,u} = Z_u^{k,t}, \quad \forall k \in M, t \in T, u \in F^k \tag{5.21}$$

$$\sum_{g \in F^k \setminus \{f\}} \lambda_{f,g}^{k,t,u} = \sum_{g \in \hat{F}^k \setminus \{f\}} \lambda_{g,f}^{k,t,u}, \quad \forall k \in M, t \in T, u, f \in F^k;\ f \neq u \tag{5.22}$$

- Ranges of Variables:

$$I_{f,i}^t \geq 0, G_{f,i}^t \geq 0, \quad \forall t \in T, f \in F, i \in N_f$$

$$P_{f,i}^{k,t} \geq 0, W_{f,i}^{k,t} \in \mathbb{Z}^+, \quad \forall t \in T, f \in F, i \in N_f, k \in M^i$$

$$R_f^{k,t} \in \mathbb{Z}^+, Z_f^{k,t} \in \{0,1\}, \quad \forall k \in M, t \in T, f \in F^k$$

$$X_{f,g}^{k,t} \in \mathbb{Z}^+, \quad \forall k \in M, t \in T, f, g \in \hat{F}^k; f \neq g$$

$$\gamma_{f,g}^{k,t_1,t_2} \in \{0,1\}, \quad \forall k \in M, t_1, t_2 \in \hat{T}, f, g \in F^k; t_1 < t_2$$

$$\alpha^{k,t} \geq 0, \beta^{k,t} \geq 0 \quad \forall k \in M, t \in T$$

$$\lambda_{f,g}^{k,t,u} \geq 0, \quad \forall k \in M, t \in T, u, g \in F^k, f \in \hat{F}^k; f \neq g.$$

5.3.1 Explanation on the Formulation

- Objective Function

The objective function is comprised of the total inventory hosting cost and the total setup cost. The latter is proportional to the total setup time, which includes both with-in period setups and carryover setups. Note that the production cost is assumed to be period-invariant and bay-invariant, and is therefore not included in the objective function.

- Flow Balance Constraint

Constraint (5.1) enforces the balance of inventory flow. Note that the amount of inventory is reduced in two cases: either it is used to satisfy the external demands, or it is consumed by the production of other products.

- Production Allocation/Batch Constraint

Constraint (5.2) allocates the production of product i to compatible bays, while Constraint (5.3) ensures that the allocated amount is produced using a sufficient number of batches. By substituting out the $P_{f,i}^{k,t}$-variables, these two constraints can be combined as follows:

$$G_{f,i}^t \leq \sum_{k \in M^i} q_{f,i}^k W_{f,i}^{k,t}, \quad \forall t \in T, f \in F, i \in N_f. \tag{5.23}$$

In this case, the exact amount of production in each bay needs to be determined from the value of $G_{f,i}^{t}$ after a solution is obtained. For example, we can assume $P_{f,i}^{k,t}$ to be proportional to the available capacity provided by the batches in different bays (i.e., $q_{f,i}^{k} W_{f,i}^{k,t}$). However, we retain the $P_{f,i}^{k,t}$-variables in the formulation for the convenience of deriving certain classes of valid inequalities later in Section 0.

- Sequencing Constraint

Constraints (5.5) and (5.6) dictate that the first changeover setup in bay k, if it exists at all, always starts with family f_0^k from period "0". During each period, the production in a bay is initiated by a carryover setup, which starts from an earlier period and ends in the current period. The product family that immediately follows an incoming carryover setup is recognized by a within-period changeover from the dummy family "0", as dictated by Constraint (5.7). The last product family produced during a period is recognized by a within-period changeover to the dummy family "0". This either indicates the end of production in this bay, or it starts a carryover setup pointing at a later period, as dictated by Constraint (5.8). For each product family, Constraints (5.9) and (5.10) ensure that the number of campaigns for a product family is equal to the number of in-coming and out-going setups, respectively.

- Carryover Setup Constraint

For a long carryover setup that straddles more than two periods, there must be at least one idle period between the two related campaigns. We assume that any setup can be accommodated entirely during an idle period, and hence, no restriction is put on the corresponding α- and β-variables in the above case. For the remaining case that a carryover setup straddles two consecutive periods, Constraints (5.11) and (5.12) allocate time to complete the setup at the end of the previous period and the beginning of the latter period.

- Campaign Production Constraint

Constraints (5.13)-(5.16) determine the numbers of campaigns based on the numbers of production batches. Constraints (5.13) and (5.14) are based on the fact that there is at least

one batch and at most $l_f^{k,t}$ batches in each campaign. The actual assignments of batches to campaigns are not indicated by the formulation, but they can be readily determined after a solution is obtained. Constraint (5.15) ensures that the production indicator is turned on if a batch of the corresponding family is produced. The right-hand side coefficient is an upper bound on the number of batches of family f that can be produced in bay k during period t. Constraint (5.16) ensures that the production indicator is turned off if no campaign is scheduled. It connects the number of campaigns (R-variables) with the production indicator (Z-variables), and also indirectly connects the Z-variables with the W-variables through Constraint (5.14) . Note that Constraint (5.15) can be replaced by the following alternative (but weaker, noting (5.14)) inequality:

$$R_f^{k,t} \leq m_f^{k,t} Z_f^{k,t}. \tag{5.24}$$

Moreover, the decoupling of the W- and Z-variables makes it easier to implement a decomposition scheme where these two variables reside in different parts of the decomposition. Also note that whenever $m_f^{k,t} \leq l_f^{k,t}$, Constraint (5.13) can be omitted, as it can be deduced from Constraints (5.15) and (5.16).

- Graph Connectivity Constraint

Constraints (5.19)-(5.22) pertain to the sequence of campaigns within a period. Since multiple production campaigns of a family may exist in the same bay during the same period, the sequencing of campaigns in bay k during period t is a High-Multiplicity Asymmetric Traveling Salesman Problem (HMATSP) (see Grigoriev and van de Klundert (2006)). Note that for all $k \in M, t \in T$, the set of active product families, $V^{k,t} \equiv \{0\} \cup \{g \in F^k : Z_g^{k,t} = 1\}$, is regarded as the set of cities in the traveling salesman problem, and the route starts from and ends at the dummy family "0". Constraints (5.19)-(5.22) are adapted from the polynomial-length HMATSP formulation of Sarin, Sherali, and Yao (2010), which is similar to the flow-based ATSP formulation proposed by Wong (1980) and further investigated by Sherali, Sarin, and Tsai (2006). Constraints (5.20) and (5.21), respectively, initiate and end the commodity flow, if and only if family u is in production (i.e., active).

Due to Constraint (5.19), the flow is allowed to follow along an edge only when the corresponding changeover setup exists. Constraint (5.22) enforces flow balance for each family.

5.3.2 Modeling Considerations

- Batch Production

Instead of assuming full-size production for all batches, we allow them to be produced in a smaller size than the maximum possible quantity. The full-size production strategy may sound reasonable; however, it leads to a "bullwhip" effect, which unnecessarily inflates the production levels. Consider the example shown in Figure 5.4, with a demand of 14 units for product P1.

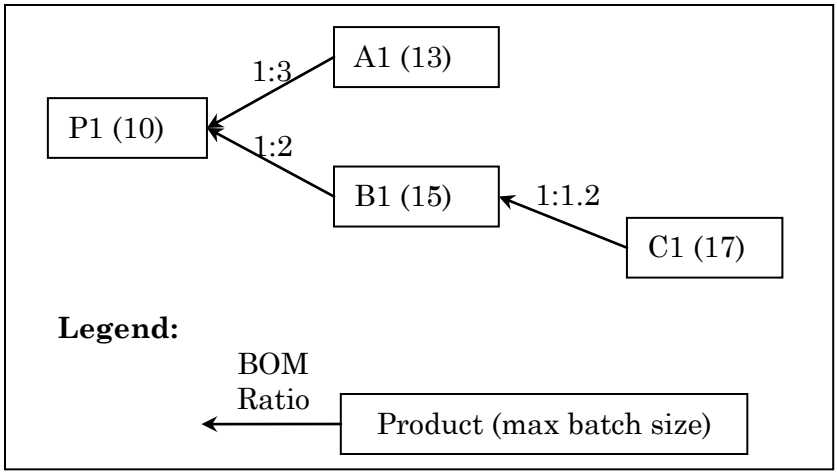


Figure 5.4: An example of a BOM tree with batch sizes

If we use full batches, the minimum production amounts are:

P1: $10 \times \left\lceil \frac{14}{10} \right\rceil = 10 \times 2 = 20$ (2 batches);

A1: $13 \times \left\lceil \frac{3 \times 20}{13} \right\rceil = 13 \times 5 = 65$ (5 batches);

B1: $15 \times \left\lceil \frac{2 \times 20}{15} \right\rceil = 15 \times 3 = 45$ (3 batches);

C1: $17 \times \left\lceil \frac{1.2 \times 45}{17} \right\rceil = 17 \times 4 = 68$ (4 batches).

If we use less than full batches, the minimum production amounts are:

P1: $14 \left(\Rightarrow \left\lceil \frac{14}{10} \right\rceil = 2 \text{ batches} \right)$;

A1: $3 \times 14 = 42 \left(\Rightarrow \left\lceil \frac{42}{13} \right\rceil = 4 \text{ batches} \right)$;

B1: $2 \times 14 = 28 \left(\Rightarrow \left\lceil \frac{28}{15} \right\rceil = 2 \text{ batches} \right)$;

C1: $1.2 \times 28 = 33.6 \left(\Rightarrow \left\lceil \frac{33.6}{17} \right\rceil = 2 \text{ batches} \right)$.

Note the reduction in the amount of production when less than full batch sizes are used.

- End Inventory

Our formulation does not require a minimum amount of inventory to be maintained at the end of the planning horizon. However, this requirement can be easily accommodated by regarding the required final inventory as additional demand in the last period. Although this transformation of data discounts the holding cost of required final inventory, it does not affect the validity of the formulation, as the required (minimum) inventory levels are fixed constants.

5.4 Alternative Formulations

In this section, we develop alternative formulations for various subproblem structures in the PPSMP. The goal is to achieve the best trade-off between model complexity and its tightness.

5.4.1 Plant Location Model of Production

If we regard the demands in different periods as pertaining to customers and the processing bays as plants, the lot-sizing subproblem in the PPMSP can be regarded as a *Plant Location Problem* (PLP), where the customer demands may be (partially) satisfied by a plant if the plant is chosen to be built (i.e., the corresponding production indicator is on). To explicitly

denote the flows from plants to customers, we disaggregate the production amount $G_{f,i}^t$ into multiple parts so that each part corresponds to the demand of a particular period, or contributes to the final inventory at the end of time horizon. This model is adapted from a similar model of Stadtler (2003). However, the formulation of Stadtler (2003) disallowed over-production, which may exist in our setting (see Section 5.5.2).

The concept of demand in the PLP is different from the (external) demands in the PPMSP. Note that, although there is no external demand for an intermediate product, a positive demand for this product may exist in the PLP, as driven by external demands for its downstream products. On the other hand, an external demand in the PPMSP does not necessarily translate into a demand in the PLP, as initial inventories may satisfy the external demand without any production. Hence, we define the *net demands* for the PLP as the least and the latest amounts of productions that are necessary to satisfy the external demands. Their values can be determined from the following linear programming formulation:

Variables:

$e_{f,i}^t$: Net demand for product i of family f during period t, $\forall t \in T, f \in F, i \in N_f$.

$D_{f,i}^t$: Cumulative net demand for product i of family f from period 1 to period t, $\forall t \in T, f \in F, i \in N_f$

Problem Net-Demand:

Minimize $$\sum_{t \in T} \sum_{f \in F} \sum_{i \in N_f} D_{f,i}^t$$

subject to:

$$I_{f,i}^0 + D_{f,i}^t \geq \sum_{\tau=1}^{t} d_{f,i}^\tau + \sum_{j \in N_f} b_{f,i,j} D_{f,j}^t,$$

$$\forall t \in T, f \in F, i \in N_f$$

$$\sum_{\tau=1}^{t} e_{f,i}^{\tau} = D_{f,i}^{t},$$

$$\forall t \in T, f \in F, i \in N_f$$

$$e_{f,i}^{t} \geq 0, D_{f,i}^{t} \geq 0$$

$$\forall t \in T, f \in F, i \in N_f.$$

Alternatively, we can calculate the net demand, $e_{f,i}^{t}$, using the following algorithm:

Net-Demand Algorithm:

Step 1. For each product i, set the initial value of its gross demand to $d_{f,i}^{t}$.

Step 2. Find a product i whose direct downstream products (which use product i as a raw material) have all been checked. If no such product exists, stop; otherwise, go to Step 3.

Step 3. Use the initial inventory of product i to satisfy its gross demands starting from the first period. Record the resulting inventory level $\hat{I}_{f,i}^{t}$ as its baseline inventory. Record the backlog value of product i in each period as its net demand value $e_{f,i}^{t}$.

Step 4. Multiply the net demands of product i by its BOM ratios to determine the required amounts of raw materials. Add the required amounts to the gross demands of its direct upstream products. Go to Step 2.

Note that the baseline inventory level also determines the baseline inventory cost:

$$C_H \equiv \sum_{t \in T} \sum_{f \in F} \sum_{i \in N_f} h_{f,i} \hat{I}_{f,i}^{t} .$$

In the following discussion, we regard $e_{f,i}^{t}$ and C_H to be fixed parameters as determined by the above method. In addition, we define the following notation:

Parameters:

$r_{f,i}$: Marginal holding cost of product i of family f, $\forall f \in F, i \in N_f$. This is the change in inventory cost, caused by the production of a unit amount of product i, as given by $r_{f,i} \equiv h_{f,i} - \sum_{j \in N_f} b_{f,j,i} h_{f,j}$.

Variables:

$H_{f,i}^{t_1,t_2}$: The amount of product i that is produced during period t_1, and utilized in period t_2 to satisfy some net demand, $\forall k \in M, t_1, t_2 \in T, f \in F^k; t_1 \leq t_2$.

$O_{f,i}^{t}$: The amount of product i that is produced during period t, but not used to satisfy any net demand, $\forall k \in M, t \in T, f \in F^k$. This amount of over-production will eventually go into final inventory.

Using the Plant Location Model of production, the PPMSP can be reformulated as follows:

Problem PPMSP-PLP:

$$\text{Minimize } C_H + \sum_{t_1<t_2\in T}\sum_{f\in F}\sum_{i\in N_f} r_{f,i}(t_2-t_1)H_{f,i}^{t_1,t_2} + \sum_{t\in T}\sum_{f\in F}\sum_{i\in N_f} r_{f,i}(|T|+1-t)O_{f,i}^{t}$$
$$+ \sigma \sum_{k\in M}\sum_{f\neq g\in F^k} s_{f,g}^{k}\left(\sum_{t\in T} X_{f,g}^{k,t} + \sum_{t_1\leq t_2\in \hat{T}} \gamma_{f,g}^{k,t_1,t_2}\right).$$

subject to: Constraints (5.3)-(5.22)

$$\sum_{t_1=1}^{t} H_{f,i}^{t_1,t} = e_{f,i}^{t},$$
$$\forall t \in T, f \in F, i \in N_f. \tag{5.25}$$

$$I_{f,i}^{t-1} + \sum_{t_2=t}^{|T|} H_{f,i}^{t,t_2} + O_{f,i}^{t} = I_{f,i}^{t} + d_{f,i}^{t} + \sum_{j\in N_f} b_{f,i,j}\left(\sum_{t_2=t}^{|T|} H_{f,j}^{t,t_2} + O_{f,j}^{t}\right),$$
$$\forall t \in T, f \in F, i \in N_f \tag{5.26}$$

$$\sum_{t_2=t}^{|T|} H_{f,i}^{t,t_2} + O_{f,i}^{t} = \sum_{k\in M^i} P_{f,i}^{k,t},$$
$$\forall t \in T, f \in F, i \in N_f. \tag{5.27}$$

Note that Constraints (5.26) and (5.27) are obtained by applying the following substitution to the Flow Balance Constraint (5.1) and the Production Allocation Constraint (5.2), respectively:

$$G_{f,i}^{t} = \sum_{t_2=t}^{|T|} H_{f,j}^{t,t_2} + O_{f,j}^{t},$$

$$\forall k \in M, t \in T, f \in F^k. \tag{5.28}$$

In case that P-variables are not used to construct additional valid inequalities, we can replace Constraints (5.3) and (5.27) with the following constraint, which is similar to Constraint (5.23):

$$\sum_{t_2=t}^{|T|} H_{f,i}^{t,t_2} + O_{f,i}^{t} \leq \sum_{k \in M^i} q_{f,i}^{k} W_{f,i}^{k,t},$$

$$\forall k \in M, t \in T, f \in F^k. \tag{5.29}$$

5.4.2 Alternative Formulations of the Graph Connectivity Constraint

Instead of using the multi-commodity flow formulation (i.e., Constraints (5.19)-(5.22)) to ensure connectivity of the setup sequence graph, we can use the following two alternative formulations.

Miller-Tucker-Zemlin-Type Formulation

Unlike the classic traveling salesman problem, the High-Multiplicity Asymmetric Traveling Salesman Problem (HMATSP) allows sub-tours within a feasible sequence. Therefore, the original formulation of MTZ SECs cannot be applied directly. To establish a consistent rank-order, we rely on the edges of first-visits, which mark the first visit of the route to each city (product family in terms of the PPMSP). We define the following additional variables:

$U_f^{k,t}$ The rank order of family f in bay k during period t, where

$U_f^{k,t} \in \mathbb{Z}^+ \cap \left[1, \left|F^k\right|\right], \forall k \in M, t \in T, f \in F^k.$

$Y_{f,g}^{k,t}$ Binary indicator to denote whether the first campaign of family g in bay k during period t is immediately preceded by a campaign of family f in the same period, $\forall k \in M, t \in T, f \in \hat{F}^k, g \in F^k; f \neq g$. Note that if $Y_{0,g}^{k,t} = 1$, the first campaign produces family g.

Proposition 5.1: *Along with the sequencing constraints* (5.5)-(5.10)*, the following set of valid inequalities induces a feasible setup sequence:*

$$\sum_{f \in \hat{F}^k \setminus \{g\}} Y_{f,g}^{k,t} = Z_g^{k,t}, \quad \forall k \in M, t \in T, g \in F^k \tag{5.30}$$

$$Y_{f,g}^{k,t} \leq X_{f,g}^{k,t}, \quad \forall k \in M, t \in T, f \in \hat{F}^k, g \in F^k; f \neq g \tag{5.31}$$

$$U_f^{k,t} - U_g^{k,t} + \left|F^k\right| \cdot Y_{f,g}^{k,t} \leq \left|F^k\right| - 1, \quad \forall k \in M, t \in T, f, g \in F^k; f \neq g. \tag{5.32}$$

Proof: Define the following sets, $\forall k \in M, t \in T$:

Set of active families, $V^{k,t} \equiv \{0\} \cup \{g \in F^k : Z_g^{k,t} = 1\}$,

Set of active edges, $E^{k,t} \equiv \left\{(f \to g) : f, g \in \hat{F}^k; f \neq g, X_{f,g}^{k,t} > 0\right\}$,

Set of first-visit edges, $\hat{E}^{k,t} \equiv \left\{(f \to g) : f \in \hat{F}^k, g \in F^k; f \neq g, Y_{f,g}^{k,t} = 1\right\}$.

For any feasible solution, the edge set $E^{k,t}$ induces a connected graph over $V^{k,t}$ by (5.5)-(5.10). A corresponding campaign sequence can be determined by using the procedure *ConvertToSequence* of Grigoriev and van de Klundert (2006). Hence, we can determine the values of the $Y_{f,g}^{k,t}$-variables according to the sequence obtained. If family g is in production, its first campaign in the sequence must be preceded by the campaign of another real product family, or by the dummy family "0". Hence, Constraint (5.30) is satisfied. Constraint (5.31) is also valid, as $Y_{f,g}^{k,t} = 1$ indicates a changeover setup from family f to family g, which is accounted for in the setup variable $X_{f,g}^{k,t}$. Note that the edge set of first-

visits, $\hat{E}^{k,t}$, induces a directed tree graph over the set of active families, as any active family can trace back to the dummy family by reversely following the edges in $\hat{E}^{k,t}$. Consequently, a full order of active families can be constructed by a preorder traversal of this tree graph. We use this rank order to determine the U-values that satisfy Constraint (5.32) (see Figure 5.5).

Conversely, if a feasible solution satisfies the constraints of Proposition 5.1, the edge set $E^{k,t}$ induces a connected graph over $V^{k,t}$. To see this, suppose that on the contrary, the graph induced by $E^{k,t}$ contains a proper subgraph Σ that is disconnected from the rest of the families in $V^{k,t}$. Let $V(\Sigma)$ be the set of families (vertices) in the subgraph Σ. For each family $g \in V(\Sigma)$, there exists a family f such that $Y_{f,g}^{k,t} = 1$, due to Constraint (5.30). Hence, there exist $|V(\Sigma)|$ edges that belong to the set $\hat{E}^{k,t}$ and that are incident at families in $V(\Sigma)$. According to Constraints (5.31), these $|V(\Sigma)|$ edges also belong to the set $E^{k,t}$. Therefore, Σ contains a cycle. This violates Constraint (5.32), since a sequential rank-order cannot be established around a cycle.■

Single Commodity Flow Formulation

In the multiple commodity flow formulation, a separate commodity is sent from the dummy family to any of the active families to ensure graph connectivity. In this section, we consider an alternative formulation in which a single commodity reaches all active families. It is based on the single commodity flow formulation of Gavish and Graves (1978) for the Asymmetric Traveling Salesman Problem.

Variables:

$\lambda_{f,g}^{k,t}$: Amount of commodity flow on edge (f, g) during time period t in bay k, $\forall k \in M, t \in T, f \neq g \in \hat{F}^k$.

Constraints:

$$\lambda_{f,g}^{k,t} \leq \left|F^k\right| \cdot X_{f,g}^{k,t}, \qquad \forall k \in M, t \in T, f, g \in \hat{F}^k; f \neq g \tag{5.33}$$

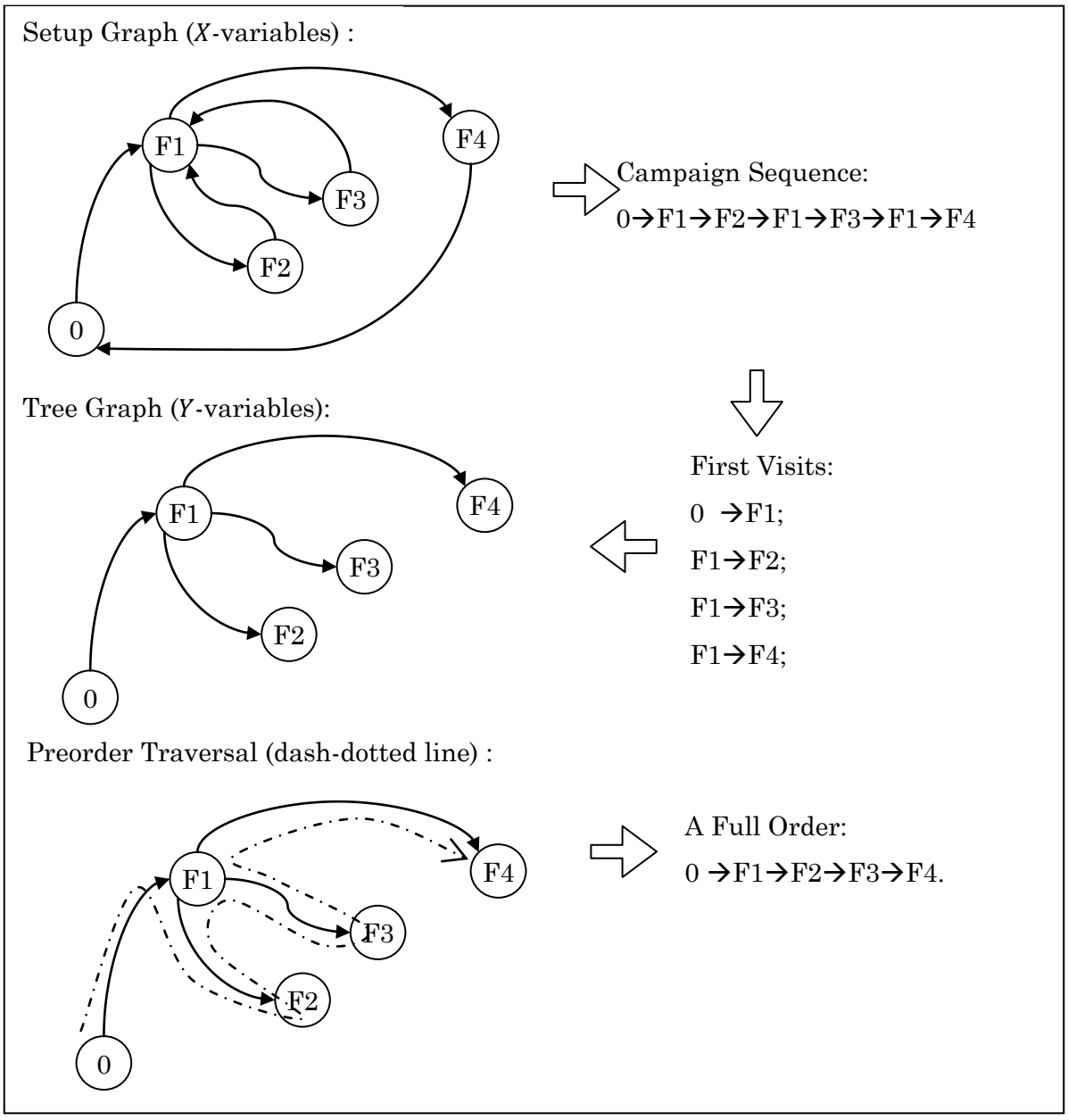


Figure 5.5: Construction of a full order based on the setup variables

$$\sum_{f\in\hat{F}^k\setminus\{g\}} \lambda_{f,g}^{k,t} = \sum_{f\in\hat{F}^k\setminus\{g\}} \lambda_{g,f}^{k,t} + Z_g^{k,t},$$

$$\forall k \in M, t \in T, g \in F^k. \qquad (5.34)$$

The flow has an initial amount of $|F^k|$ originating from the dummy family "0". Each active family consumes a unit amount when it is visited by the flow. In case that some family is not produced (i.e., not active), the number of active families will be less than $|F^k|$. The extra amount of flow will be sent back to the dummy family.

5.5 Valid Inequalities

In this subsection, we derive various classes of valid inequalities to further tighten the model representations.

5.5.1 Inequalities Based on the Time Capacity Constraint

Proposition 5.2: *If there exists a product* $i^* \in N_f \cap N^k$ *such that* $p_{f,i^*}^k > \frac{c^{k,t}}{m_f^{k,t}}$, *then inequality* (5.15) *can be strengthened by also incorporating the following valid inequality:*

$$\sum_{i\in N_f\cap N^k} p_{f,i}^k W_{f,i}^{k,t} \le c^{k,t} Z_f^{k,t}, \qquad \forall k \in M, t \in T, f \in F^k. \qquad (5.35)$$

Proof: First, note that the validity of (5.35) follows from the Time Capacity Constraint (5.4). On the other hand, a solution of $W_{f,i^*}^{k,t} = m_f^{k,t}$, $W_{f,i}^{k,t} = 0, \forall i \neq i^*$, and $Z_f^{k,t} = 1$ is clearly feasible for (5.15). But it is infeasible for (5.35) as $c^{k,t} < p_{f,i^*}^k m_f^{k,t} = p_{f,i^*}^k W_{f,i^*}^{k,t}$. ■

As an illustration, consider the following example with two products (Products 1 and 2) of the same family f that share the same bay k. Suppose $p_{f,1}^k = 3$, $p_{f,2}^k = 5$, $Z_f^{k,t} = 1$, and $c^{k,t} = 11$. The inequalities take the following forms:

Inequality (5.15) $\qquad W_{f,1}^{k,t} + W_{f,2}^{k,t} \le \left\lfloor \frac{11}{3} \right\rfloor = 3;$

Inequality (5.35) $3W_{f,1}^{k,t} + 5W_{f,2}^{k,t} \leq 11.$

As shown in Figure 5.6, note that, neither of the two inequalities (5.15) and (5.35) dominates the other. Hence, they can be simultaneously added to the model.

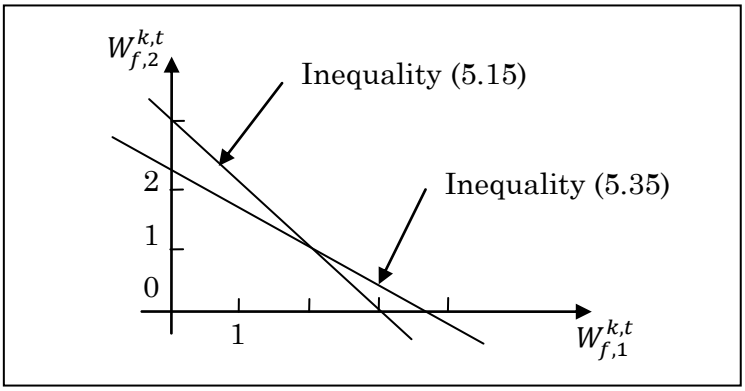


Figure 5.6: Examples of valid inequalities

Furthermore, we can strengthen inequality (5.35) itself by deriving a smaller right-hand side coefficient from the following subset-sum problem, $\forall k \in M, t \in T, f \in F^k$:

$$\hat{c}_f^{k,t} = \text{Maximize} \quad \sum_{i \in N_f \cap N^k} p_{f,i}^k W_{f,i}^{k,t}$$

$$\text{subject to:} \quad \sum_{i \in N_f \cap N^k} p_{f,i}^k W_{f,i}^{k,t} \leq c^{k,t}$$

$$W_{f,i}^{k,t} \in \mathbb{Z}^+.$$

The optimal objective $\hat{c}_f^{k,t}$ provides an upper bound on the production time consumed by batches of family f in bay k during period t. Consequently, inequality (5.35) can be modified as:

$$\sum_{i \in N_f \cap N^k} p_{f,i}^k W_{f,i}^{k,t} \leq \hat{c}_f^{k,t} Z_f^{k,t},$$

$$\forall k \in M, t \in T, f \in F^k. \qquad (5.36)$$

By considering the production of each product separately, we can further add the following Chvatal-type (Nemhauser and Wolsey, 1999) valid inequality:

$$W_{f,i}^{k,t} \leq \left\lfloor \frac{c^{k,t}}{p_{f,i}^{k}} \right\rfloor Z_f^{k,t},$$

$$\forall k \in M, t \in T, f \in F^k. \tag{5.37}$$

5.5.2 Inequalities Based on Upper Bounds for Production Amounts

In the basic model, an increment in the production level $\left(P_{f,i}^{k,t}\right)$ tends to drive up the number of batches $\left(W_{f,i}^{k,t}\right)$, which, in turn, make the production indicator $\left(Z_f^{k,t}\right)$ positive. We consider the following valid inequality, which directly connects the production level with the production indicator:

$$P_{f,i}^{k,t} \leq \theta_{f,i}^{k,t} Z_f^{k,t},$$

$$\forall t \in T, k \in M, f \in F^k, i \in N_f \cap N^k, \tag{5.38}$$

where the parameter $\theta_{f,i}^{k,t}$ is an upper bound on the amount of product i of family f that is produced in bay k during period t. By Constraints (5.3) and (5.37), we can let

$$\theta_{f,i}^{k,t} = q_{f,i}^{k} \left\lfloor \frac{c^{k,t}}{p_{f,i}^{k}} \right\rfloor.$$

However, the value of $\theta_{f,i}^{k,t}$ can be further reduced as follows:

The production in the PPMSP is driven by two facts: the need to satisfy external demand and the desire to reduce inventory. *Production-Driven-by-Demand* (PDD) is motivated by the former while *Production-Driven-by-Inventory* (PDI) is motivated by a negative value of the marginal holding cost. Although these two types of production may overlap with each other, the summation of their upper bound values provides a valid upper bound on the amount of production in an optimal solution. To derive this upper bound value, we consider two complementary relaxations of the original problem.

In the relaxation for determining PDD, we do not assume existence of initial inventory, capacity constraints, or setup costs. Note that PDI is automatically excluded in this

relaxation, as there does not exist initial inventory to justify over-production. Moreover, since there are no constraints on capacity, the production will be carried out as late as possible in order to drive the inventory cost to zero, and the corresponding amounts of production can be computed based on the demand values and the BOM ratios. We denote these production amounts by $\hat{G}_{f,i}^t$ as defined below:

$\hat{G}_{f,i}^t$: The amount of product i of family f that is produced during period t, $\forall t \in T, f \in F, i \in N_f$.

We determine the values of $\hat{G}_{f,i}^t$ by solving the following linear program:

$$\text{Minimize} \quad \sum_{t \in T} \sum_{f \in F} \sum_{i \in N_f} \hat{G}_{f,i}^t$$

$$\text{subject to:} \quad \hat{G}_{f,i}^t = d_{f,i}^t + \sum_{j \in N_f} b_{f,i,j} \hat{G}_{f,i}^t, \qquad \forall t \in T, f \in F, i \in N_f$$

$$\hat{G}_{f,i}^t \geq 0, \qquad \forall t \in T, f \in F, i \in N_f.$$

In the original PPMSP, PDD may need to be scheduled in an earlier period due to capacity constraints. Therefore, $\sum_{\tau=t}^{|T|} \hat{G}_{f,i}^{\tau}$ gives an upper bound on the amount of PDD in period t, $\forall t \in T$.

In the relaxation for determining PDI, we assume that there is no external demand, capacity constraints, or setup cost. In the original PPMSP, some initial inventory may be consumed by PDD, thereby making PDI less likely to occur. By assuming no external demand, we exclude PDD from this relaxation. However, we cannot simply minimize the total inventory cost for the current relaxation and claim the corresponding production amount to be an upper bound on PDI. This is due to the fact that multiple products may compete for the same raw material inventory, and the actual outcome depends on the capacity constraints that have been relaxed in the current relaxation. To see this, consider the example in Figure 5.7. If both product A and product B have some initial inventory, the production of either Product C or Product D is beneficial in reducing inventory holding costs. If we simply minimize the total inventory cost for the current relaxation, only

Product C will have a positive production amount. In the original problem, however, PDI of Product D may have a positive value as a result of the limited production of Product C due to other factors (such as capacity constraints). Therefore, we determine the maximum amount of PDI for each product individually, and use it as an upper bound for the original PPMSP.

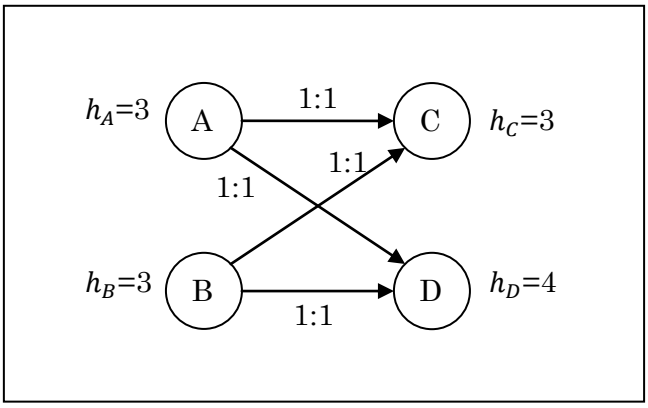


Figure 5.7: An example of BOM relations and product holding costs

Note that, due to the absence of capacity constraints, all production will occur in the first period. Hence, we consider the current relaxation for one period only and denote the upper bound value by:

$\tilde{G}_{f,i}$: Upper bound on the PDI for product i of family f, $\forall f \in F, i \in N_f$.

We define an *attractive product* as a product having a negative marginal holding cost. We claim that PDI always ends up in the inventory of an attractive product. This follows by the fact that an increase in the end inventory of a non-attractive product always increases the inventory holding cost. Note that PDI does not necessarily involve the production of only attractive products, as the production of a non-attractive product may be necessary to supply raw materials for an attractive product occurring downstream. The production of such a non-attractive product is also regarded as PDI. We use a two-step procedure to determine an upper bound on PDI:

Step 1. Consider each attractive product individually. Let it be product i of family f. Determine its maximum profitable amount of production, $\pi_{f,i}$, by solving a localized problem, which only allows itself and its upstream products to be produced and minimizes the total inventory cost.

Step 2. Determine the required amount of each product i based on the production of its downstream attractive products. This required amount is then decreased by its initial inventory to determine the level of production, $\kappa_{f,i}$. If the current product is a non-attractive product, $\kappa_{f,i}$ gives an upper bound on its PDI value (i.e., $\tilde{G}_{f,i} = \kappa_{f,i}$); otherwise, the upper bound value is given by $\tilde{G}_{f,i} = \max\{\kappa_{f,i}, \pi_{f,i}\}$.

Note that, in the original PPMSP, PDI may be postponed due to capacity constraints and conflicts between products. The above discussion establishes the following result:

Proposition 5.3: *An upper bound on the amount of product i that can be produced in bay k during period t is given by:*

$$\theta_{f,i}^{k,t} = \min\left\{ q_{f,i}^{k} \left\lfloor \frac{c^{k,t}}{p_{f,i}^{k}} \right\rfloor, \sum_{\tau=t}^{|T|} \hat{G}_{f,i}^{\tau} + \tilde{G}_{f,i} \right\}. \blacksquare$$

Similarly to (5.38), we can introduce the following inequality to make a direct connection between the production level $G_{f,i}^{t}$ and the production indicator $Z_f^{k,t}$:

$$G_{f,i}^{t} \leq \sum_{k \in M^i} \theta_{f,i}^{k,t} Z_f^{k,t}, \qquad \forall t \in T, f \in F, i \in N_f. \tag{5.39}$$

Note that, (5.39) can be deduced from inequalities (5.2) and (5.38). Therefore, it is redundant when (5.38) is present. However, applying (5.39) instead of (5.38) may be beneficial as this permits to eliminate the P-variables from the model formulation. Also, note that for the Plant Location Model, inequality (5.39) is adapted as:

$$\sum_{t_2=t}^{|T|} H_{f,i}^{t,t_2} + O_{f,i}^{t} \leq \sum_{k \in M^i} \theta_{f,i}^{k,t} Z_f^{k,t}, \qquad \forall t \in T, f \in F, i \in N_f. \tag{5.40}$$

5.5.3 Inequalities Based on the Lot-Sizing Subproblem

Proposition 5.4: *If $\zeta_{f,i}^{t}$ represents the production indicator for product i of family f in period t, $\forall t \in T, f \in F, i \in N_f$; and $\eta_{f,i,j}$ is the amount of product i of family f needed (directly or indirectly) to produce a unit amount of product j of the same family (in particular, $\eta_{f,j,j} \triangleq 1$), $\forall f \in F$ and $i, j \in N_f$, then*

$$\sum_{j \in N_f} \eta_{f,i,j}\, I_{f,j}^{t-1} \geq \left(1 - \zeta_{f,i}^{t}\right) \sum_{j \in N_f} \eta_{f,i,j}\, d_{f,j}^{t}\,, \quad \forall t \in T, f \in F, i \in N_f. \tag{5.41}$$

Proof: This follows by the fact that if there is no production in the current period, then the demand has to be satisfied by the inventory carried over from the previous period. ■

Note that the value of $\eta_{f,i,j}$ is positive if and only if product i is an upstream product with respect to product j. The $\eta_{f,i,j}$-values can be determined by using the following algorithm:

Generalized BOM Ratio

For each $f \in F$:

For each $j \in N_f$:

Let $\eta_{f,i,j} = 0, \forall i \in N_f$;

Define a temporary variable $x_{f,i,j}, \forall i \in N_f$;

Let $x_{f,j,j} = 1$ and $x_{f,i,j} = 0, \forall i \in N_f; i \neq j$;

While there exists a product $i \in N_f$ such that $x_{f,i,j} > 0$:

Let $\eta_{f,i,j} \leftarrow \eta_{f,i,j} + x_{f,i,j}$;

For each $k \in N_f$ such that $b_{f,k,i} > 0$:

Let $x_{f,k,j} \leftarrow x_{f,k,j} + b_{f,k,i} x_{f,i,j}$;

End for

Let $x_{f,i,j} = 0$;

End while

End for

End for

Note that the following constraints establish the definitional roles of $\zeta_{f,i}^t$:

$$W_{f,i}^{k,t} \leq \left\lfloor \frac{c^{k,t}}{p_{f,i}^k} \right\rfloor \zeta_{f,i}^t, \quad \forall k \in M, t \in T, f \in F^k, i \in N_f \cap N^k \tag{5.42}$$

$$\sum_{k \in M^i} W_{f,i}^{k,t} \geq \zeta_{f,i}^t, \quad \forall t \in T, \forall f \in F, \forall i \in N_f. \tag{5.43}$$

The smaller the value of $\zeta_{f,i}^t$, the stronger is inequality (5.41). Hence, a lower bounding constraint (5.42) on $\zeta_{f,i}^t$ will not help to strengthen (5.41). On the other hand, inequality (5.37) also implies an upper bound value of $\left\lfloor \frac{c^{k,t}}{p_{f,i}^k} \right\rfloor$ for $W_{f,i}^{k,t}$. Hence, we can omit inequality (5.42) if inequality (5.37) is present.

Moreover, we can add the following valid inequality:

$$\sum_{k \in M^i} Z_f^{k,t} \geq \zeta_{f,i}^t, \qquad \forall t \in T, f \in F, i \in N_f. \tag{5.44}$$

Furthermore, if the Plant Location Model is used, we can incorporate the following valid inequality:

$$H_{f,i}^{t_1,t_2} \leq e_{f,i}^{t_2} \zeta_{f,i}^{t_1}, \quad \forall t_1 \leq t_2 \in T, f \in F, i \in N_f \tag{5.45}$$

Atamtürk and Muñoz (2004) have proposed bottleneck cover inequalities that are based on the same idea as that of Proposition 5.4. Their results have been extended by Glover and Sherali (2005) to consider the production status of multiple periods in general. For the PPMSP, we can consider the production indicators of the next two periods and derive the following valid inequality:

$$\sum_{j\in N_f} \eta_{f,i,j}\, I_{f,j}^{t-1} \geq \left(1-\zeta_{f,i}^{t}\right)\hat{d}_{f,i}^{t} + \left(1-\zeta_{f,i}^{t+1}\right)\min\{\hat{d}_{f,i}^{t+1}, \max\{\hat{d}_{f,i}^{t} + \hat{d}_{f,i}^{t+1} - \omega_{f,i}^{t}, 0\}\},$$
$$\forall t \in T, f \in F, i \in N_f; t < |T|, \qquad (5.46)$$

where $\hat{d}_{f,i}^{t}$ is the total converted demand that is defined as $\hat{d}_{f,i}^{t} \equiv \sum_{j\in N_f} \eta_{f,i,j}\, d_{f,j}^{t}$, and where $\omega_{f,i}^{t}$ is the maximum production amount of product i of family f during period t, which can be taken as $\omega_{f,i}^{t} \equiv \sum_{k\in M^i} \theta_{f,i}^{k,t}$, where, as defined for (5.38), $\theta_{f,i}^{k,t}$ is an upper bound on the amount of product i that is produced in bay k during period t.

Note that inequality (5.41) establishes a relation between the aggregated production indicator $\zeta_{f,i}^{t}$ and the inventory level. We can avoid the introduction of the new ζ-variables by considering the following disaggregated constraint, which is established by Proposition 5.5 below:

$$\sum_{j\in N_f} \eta_{f,i,j}\, I_{f,j}^{t-1} \geq \sum_{k\in M^i} \left(1 - Z_f^{k,t}\right)\epsilon_{f,i}^{k,t},$$
$$\forall t \in T, f \in F, i \in N_f, s.t. \sum_{k\in M^i} \epsilon_{f,i}^{k,t} \leq \hat{d}_{f,i}^{t}, \qquad (5.47)$$

where $\epsilon_{f,i}^{k,t}$ is defined as the shortage in capacity if bay k is not used to produce product i during period t, i.e., $\epsilon_{f,i}^{k,t} = \max\left\{0, \hat{d}_{f,i}^{t} - \sum_{\hat{k}\in M^i\setminus\{k\}} \theta_{f,i}^{\hat{k},t}\right\}$.

Proposition 5.5: *Inequality* (5.47) *is valid if* $\sum_{k\in M^i} \epsilon_{f,i}^{k,t} \leq \hat{d}_{f,i}^{t}$, $\forall t \in T, f \in F, i \in N_f$.

Proof: If $\left|M^i\right| = 1$, then $\epsilon_{f,i}^{k,t} = \hat{d}_{f,i}^{t}$ for the only bay k in M^i. Hence, inequality (5.47) reduces to inequality (5.41), which has been shown to be valid. Hence, suppose that $\left|M^i\right| > 1$. We first show that $\hat{d}_{f,i}^{t} \leq \sum_{k\in M^i} \theta_{f,i}^{k,t}$. On the contrary, suppose that $\hat{d}_{f,i}^{t} > \sum_{k\in M^i} \theta_{f,i}^{k,t}$. We then have $\epsilon_{f,i}^{k,t} = \hat{d}_{f,i}^{t} - \sum_{\hat{k}\in M^i\setminus\{k\}} \theta_{f,i}^{\hat{k},t}$, $\forall k \in M^i$. Hence,

$$\sum_{k\in M^i} \epsilon_{f,i}^{k,t} = \sum_{k\in M^i} \left(\hat{d}_{f,i}^{t} - \sum_{\hat{k}\in M^i\setminus\{k\}} \theta_{f,i}^{\hat{k},t} \right)$$

$$= \hat{d}_{f,i}^{t} + \left(\left|M^i\right| - 1\right) \cdot \left(\hat{d}_{f,i}^{t} - \sum_{k \in M^i} \theta_{f,i}^{k,t}\right) > \hat{d}_{f,i}^{t}.$$

This contradicts the assumption that $\sum_{k \in M^i} \epsilon_{f,i}^{k,t} \leq \hat{d}_{f,i}^{t}$. Therefore, $\hat{d}_{f,i}^{t} \leq \sum_{k \in M^i} \theta_{f,i}^{k,t}$.

We further define $\widetilde{M}^i = \left\{k \in M^i : \epsilon_{f,i}^{k,t} > 0, Z_f^{k,t} = 0\right\}$. If $\widetilde{M}^i = \phi$, inequality (5.47) is trivially valid. Hence, suppose that $\left|\widetilde{M}^i\right| \geq 1$. Note that family f is not in production on the bays of $\widetilde{M}^i$. To satisfy the converted demand of product i in period t, we must have

$$\sum_{j \in N_f} \eta_{f,i,j} I_{f,j}^{t-1} \geq \hat{d}_{f,i}^{t} - \sum_{k \in M^i \setminus \widetilde{M}^i} \theta_{f,i}^{k,t}.$$

But

$$\left(\hat{d}_{f,i}^{t} - \sum_{k \in M^i \setminus \widetilde{M}^i} \theta_{f,i}^{k,t}\right) - \sum_{k \in \widetilde{M}^i} \left(\hat{d}_{f,i}^{t} - \sum_{\hat{k} \in M^i \setminus \{k\}} \theta_{f,i}^{\hat{k},t}\right)$$

$$= \hat{d}_{f,i}^{t} - \sum_{k \in M^i \setminus \widetilde{M}^i} \theta_{f,i}^{k,t} - \left|\widetilde{M}^i\right| \cdot \left(\hat{d}_{f,i}^{t} - \sum_{k \in M^i} \theta_{f,i}^{k,t}\right) - \sum_{k \in \widetilde{M}^i} \theta_{f,i}^{k,t}$$

$$= \left(1 - \left|\widetilde{M}^i\right|\right) \cdot \left(\hat{d}_{f,i}^{t} - \sum_{k \in M^i} \theta_{f,i}^{k,t}\right) \geq 0.$$

Therefore,

$$\sum_{j \in N_f} \eta_{f,i,j} I_{f,j}^{t-1} \geq \hat{d}_{f,i}^{t} - \sum_{k \in M^i \setminus \widetilde{M}^i} \theta_{f,i}^{k,t}$$

$$\geq \sum_{k \in \widetilde{M}^i} \left(\hat{d}_{f,i}^{t} - \sum_{\hat{k} \in M^i \setminus \{k\}} \theta_{f,i}^{\hat{k},t}\right) = \sum_{k \in M^i} \left(1 - Z_f^{k,t}\right) \epsilon_{f,i}^{k,t}. \blacksquare$$

5.5.4 Inequalities Based on the Carryover Setup Structure

Consider the following aggregated carryover setup variables:

γ^{k,t_1,t_2}: Indicator for a carryover setup, which starts at the end of period t_1, and ends at the beginning of period t_2, $\forall k \in M, t_1, t_2 \in \hat{T}; t_1 < t_2$.

Note that the above variables do not distinguish among families involved in the setup changeover. Hence, their values can be determined from the aggregated production indicator defined as:

$Z^{k,t}$: Production indicator for bay k during period t, $\forall k \in M, t \in T$.

The values of the $Z^{k,t}$-variables can be fixed by modifying the Time Capacity Constraint (5.4) as follows:

$$\sum_{f \in F} \sum_{i \in N_f \cap N^k} p_{f,i}^k W_{f,i}^{k,t} + \sum_{f \neq g \in F^k} s_{f,g}^k X_{f,g}^{k,t} + \alpha^{k,t} + \beta^{k,t} \leq c^{k,t} Z^{k,t}, \qquad (5.48)$$

$$\forall t \in T, k \in M.$$

Consequently, we have

$$\sum_{t_2 \in T} \gamma^{k,0,t_2} \leq 1, \qquad \forall k \in M \qquad (5.49)$$

$$Z^{k,t_2} = \sum_{\substack{t_1 \in \hat{T}, \\ t_1 < t_2}} \gamma^{k,t_1,t_2}, \qquad \forall k \in M, t_2 \in T \qquad (5.50)$$

$$Z^{k,t_1} \geq \sum_{\substack{t_2 \in T, \\ t_2 > t_1}} \gamma^{k,t_1,t_2}, \qquad \forall k \in M, t_1 \in T. \qquad (5.51)$$

Note that inequalities (5.49), (5.50) and (5.51) are similar to inequalities (5.5), (5.7) and (5.8) of the Sequencing Constraint.

Proposition 5.6: *The following inequalities are valid and are not necessarily implied by the original formulation:*

$$\sum_{f \in F^k} Z_f^{k,t} \geq Z^{k,t}, \qquad \forall t = T, k \in M \qquad (5.52)$$

$$\gamma^{k,t_1,t_2} = \sum_{f,g \in F^k} \gamma_{f,g}^{k,t_1,t_2}, \qquad \forall k \in M, t_1, t_2 \in \hat{T}; t_1 < t_2. \qquad (5.53)$$

Proof: The validity of inequality (5.52) is immediate due to the definitions of the variables $Z_f^{k,t}$ and $Z^{k,t}$. Since γ^{k,t_1,t_2} is defined as an indicator for aggregated carryover, inequality (5.53) also holds true. To establish the tightening of the PPMSP formulation with the addition of these inequalities, we construct a feasible LP solution to the original model that is infeasible to these added inequalities.

Suppose that the problem consists of only one bay (bay 1) and two periods (periods 1 and 2). Three products from three different families are considered for production. They are, respectively, product $\mathfrak{p}$ of family A, product $\mathfrak{q}$ of family B, and product $\mathfrak{r}$ of family C. Consider a feasible solution to the original problem, such that product $\mathfrak{p}$ is only produced in period 1, and products $\mathfrak{q}$ and $\mathfrak{r}$ are only produced in period 2. By inequality (5.48), we have

$$Z^{1,1} \geq \frac{p_{A,\mathfrak{p}}^1 W_{A,\mathfrak{p}}^{1,1}}{c^{1,1}};$$

$$Z^{1,2} \geq \frac{p_{B,\mathfrak{q}}^1 W_{B,\mathfrak{q}}^{1,2} + p_{C,\mathfrak{r}}^1 W_{C,\mathfrak{r}}^{1,2}}{c^{1,2}}.$$

Furthermore, according to inequalities (5.49) and (5.50), we have

$$\gamma^{1,0,1} + \gamma^{1,0,2} \leq 1;$$

$$Z^{1,1} = \gamma^{1,0,1};$$

$$Z^{1,2} = \gamma^{1,0,2} + \gamma^{1,1,2}.$$

Hence,

$$\gamma^{1,1,2} = Z^{1,2} - \gamma^{1,0,2} \geq Z^{1,2} + \gamma^{1,0,1} - 1 \geq Z^{1,2} + Z^{1,1} - 1$$

$$\geq \frac{p_{A,\mathfrak{p}}^1 W_{A,\mathfrak{p}}^{1,1}}{c^{1,1}} + \frac{p_{B,\mathfrak{q}}^1 W_{B,\mathfrak{q}}^{1,2} + p_{C,\mathfrak{r}}^1 W_{C,\mathfrak{r}}^{1,2}}{c^{1,2}} - 1.$$

On the other hand, inequality (5.53) requires that

$$\gamma^{1,1,2} = \gamma_{A,A}^{1,1,2} + \gamma_{A,B}^{1,1,2} + \gamma_{A,B}^{1,1,2} + \gamma_{B,A}^{1,1,2} + \gamma_{B,B}^{1,1,2} + \gamma_{B,C}^{1,1,2} + \gamma_{C,A}^{1,1,2} + \gamma_{C,B}^{1,1,2} + \gamma_{C,C}^{1,1,2}$$

$$= \gamma_{C,A}^{1,1,2} + \gamma_{C,B}^{1,1,2}.$$

Hence, the solution violates the added inequalities, if the following condition holds:

$$\frac{p_{A,\mathfrak{p}}^1 W_{A,\mathfrak{p}}^{1,1}}{c^{1,1}} + \frac{p_{B,\mathfrak{q}}^1 W_{B,\mathfrak{q}}^{1,2} + p_{C,\mathfrak{r}}^1 W_{C,\mathfrak{r}}^{1,2}}{c^{1,2}} - 1 > \gamma_{C,A}^{1,1,2} + \gamma_{C,B}^{1,1,2}.$$

For example, consider the following production pattern, which satisfies the original model

t	$W_{A,\mathfrak{p}}^{1,t}$	$W_{B,\mathfrak{q}}^{1,t}$	$W_{C,\mathfrak{r}}^{1,t}$	$c^{1,t}$
1	3	0	0	24
2	0	2	1	24

If the batch processing times are $p_{A,\mathfrak{p}}^1 = 7$, $p_{B,\mathfrak{q}}^1 = 6$, and $p_{C,\mathfrak{r}}^1 = 4$, we have the maximum possible numbers of batches as $m_A^{1,t} = \left\lfloor \frac{24}{7} \right\rfloor = 3, m_B^{1,t} = \left\lfloor \frac{24}{6} \right\rfloor = 4$, and $m_C^{1,t} = \left\lfloor \frac{24}{4} \right\rfloor = 6$. According to the campaign production constraints (5.13)-(5.16) and the sequencing constraints (5.5)-(5.10), the setup values in Figure 5.8 constitute a feasible LP solution.

However,

$$\begin{aligned}
&\frac{p_{A,\mathfrak{p}}^1 W_{A,\mathfrak{p}}^{1,1}}{c^{1,1}} + \frac{p_{B,\mathfrak{q}}^1 W_{B,\mathfrak{q}}^{1,2} + p_{C,\mathfrak{r}}^1 W_{C,\mathfrak{r}}^{1,2}}{c^{1,1}} - 1 \\
&= \frac{7 \times 3}{24} + \frac{6 \times 2 + 4 \times 1}{24} - 1 \\
&= \frac{7}{8} + \frac{2}{3} - 1 = \frac{13}{24} \\
&> \frac{1}{2} + 0 = \gamma_{C,A}^{1,1,2} + \gamma_{C,B}^{1,1,2}.
\end{aligned}$$

This implies a violation of inequality (5.53).

According to inequality (5.48), we also have

$$\begin{aligned}
Z^{1,2} &\geq \frac{p_{B,\mathfrak{q}}^1 W_{B,\mathfrak{q}}^{1,2} + p_{C,\mathfrak{r}}^1 W_{C,\mathfrak{r}}^{1,2} + X_A^{1,2} s_{A,B}^1}{c^{1,2}} \\
&\geq \frac{6 \times 2 + 4 \times 1 + \frac{1}{6} s_{A,B}^1}{24} \\
&> 0 + \frac{1}{2} + \frac{1}{6} \\
&= Z_A^{1,2} + Z_B^{1,2} + Z_C^{1,2}.
\end{aligned}$$

This is a violation of inequality (5.52).■

Note that to yield feasibility of the constructed solution to the added inequalities, the value of $\gamma_{C,A}^{1,1,2} + \gamma_{C,B}^{1,1,2}$ has to be at least $\frac{13}{24}$. In case that $s_{C,B}^1 > 4s_{A,B}^1$, the total setup time in period 2 will increase as shown by the following argument.

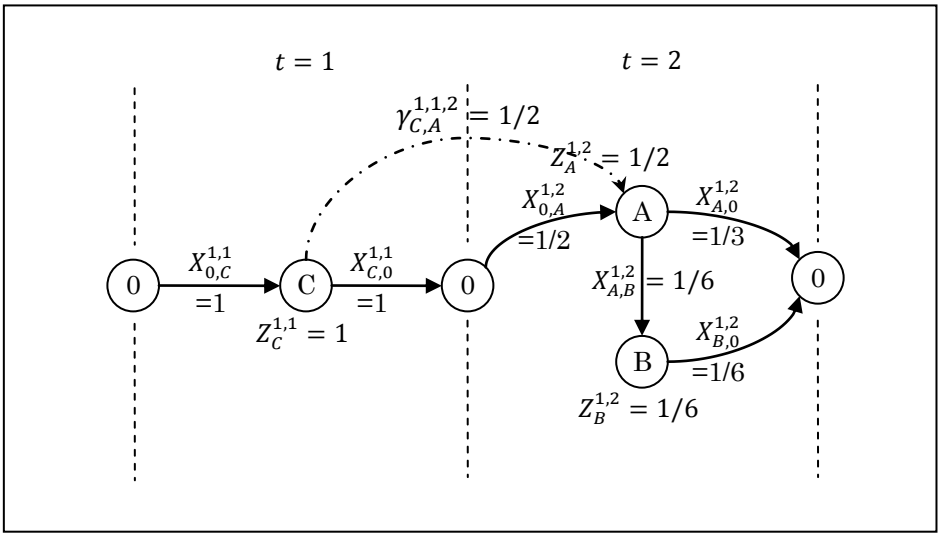


Figure 5.8: A feasible solution to the LP relaxation

Note that $X_{B,A}^{1,2} + \gamma_{C,A}^{1,1,2} = X_{B,A}^{1,2} + X_{0,A}^{1,2} \geq Z_A^{1,2} \geq \frac{1}{2}$. Hence, $X_{B,A}^{1,2} \geq \frac{1}{2} - \gamma_{C,A}^{1,1,2}$.

If $\gamma_{C,A}^{1,1,2} \leq \frac{1}{2}$, then

$$\begin{aligned}
&\gamma_{C,A}^{1,1,2} \cdot s_{C,A}^1 + X_{B,A}^{1,2} \cdot s_{B,A}^1 + \gamma_{C,B}^{1,1,2} \cdot s_{C,B}^1 \\
&\geq \gamma_{C,A}^{1,1,2} \cdot s_{C,A}^1 + \left(\frac{1}{2} - \gamma_{C,A}^{1,1,2}\right) \cdot s_{B,A}^1 + \left(\frac{13}{24} - \gamma_{C,A}^{1,1,2}\right) \cdot s_{C,B}^1 \\
&= \gamma_{C,A}^{1,1,2} \cdot s_{C,A}^1 + \left(\frac{1}{2} - \gamma_{C,A}^{1,1,2}\right) \cdot \left(s_{B,A}^1 + s_{C,B}^1\right) + \frac{1}{24} s_{C,B}^1 \\
&\geq \gamma_{C,A}^{1,1,2} \cdot s_{C,A}^1 + \left(\frac{1}{2} - \gamma_{C,A}^{1,1,2}\right) \cdot s_{C,A}^1 + \frac{1}{24} s_{C,B}^1 \text{(assuming triangle inequality)} \\
&= \frac{1}{2} s_{C,A}^1 + \frac{1}{24} s_{C,B}^1 > \frac{1}{2} s_{C,A}^1 + \frac{1}{6} s_{A,B}^1.
\end{aligned}$$

If $\gamma_{C,A}^{1,1,2} > \frac{1}{2}$, then

$$\begin{aligned}
&\gamma_{C,A}^{1,1,2} \cdot s_{C,A}^1 + X_{A,B}^{1,2} \cdot s_{A,B}^1 + \gamma_{C,B}^{1,1,2} \cdot s_{C,B}^1 \\
&\geq \gamma_{C,A}^{1,1,2} \cdot s_{C,A}^1 + \left(X_{A,B}^{1,2} + \gamma_{C,B}^{1,1,2}\right) \cdot s_{A,B}^1 \\
&\geq \gamma_{C,A}^{1,1,2} \cdot s_{C,A}^1 + Z_B^{1,2} \cdot s_{A,B}^1
\end{aligned}$$

$$> \frac{1}{2} s^1_{C,A} + \frac{1}{6} s^1_{A,B}.$$

Hence, the added inequalities of Proposition 5.6 induce a higher objective value for the LP solution. Note that $\left(\frac{1}{2} s^1_{C,A} + \frac{1}{6} s^1_{A,B}\right)$ is the value of total setup time in Figure 5.8.

5.6 Column Generation Method

We use the column generation approach for the solution of the PPMSP in order to exploit its inherent structure through this decomposition technique. The relationships between the variables and constraints of the PPMSP are shown in Figure 5.9.

5.6.1 Decomposition Schemes

There are the following alternative decomposition schemes that can be used:

HMATSP: The subproblem solves the HMATSP to fix the campaign sequence in each bay;

HMATSP2: This is similar to the HMATSP scheme; the subproblem disaggregates by periods and bays.

B&S: The subproblem solves a Batching and Sequencing (B&S) problem to determine the number of batches and the campaign sequence in each bay;

Knapsack: The subproblem solves an integer knapsack problem to determine the number of batches in each period and each bay;

LSP: The subproblem solves a lot-sizing problem to determine the number of batches and production level for each product family;

LSP2: The subproblem solves a lot sizing problem, and also determines the numbers of campaigns;

UCLSP: The subproblem solves an uncapacitated lot-sizing problem.

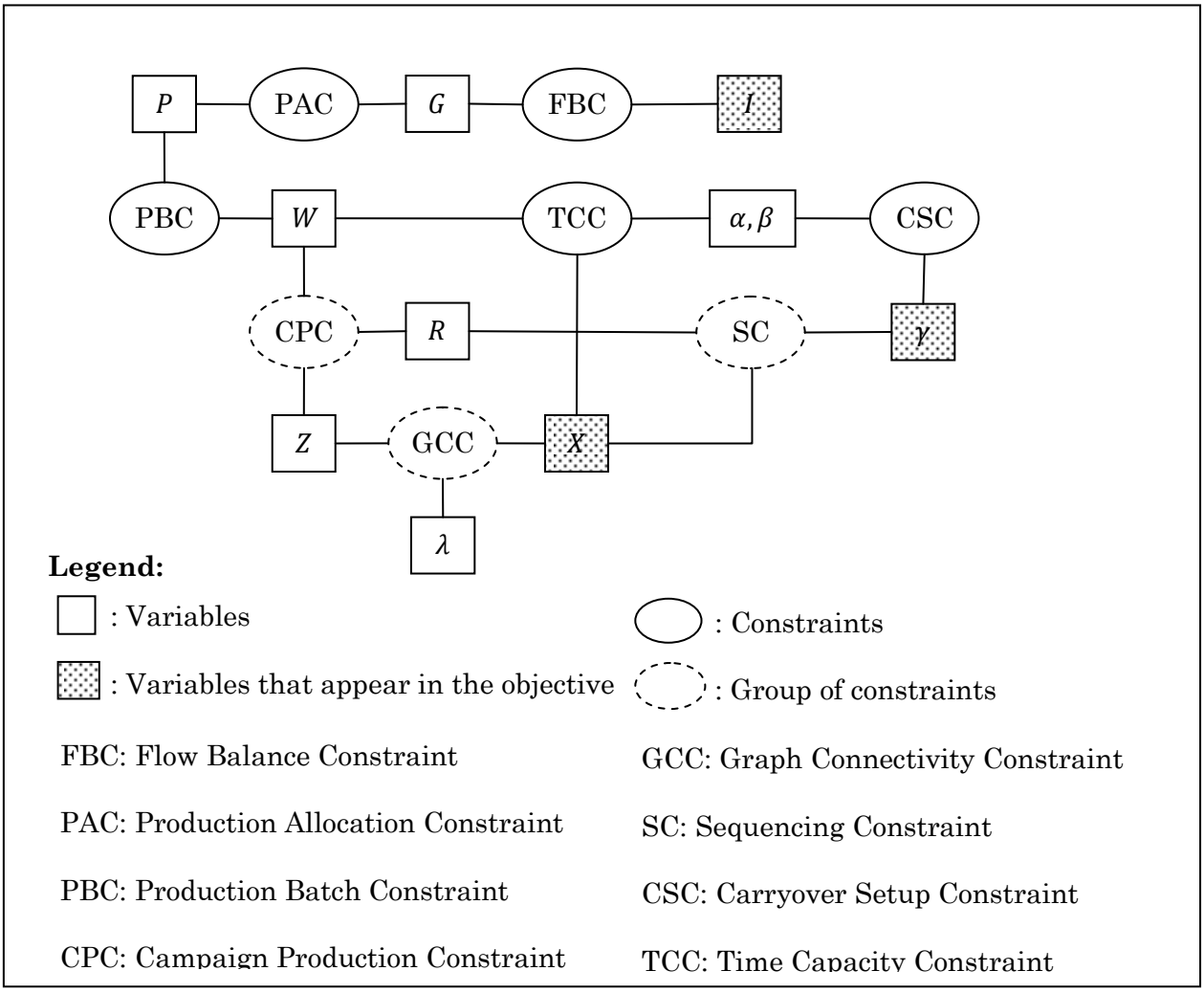


Figure 5.9: Variables and constraints in model PPMSP

- **HMATSP**

The subproblem determines the number of campaigns (R) and setup sequences (X, γ), according to a relaxed version of the Time Capacity Constraint and other constraints. The production indicator vector (Z) is also determined in the subproblem, but does not appear in the coefficients of a column (see Figure 5.10). Note that the subproblem is decomposable by bays. The advantage of this decomposition scheme is that the subproblem is decomposable

and relatively easy to solve. However, since the time capacity constraint is relaxed (by omitting production time), the columns may be infeasible. The problem formulations for this decomposition scheme are as follows:

Sets:

Λ^k: Index set of the columns corresponding to bay k, $\forall k \in M$.

Parameters:

$(\_)_\iota$: Value of subproblem variable "$\_$" fixed in the ι^{th} column of bay k, $\forall k \in M$, $\iota \in \Lambda^k$. Note that index k is implicitly indicated by the enclosed variable "$\_$".

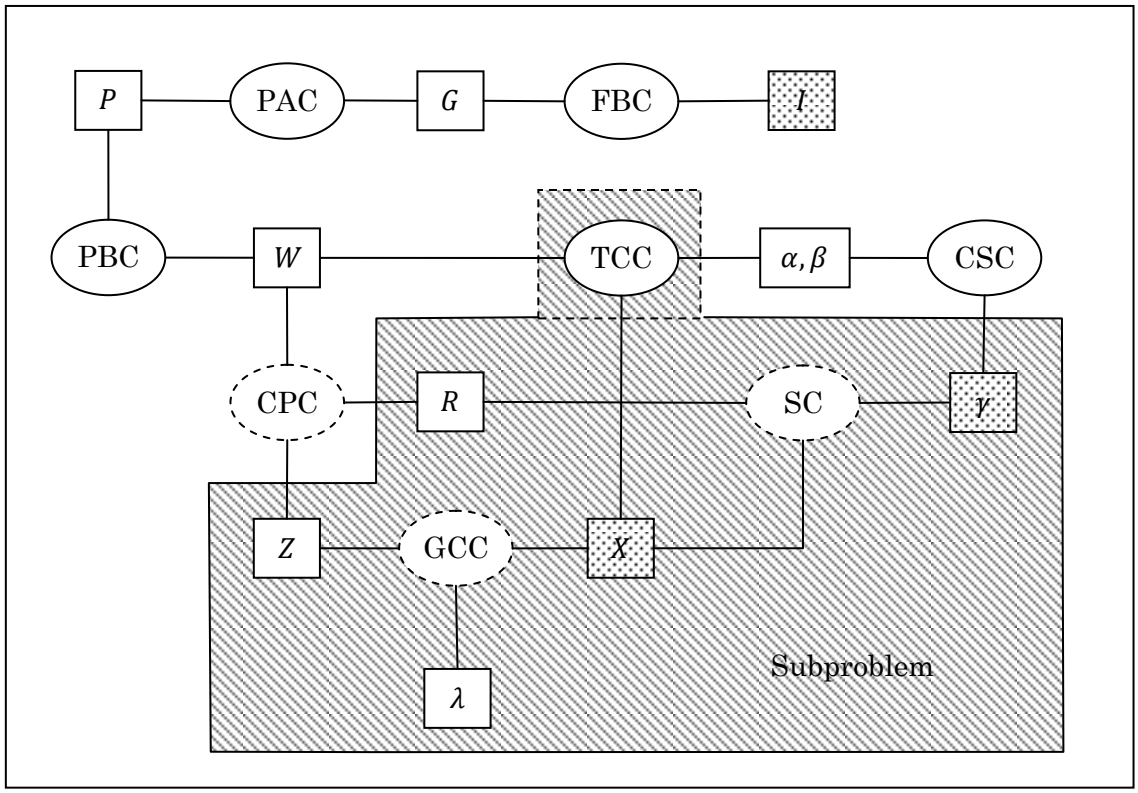


Figure 5.10: HMATSP decomposition scheme

Variables:

Q_ι^k: Binary variable indicating the inclusion/exclusion of the ι^{th} column of bay k, $\forall k \in M, \iota \in \Lambda^k$, and relaxed to take continuous values.

v^k: Dual variable associated with the Convexity Constraint (5.54), $\forall k \in M$.

$\mu^{k,t}$: Dual variable associated with the Time Capacity Constraint (5.55), $\mu^{k,t} \leq 0$, $\forall k \in M, t \in T$.

$\varsigma^{k,t}$: Dual variable associated with Constraint (5.56) or (5.57), $\forall k \in M, t \in T$.

$\xi_f^{k,t}$: Dual variable associated with Constraint (5.58), $\xi_f^{k,t} \leq 0$, $\forall k \in M, t \in T, f \in F^k$.

$\nu_f^{k,t}$: Dual variable associated with Constraint (5.59), $\nu_f^{k,t} \geq 0$, $\forall k \in M, t \in T, f \in F^k$.

$\phi_f^{k,t}$: Dual variable associated with Constraint (5.60), $\phi_f^{k,t} \leq 0$, $\forall k \in M, t \in T, f \in F^k$.

The problem decomposition is as follows:

HMATSP-Master Program:

$$\text{Minimize} \sum_{t\in T}\sum_{f\in F}\sum_{i\in N_f} h_{f,i} I_{f,i}^t + \sigma \sum_{k\in M}\sum_{f\neq g\in F^k} s_{f,g}^k \sum_{\iota\in\Lambda^k} Q_\iota^k \left[\sum_{t\in T}\left(X_{f,g}^{k,t}\right)_\iota + \sum_{t_1<t_2\in\hat{T}}\left(\gamma_{f,g}^{k,t_1,t_2}\right)_\iota\right]$$

subject to: Constraints (5.1)-(5.3)

$$\sum_{\iota\in\Lambda^k} Q_\iota^k = 1, \qquad \forall t \in T, k \in M \tag{5.54}$$

$$\sum_{f\in F^k}\sum_{i\in N_f\cap N^k} p_{f,i}^k W_{f,i}^{k,t} + \sum_{f\neq g\in F^k} s_{f,g}^k \sum_{\iota\in\Lambda^k} Q_\iota^k \left(X_{f,g}^{k,t}\right)_\iota + \alpha^{k,t} + \beta^{k,t} \leq c^{k,t}, \qquad \forall t \in T, k \in M \tag{5.55}$$

$$\sum_{f,g\in F^k} s_{f,g}^k \sum_{\iota\in\Lambda^k} Q_\iota^k \left(\gamma_{f,g}^{k,t-1,t}\right)_\iota = \beta^{k,t}, \qquad \forall k \in M, t = 1 \tag{5.56}$$

$$\sum_{f,g\in F^k} s_{f,g}^k \sum_{\iota\in\Lambda^k} Q_\iota^k \left(\gamma_{f,g}^{k,t-1,t}\right)_\iota = \alpha^{k,t-1} + \beta^{k,t}, \qquad \forall k \in M, t \in T; t > 1 \tag{5.57}$$

$$\sum_{i\in N_f\cap N^k} W_{f,i}^{k,t} \le l_f^{k,t} \sum_{\iota\in\Lambda^k} Q_\iota^k \left(R_f^{k,t}\right)_\iota, \qquad \forall k\in M, t\in T, f\in F^k \tag{5.58}$$

$$\sum_{i\in N_f\cap N^k} W_{f,i}^{k,t} \ge \sum_{\iota\in\Lambda^k} Q_\iota^k \left(R_f^{k,t}\right)_\iota, \qquad \forall k\in M, t\in T, f\in F^k \tag{5.59}$$

$$\sum_{i\in N_f\cap N^k} W_{f,i}^{k,t} \le m_f^{k,t} \sum_{\iota\in\Lambda^k} Q_\iota^k \left(Z_f^{k,t}\right)_\iota, \qquad \forall k\in M, t\in T, f\in F^k \tag{5.60}$$

$$Q_\iota^k \ge 0, \qquad \forall k\in M, \iota\in\Lambda^k$$

$$I_{f,i}^t \ge 0, G_{f,i}^t \ge 0, \qquad \forall t\in T, f\in F, i\in N_f$$

$$P_{f,i}^{k,t} \ge 0, W_{f,i}^{k,t} \in \mathbb{Z}^+, \qquad \forall t\in T, f\in F, i\in N_f, k\in M^i.$$

HMATSP-Subproblem, $\forall k\in M$:

$$\text{Minimize} \sum_{f\ne g\in F^k} s_{f,g}^k \left[\sum_{t\in T}\left(\sigma-\mu^{k,t}\right)X_{f,g}^{k,t} + \sum_{t_1<t_2\in\hat{T}}\left(\sigma-\varsigma^{k,t_2}\right)\gamma_{f,g}^{k,t_1,t_2}\right] - \upsilon^k + \sum_{f\in F^k}\sum_{t\in T}\left(\xi_f^{k,t} l_f^{k,t} R_f^{k,t} + \nu_f^{k,t} R_f^{k,t} + \phi_f^{k,t} m_f^{k,t} Z_f^{k,t}\right)$$

subject to: Constraints(5.5)-(5.10), (5.16)-(5.22)

$$R_f^{k,t} \le m_f^{k,t} Z_f^{k,t}, \qquad \forall f\in F^k, t\in T \tag{5.61}$$

$$\sum_{f\in F^k} \min_{i\in N_f\cap N^k}\left\{p_{f,i}^k\right\} R_f^{k,t} + \sum_{f\ne g\in F^k} s_{f,g}^k X_{f,g}^{k,t} \le c^{k,t} \tag{5.62}$$

$$R_f^{k,t}\in\mathbb{Z}^+, Z_f^{k,t}\in\{0,1\}, \qquad \forall t\in T, f\in F^k$$

$$X_{f,g}^{k,t}\in\mathbb{Z}^+, \qquad \forall t\in T, f\ne g\in\hat{F}^k$$

$$\gamma_{f,g}^{k,t_1,t_2}\in\{0,1\}, \qquad \forall t_1,t_2\in\hat{T}, f,g\in F^k; t_1<t_2$$

$$\lambda_{f,g}^{k,t,u}\ge 0, \qquad \forall t\in T, u, g\in F^k, f\in\hat{F}^k; f\ne g.$$

Constraint (5.61) is based on Constraint (5.24), which is induced by (5.14) and (5.15), and dictates that $Z_f^{k,t} = 1$ if $R_f^{k,t} > 0$. Constraint (5.62) is a relaxed version of the Time Capacity Constraint (5.4). If the optimal objective value of the subproblem is negative, the column index ι is increased by 1 and the corresponding values of $R_f^{k,t}$, $Z_f^{k,t}$, $X_{f,g}^{k,t}$ and $\gamma_{f,g}^{k,t_1,t_2}$ variables are recorded as coefficients of the new column for the master problem.

- **HMATSP2**

This is similar to the HMATSP scheme. However, the subproblem is further decomposed by periods (along with by bays) (see Figure 5.11). Consequently, the starting and ending campaigns of a period are included in the column so that it can be connected with columns of adjacent periods. Again, the subproblem is easy to solve. However, since the time capacity constraint is relaxed (by omitting production time), there is no guarantee that the generated columns will be feasible. The model formulation of this decomposition scheme is provided in Appendix C.1.

- **B&S**

This scheme is similar to HMATSP, except that the number of batches (W) is included in the subproblem, so that the original Time Capacity Constraint still holds (see Figure 5.12). Note that the subproblem is decomposable by bays. The time capacity constraint is not violated by any column. However, the subproblem is relatively hard to solve as it includes both batching and sequencing decisions. The model formulation of this decomposition scheme is provided in Appendix C.2.

- **Knapsack**

The subproblem determines the number of batches (W) to be produced in each bay during each period according to a relaxed version of the Time Capacity Constraint. This is an integer knapsack problem, with production batches as items (see Figure 5.13). The subproblem is decomposable by bays and periods, and it is relatively easy to solve. However, since the time capacity constraint is relaxed (by omitting setup time), the generated columns may be infeasible. The model formulation of this decomposition scheme is provided in Appendix C.3.

- **LSP**

The subproblem determines the number of batches (W) and the production level (G, P), subject to a relaxed version of the Time Capacity Constraint, Flow Balance Constraint, Production Allocation Constraint, and Production Batch Constraint (see Figure 5.14). The subproblem is decomposable by families. In this decomposition scheme, the Time Capacity Constraint is further relaxed to ignore the conflicts between families that are sharing the same bay. The subproblem is relatively easy to solve. However, since the time capacity constraint is relaxed (by omitting setup times), there is no guarantee on the feasibility of the generated columns. The model formulation of this decomposition scheme is provided in Appendix C.4.

- **LSP2**

This is similar to the LSP decomposition approach, except that it includes the numbers of campaigns in the subproblem (see Figure 5.15). The subproblem is relatively easy to solve. However, the subproblem does not contain any feasibility constraint for the pattern of campaigns. Therefore, it tends to generate infeasible columns. The model formulation of this decomposition scheme is provided in Appendix C.5.

- **UCLSP**

In this scheme, the Time Capacity Constraint is not included in the subproblem, as un-capacitated lot-sizing is easier to solve (see Figure 5.16). The subproblem has a very simple structure. The generated columns are relatively simple. The subproblem is decomposable by families and is easy to solve. However, since several sets of constraints are excluded from the subproblem, there is no guarantee on the feasibility of the generated columns. The model formulation of this decomposition scheme is provided in Appendix C.6.

The corresponding separation of decision variables for each decomposition scheme is summarized in Table 5.1. Note that the subproblem variables whose values determine the coefficients of the columns inserted into the master program are highlighted in bold font.

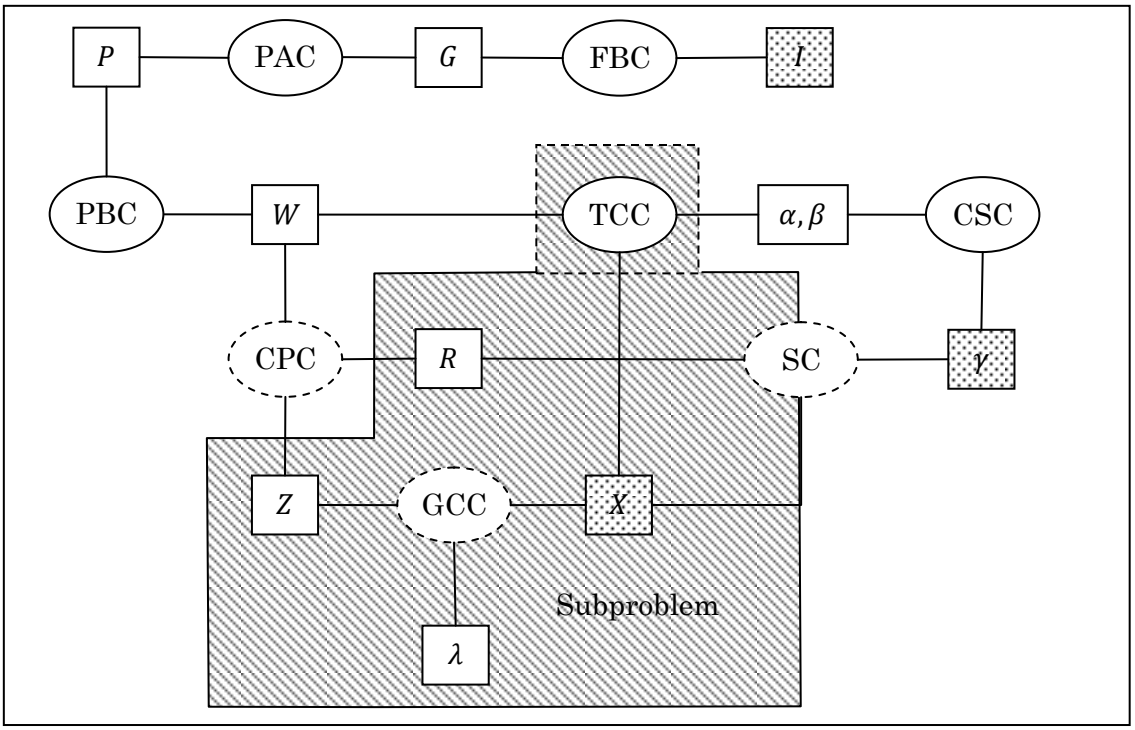


Figure 5.11: HMATSP2 decomposition scheme

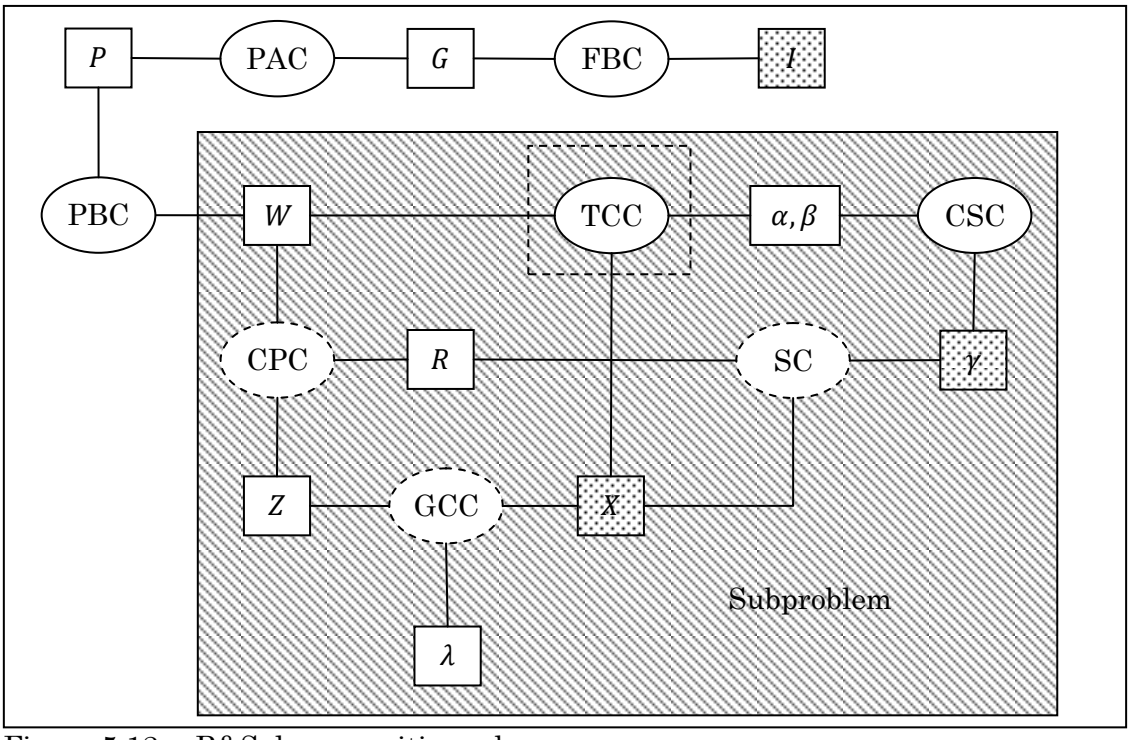


Figure 5.12: B&S decomposition scheme

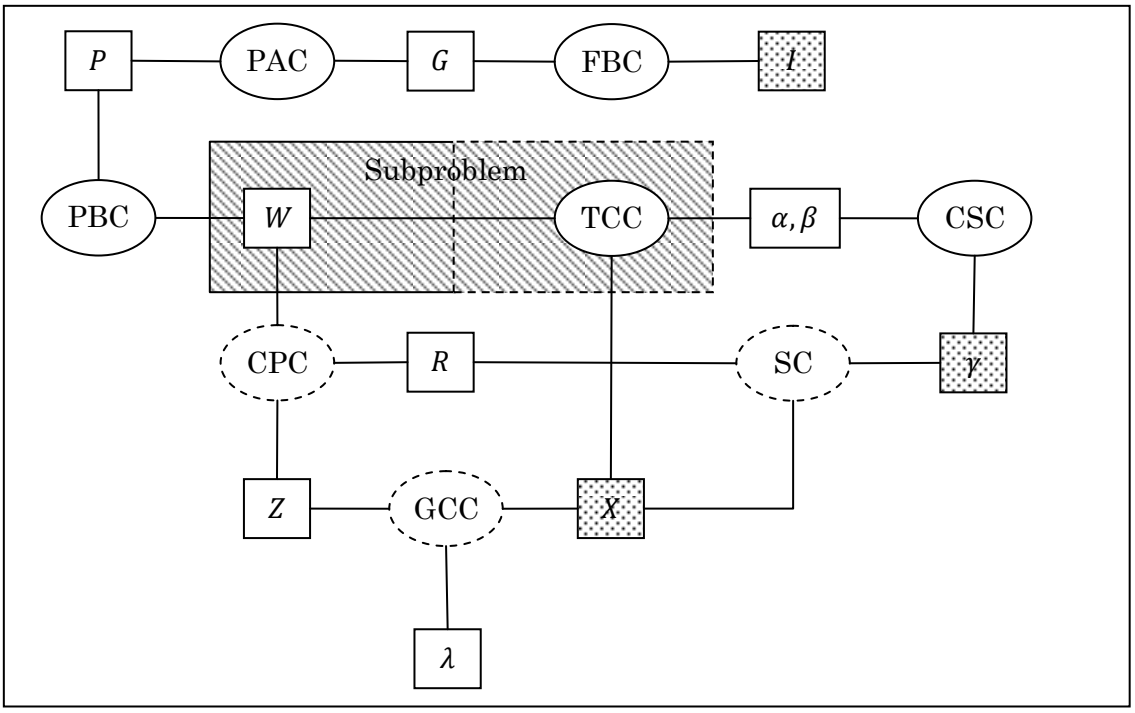


Figure 5.13: Knapsack decomposition scheme

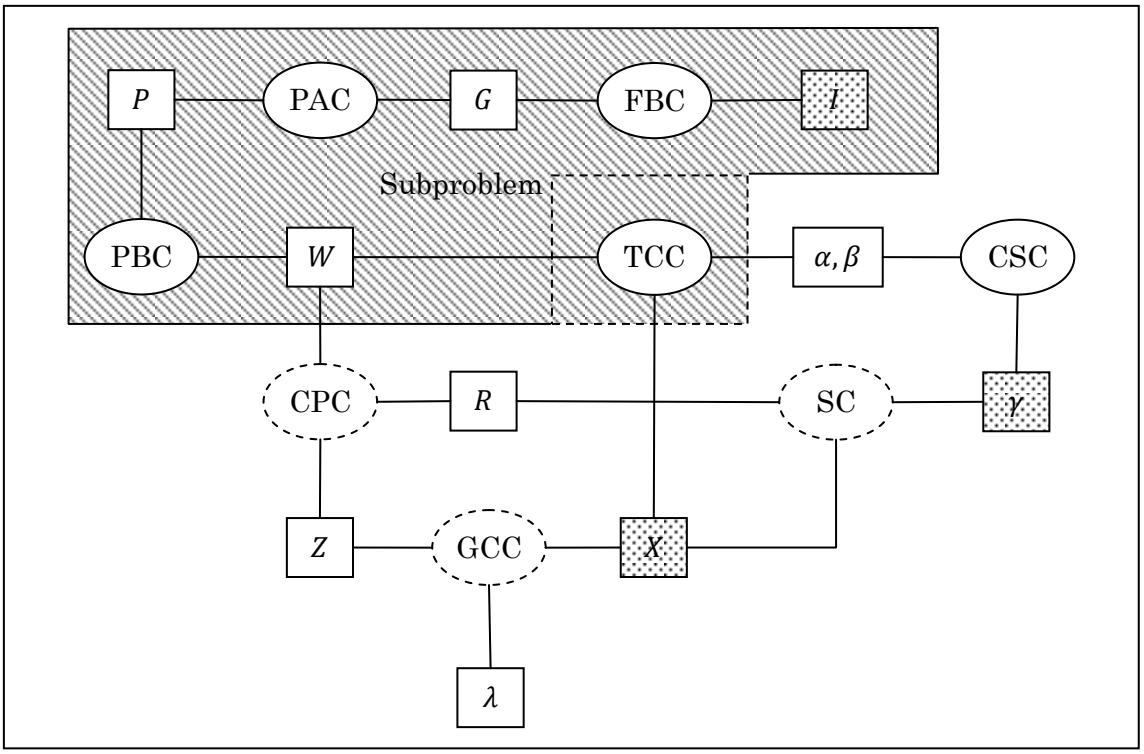


Figure 5.14: LSP decomposition scheme

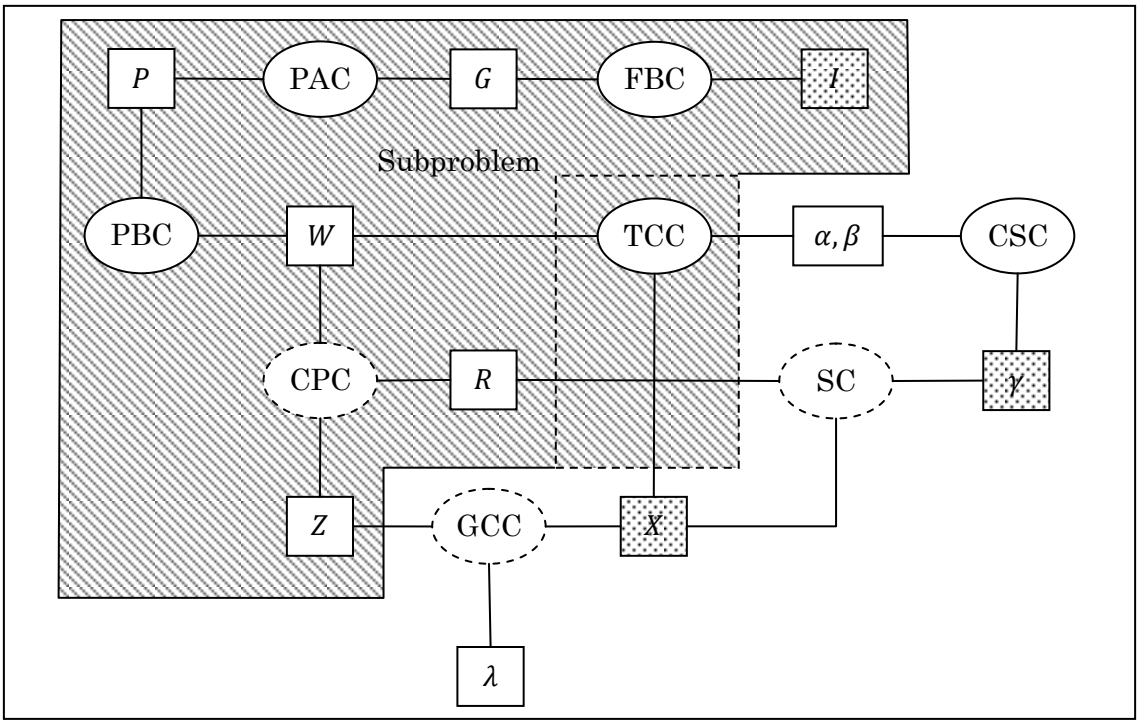


Figure 5.15: LSP2 decomposition scheme

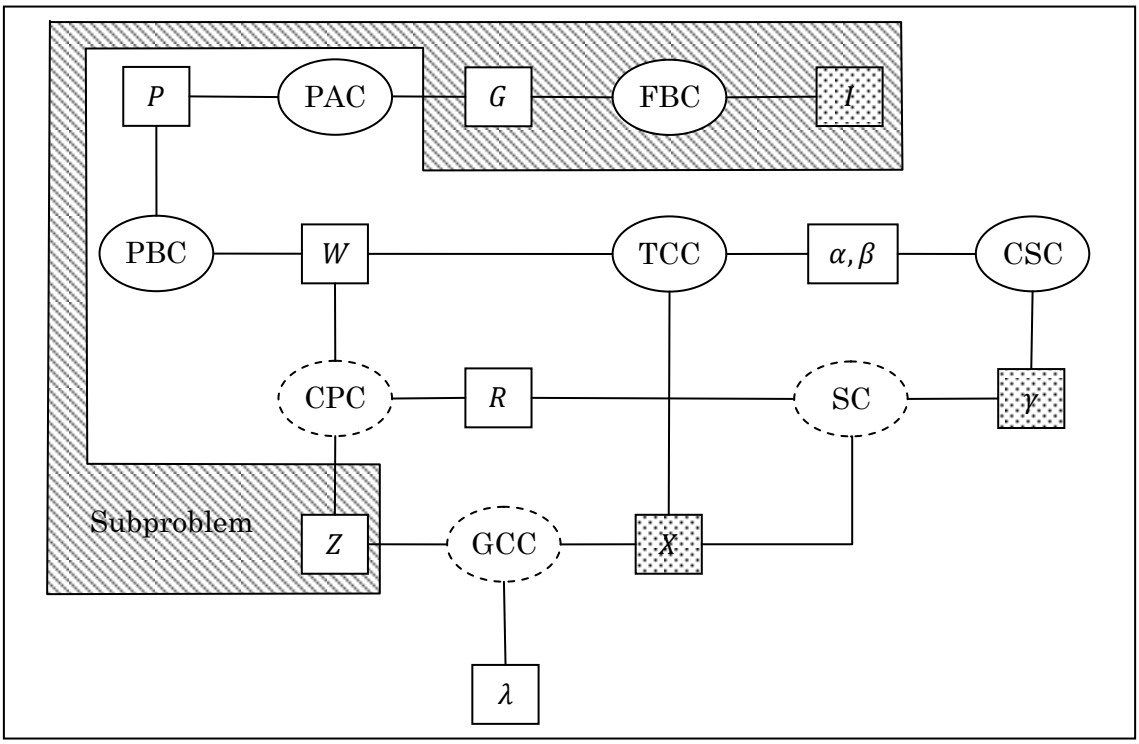


Figure 5.16: UCLSP decomposition scheme

Table 5.1: Summary of decomposition schemes for column generation

Decomposition Scheme	Stage I variables	Stage II variables
HMATSP	I, G, W, α, β	$\boldsymbol{R}, Z, \boldsymbol{X}, \boldsymbol{\gamma}, \lambda$
HMATSP2	$\mathrm{I}, \mathrm{G}, \mathrm{W}, \alpha, \beta, \gamma$	$\mathrm{R}, \mathrm{Z}, \mathrm{X}, \lambda$
B&S	I, G	$\boldsymbol{W}, R, Z, X, \gamma, \lambda, \alpha, \beta$
Knapsack	$I, G, R, Z, X, \gamma, \lambda, \alpha, \beta$	$\boldsymbol{W}$
LSP	$R, Z, X, \gamma, \lambda, \alpha, \beta$	$I, G, \boldsymbol{W}$
LSP2	$X, \gamma, \lambda, \alpha, \beta$	$I, G, \boldsymbol{W}, \boldsymbol{R}, \boldsymbol{Z}$
UCLSP	$W, R, X, \gamma, \lambda, \alpha, \beta$	$I, \boldsymbol{G}, \boldsymbol{Z}$

5.6.2 Generation of an Initial Feasible Solution

In order to accelerate the convergence of the column generation method, it is desirable to have a set of good initial columns, instead of starting from scratch with a Phase-I linear program. Accordingly, we implement the following heuristic methods to obtain a feasible solution for the PPMSP, which can be used to construct a set of initial columns.

Generation of a Feasible Solution from Scratch

- **Period-wise Heuristic**

This is an iterative method. During each iteration, the numbers of batches (W) for a period are regarded as integer variables. The W-variables for other periods are either fixed by previous iterations, or are assumed to be continuous variables to make the problem easier to solve. If a feasible solution is found, the values of the integer W-variables are fixed as parameters for future iterations. We start from the final period and assume the corresponding W-variables to be integral, and then, proceed backwards (towards the first period), fixing W-values for one period at a time. If a feasible solution cannot be found during any iteration, the most recently-fixed W-values are allowed to change as integers. This problem is re-solved to obtain a feasible solution.

- **Level-wise Heuristic**

In this method, we solve the problem by decoupling items in the BOM tables. Note that

external demands only exist for end products. In the first iteration, we ignore all BOM relations and solve the problem to satisfy the external demands only. The values of G-variables in the solution are recorded for all the products. We regard these values as lower bounds for the G-variables in future iterations. We also determine the amounts of other products that are needed to produce the required end products. These internal demands will be satisfied in the next iteration, during which the problem is modified to recognize these demands and is solved without lowering the current production levels. The heuristic continues level by level until the cascading effect of internal demands comes to an end. If a feasible solution is obtained in the final iteration, all demand would have been consequently satisfied.

- **Relaxation-Fixing Heuristic**

This method consists of two steps. In the first step, the W-variables are relaxed to take continuous values. The problem is solved to obtain a solution with integral R-values (number of campaigns). These R-values are recorded and fixed as parameters. In the second step, the problem is re-solved, regarding the W-variables as integer variables. Since the numbers of campaigns have been fixed, the problem in the second step is relatively easy to solve.

- **Relaxation-Bounding Heuristic**

The Relaxation-Bounding heuristic is based on the Relaxation-Fixing heuristic with several revisions. First, instead of recording only the optimal solution in the first step, multiple incumbent solutions are kept as candidates for the second step. Second, since relaxing the W-variables to take continuous values (in the first step) tends to yield an underestimate of the R-values, it is preferable to treat these R-values as lower bounds rather than as fixed parameters in the second step. The refined method is described as follows:

Step 1. Relax the W-, X- and γ-variables to take continuous values, and solve the problem using CPLEX to obtain multiple incumbent values of R.

Step 2. Modify the formulation PPMSP, so that, for each bay, one set of values of R is

selected from the pool of incumbents to serve as lower bounds for the numbers of campaigns as follows:

$$\sum_{\iota \in \Xi} Q_\iota^k \left(R_f^{k,t}\right)_\iota \leq R_f^{k,t}, \qquad \forall k \in M, t \in T, f \in F^k$$

$$\sum_{\iota \in \Xi} Q_\iota^k = 1, \qquad \forall k \in M$$

$$Q_\iota^k \in \{0,1\}, \qquad \forall k \in M, \iota \in \Xi$$

where, Ξ denotes the index set of incumbent solutions, Q_ι^k is the selection variable, and $\left(R_f^{k,t}\right)_\iota$ is the number of campaigns in the ι^{th} incumbent solution.

The modified problem is solved without relaxing any integrality restriction. Note that the relaxation adopted in Step 1 makes the resulting problem easier to solve. Moreover, since the PPMSP is only allowed to search over a restricted set of solutions in Step 2, the computational effort is significantly reduced. Yet the restriction is quite flexible as it allows the production patterns of different incumbent solutions to be re-combined on a bay-by-bay basis, which often leads to good quality solutions.

5.6.3 Refinements of the Column Generation Method

- **Dual Stabilization Technique**

Note that to implement a decomposition scheme, the coupling constraints that contain both master problem variables and subproblem variables are revised by replacing the subproblem variables with convex combinations of their extreme point values. This transformation may introduce additional degeneracy in the master problem, as pointed out by Tebboth (2001), which slows convergence and requires a large number of iterations. Also, the dual solution tends to oscillate excessively, particularly during initial iterations. Dual stabilization techniques can be used to alleviate this effect. In particular, we consider the Box-Step method to restrict the range of dual variables in the master problem. The following bounds are added to each dual variable, which corresponds to a coupling

constraint in the Restricted Master Problem (RMP):

$$\mu \in [\hat{\mu} - \delta|\hat{\mu}|, \hat{\mu} + \delta|\hat{\mu}|],$$

where $\hat{\mu}$ is the dual variable value obtained during the previous iteration. The initial value of δ is set to 0.1, and is doubled in the next iteration if the imposed dual bounds are active in the current solution. In the primal formulation of the RMP, this strategy translates into additional slack and surplus variables in the corresponding coupling constraint. These variables are penalized with a unit cost of δ in the objective function.

- **Subproblem Heuristic**

For some problem decomposition schemes, the subproblem is relatively difficult to solve. To speed up the column generation method, we can seek good solutions with a negative reduced cost without solving the subproblem to optimality. For example, we can solve the subproblem to a 10% optimality gap and use the best solution found so far to generate a new column. In case that the B&S decomposition scheme is used, we can apply the Relaxation-Bounding Heuristic to the subproblem. During the second step of the heuristic, we only attempt to find a column having a negative reduced cost. Note that, during the first step of the heuristic method, a relaxed version of the subproblem is solved. Therefore, we also obtain a lower bound on the reduced cost for that subproblem. Consequently, a lower bound can be derived for the objective value of the master problem.

- **Complementary Column Generation**

In a column generation algorithm, the RMP is solved as a linear program (LP). At the end of column generation, we may attempt to solve the problem as a mixed-integer program to obtain integer feasible solutions to the original problem. When the subproblem is decomposed into multiple problems, each subproblem gives rise to a separate set of columns. These columns are not necessarily compatible with each other, as their combination may fail to satisfy the coupling constraints in the master problem. Therefore, it is desirable to generate complementary columns that are mutually compatible, so that a good quality integer feasible solution to the master problem can be obtained afterwards. A complementary column generation algorithm developed by Ghoniem and Sherali (2009) is

as follows (see the steps below): For each subproblem, first search for a column having the most negative reduced cost. If a negative reduced cost column is found, an attempt is made to generate a set of complementary columns, one for each of the other subproblems. To achieve compatibility of these columns, the subproblems are solved sequentially, and additional constraints are added to account for the effect of existing columns within the current set. The procedure is described below:

Complementary Column Generation for the PPMSP

For each subproblem:
  Find the column with the most negative reduced cost;
  If the reduced cost is negative
    Add this column to an empty compatible set $\mathcal{C}$;
    For each of the other subproblems:
      Find the column that is a best addition to $\mathcal{C}$;
    End for
    Add columns in $\mathcal{C}$ to the master problem;
  End if
End for

5.7 Local Search Heuristic Methods

In Section 5.6.2, we have introduced heuristic methods to generate initial feasible solutions, which help in speeding up the convergence of the column generation method. Using these initial feasible solutions as a starting point, we can also develop local search heuristic methods to improve the solution quality. The goal here is to generate good feasible solutions for large-sized problem instances with moderate computational effort.

A local search heuristic method is typically an iterative procedure. In each step, the

search area is restricted to a number of candidate solutions in order to avoid the complexity of optimizing over the entire solution space. Such candidate solutions are usually determined based on their vicinity to the current solution value, and hence, they are also collectively called the local search neighborhood. Upon finding the best solution in the neighborhood, the local search step is repeated, starting from the current best solution found. This iterative procedure continues, until no improvement can be obtained.

5.7.1 Search by Problem Partitioning

In this subsection, we construct several local search procedures, which rely on the partitioning of the PPMSP based on its inherent problem structure. In other words, the decision variables in the PPMSP are divided into groups according to a given criterion and a local search neighborhood is defined by allowing only one group of decision variables to change at a time.

- **Local search by bays**

Step 1. For each bay, do the following:

a. Solve a restricted problem, which follows the current best solution, but allows production in the current bay to change.

b. If the objective value obtained is smaller than the best recorded value, the best solution is updated accordingly.

Step 2. If there is no improvement during the last round of local search, stop; otherwise, go to Step 1.

- **Local search by product families**

Step 1. For each product family, do the following:

a. Solve a restricted problem, where the numbers of batches follow the best solution found, except that the W-values of the current family are allowed to change. Note that all the R- and X-variables are allowed to change.

b. If the objective value obtained is smaller than the best recorded value, update

the best solution to be the local optimal solution just obtained.

Step 2. If there is no improvement during the last round of local search, stop; otherwise, go to Step 1.

- **Local search by periods**

Step 1. For each time period, do the following:

a. Solve a restricted problem, where the numbers of batches and campaigns follow the best solution found, except that the production in the current period is allowed to change.

b. If the objective value obtained is smaller than the best recorded value, update the best solution to the local optimal solution just obtained.

Step 2. If there is no improvement during the last round of local search, stop; otherwise, go to Step 1.

5.7.2 Guided Search

In Step 2 of the Relaxation-Bounding heuristic method (Section 5.6.2), we have observed that adding additional bound restrictions on the R-variables make the resulting problem relatively easy to solve. We have applied lower bounds on the R-variables therein. An iterative local search method can be further developed by changing the values and directions of the bounds adaptively in order to reach better solutions. We call this heuristic method as the *Guided Search*.

Assume that a feasible solution has been obtained by using the Relaxation-Bounding heuristic. We start with a downward search direction by adding the following bound restrictions to the PPMSP:

$$R_f^{k,t} \le \hat{R}_f^{k,t}, \qquad \forall k \in M, t \in T, f \in F^k \tag{5.63}$$

where $\hat{R}_f^{k,t}$ is the corresponding value in the given feasible solution. In order to determine a good search direction, we first solve the above restricted problem as a linear program. If the

dual variable values associated with Constraint (5.63) are negative, the corresponding constraint is binding. We flip the bound direction accordingly, as follows:

$$R_f^{k,t} \geq \hat{R}_f^{k,t}\,, \qquad \forall k \in M, t \in T, f \in F^k \tag{5.64}$$

After the above adjustments to the search directions, we solve the restricted problem as an MIP. The local optimal solution obtained is then recorded if it is the best solution found so far. The bound values ($\hat{R}_f^{k,t}$) are also updated according to this new solution. We, then, start a new iteration using the updated bound values and search directions. This iterative procedure continues until no adjustment is made to the bound values and search directions, or a prescribed time/iteration limit is reached.

5.8 Computational Investigation

We carried out a computational investigation to test and evaluate the efficacy of the aforementioned modeling and solution approaches for the PPMSP. The testbed of data used for our experimentation is based on the real-life data for the period (2006-2008) collected from Boehringer-Ingelheim's Primary Pharmaceutical Manufacturing Plant located in Petersburg, Virginia. Various problem characteristics were summarized statistically, and then, used to generate sample problem instances.

5.8.1 Generation of Problem Instances

Our algorithm for generating sample problems is comprised of five components, corresponding to the determination of product families, bays, routes, setups and demands, respectively. We present detailed steps of these components in the following.

- **Product Families**

Step 1. Generate $|N|$ different products. Determine the holding cost coefficient $h_{f,i}$ from a continuous uniform distribution $\mathcal{U}(\underline{h}, \bar{h})$; determine the initial inventory level $I_{f,i}^0$

from $\mathcal{U}(\underline{I}^0, \bar{I}^0)$; determine the end inventory level $E_{f,i}$ from $\mathcal{U}(\underline{E}, \bar{E})$.

Step 2. Partition the products into $|F|$ different families as follows:

a. Consider a sequence of $(|N| - 1)$ positions. Randomly select $(|F| - 1)$ positions as separators to construct $|F|$ families.

b. Fill each of the unselected positions with a product, and then, add one more product to each of the families obtained in step 2a.

Step 3. Randomly create the BOM structure for each product family:

a. Assign distinct ranks to all products in the family.

b. Randomly generate BOM relations, which are consistent with the ranks (going from the high ranking products to the low ranking products).

c. Generate the ratio for each BOM relation using a uniform distribution: $\mathcal{U}(\underline{b}, \bar{b})$.

Note that Step 2a results in $|F|$ (potentially empty) families with a total number of $(|N| - |F|)$ products. Step 2b further adds $|F|$ products so that the final number of products is $|N|$ and there is at least one product per family. For example, if $|N| = 7$ and $|F| = 3$, one possible outcome is illustrated in Figure 5.17, where white boxes represent positions, shaded boxes represent separators and circles represents products.

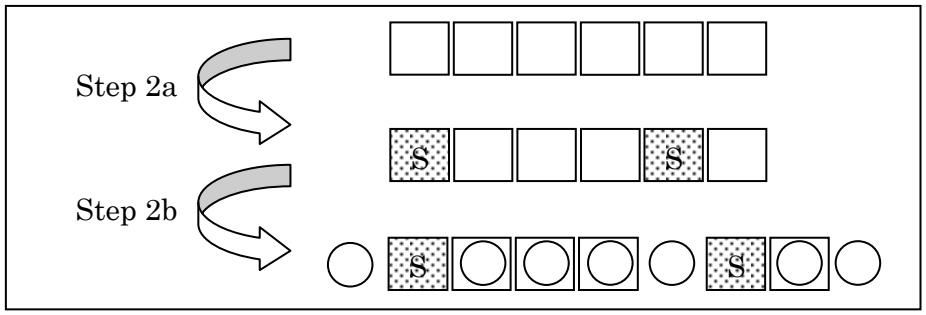


Figure 5.17: Random generation of 7 products in 3 product families

Step 3 is based on the correspondence between a valid BOM structure and a *directed acyclic graph* (DAG), which imposes a partial order over the products. As suggested by Karrer and Newman (2009), we first generate a full order over the products, and then, construct a DAG that is consistent with this full order. A detailed description of Step 3b is as follows:

i. Calculate the number of edges: $|E_f| = 1.5 \times \rho \cdot (|N_f| - 1)^{\dagger}$ according to a graph density factor ρ, which follows a uniform distribution $\mathcal{U}(\underline{\rho}, \bar{\rho})$.

ii. Fix the in-degrees (δ_i) and out-degrees ($\hat{\delta}_i$) of the $|N_f|$ nodes as follows:
Generate random numbers A_i from a discrete uniform distribution on $\{0,1,\dots,|E_f|\}$. Determine their order statistics, represented as $A_{(i)}$, $\forall i = 1,\cdots,|N_f| - 2$, and the values of δ_i according to the following system of equations:

$$\begin{cases} \sum_{j=1}^{i} \delta_j = A_{(i)}, & \forall i = 1,\cdots,|N_f| - 2 \\ \sum_{j=1}^{|N_f|-1} \delta_j = |E_f|, & \\ \sum_{j=1}^{|N_f|} \delta_j = |E_f|; & \end{cases}$$

For $i = 1,\dots,|N_f|$, determine the value of $\sum_{j=1}^{i} \hat{\delta}_j$ from a discrete uniform distribution over $[\sum_{j=1}^{i-1} \hat{\delta}_j, \sum_{j=1}^{i-1} \delta_j] \cap \mathbb{Z}$; ‡
Calculate the individual values of $\hat{\delta}_i$ based on the values of their summations determined above, $\forall i = 1,\dots,|N_f|$.

iii. Generate the DAG by randomly matching the in-degrees and out-degrees of the nodes, starting from the node with the lowest of rank.

For example, if the in-/out-degrees are as given in Table 5.2, three different DAGs can be generated (see Figure 5.18).

---

† A multiplier of 1.5 is applied to compensate for redundant edges, which can be generated in later steps.
‡ This ensures that the following inequality holds for a valid DAG to exist:
$\sum_{j=1}^{i-1} \delta_j \geq \sum_{j=1}^{i} \hat{\delta}_j$, $\forall i = 1,\dots,|N_f|$.

Table 5.2: In-degrees and out-degrees of nodes (products) in a DAG

Product i	1	2	3	4	5	6	7
δ_i	2	2	1	0	1	0	0
$\hat{\delta}_i$	0	1	0	3	0	2	0

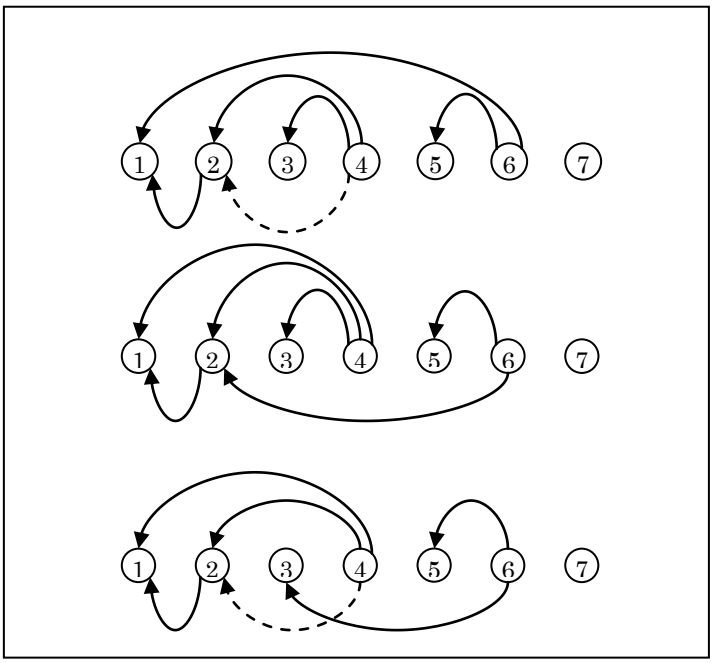


Figure 5.18: Different DAGs with the same in-degrees and out-degrees

- **Bays**

We assume the time capacity to be the same for all the bays in any given period. Hence we create $|M|$ bays and determine their common time capacity for each of the $|T|$ periods by sampling from a continuous uniform distribution $\mathcal{U}(\underline{c}, \bar{c})$.

- **Routes**

Step 1. Randomly determine matches between families and bays.

The number of such matches is calculated as:

$$m = \min\{|M| \cdot |F|, [\max\{|M|, |F|\} + \lfloor |M| \cdot |F| \cdot \rho_F + 0.5 \rfloor]\},$$

where ρ_F is a family-wise route density factor.

Randomly select m family-to-bay matches from the $|M| \cdot |F|$ possible combinations. Check the selections to ensure that there is at least one family selected for each bay, and there is at least one bay selected for each family.

Step 2. Randomly determine matches between the products and bays.

Determine the number of compatible matches as:

$$m_f = \min\left\{|M_f| \cdot |N_f|, \begin{pmatrix} \max\{|M_f|, |N_f|\} \\ +\lfloor |M_f| \cdot |N_f| \cdot \rho_P + 0.5 \rfloor \end{pmatrix}\right\},$$

where ρ_P is a product-wise route density factor.

Randomly select m_f matches between a product in N_f and a bay in M_f from the $|M_f| \cdot |N_f|$ possible combinations. Check the selections to ensure that there is at least one product of family f selected for each compatible bay in M_f, and there is at least one bay selected for each product of family f. Note that the set of compatible bays (M_f) has been fixed in Step 1.

Step 3. Generate batch processing times from $\mathcal{U}(\underline{p}, \bar{p})$; and generate maximum batch sizes from $\mathcal{U}(\underline{q}, \bar{q})$.

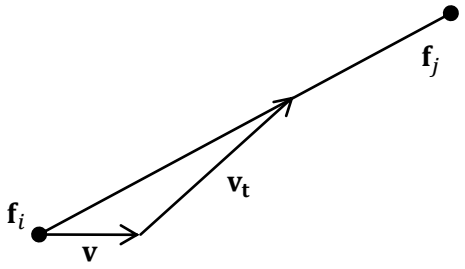
- **Changeover Setups**

For each bay $k \in M$, we use the following steps to generate its setup time matrix, which is asymmetric, and it satisfies the triangle inequality:

Step 1. Generate $|F_k|$ random points in a d-dimensional unit cube ($d = 5$ for example).

Step 2. Randomly generate a vector $\mathbf{v}$ in $\mathbb{R}^d$, with $\|\mathbf{v}\| = \frac{1}{2}$.

Step 3. Calculate the setup time from point $\mathbf{f}_i$ to point $\mathbf{f}_j$ as $\frac{2\|\mathbf{f}_j - \mathbf{f}_i\|}{\sqrt{3+D}+D}$, where $D \equiv \frac{\mathbf{v} \cdot (\mathbf{f}_j - \mathbf{f}_i)}{\|\mathbf{v}\| \cdot \|\mathbf{f}_j - \mathbf{f}_i\|}$. Note that the setup time is equal to the shortest traversal time from point $\mathbf{f}_i$ to point $\mathbf{f}_j$ with a speed of $\|\mathbf{v_t}\| = 1$, relative to a flow that has the velocity of **v.**



Step 4. Multiply the setup time by a given setup multiplier z.

- **Demands**

Step 1. Calculate the maximum amount of production for each product based on the assumption that production is continuous and the capacity of each bay is equally shared by compatible products.

Step 2. Determine the tentative demand for each end product j according to the production capacities calculated in Step 1. If product i is consumed (directly, or indirectly) for the production of end product j, reduce the demand for product j accordingly so that the induced amount of product i does not exceed its maximum capacity.

Step 3. Calculate the real demand value for each product i, by multiplying the tentative value (determined in Step 2) with a demand factor, which is generated from a uniform distribution $\mathcal{U}(\underline{\Delta}, \overline{\Delta})$.

We considered two datasets with different problem scales (see Table 5.3) and some common parameter values (see Table 5.4), which are determined according to the real-life data. For Dataset A, we generated 30 problem instances. For Dataset B, which involves larger-sized problems, we considered 10 problem instances.

Table 5.3: Problem scales of different datasets

Parameter		$\|F\|$	$\|N\|$	$\|M\|$	$\|T\|$
Value	Dataset A	5	15	5	12
	Dataset B	7	52	21	12

Table 5.4: Parameter values for problem generation

Parameter	$(\underline{h}, \bar{h})$	$(\underline{I^0}, \bar{I^0})$	$(\underline{E}, \bar{E})$	$(\underline{\rho}, \bar{\rho})$
Value	(0.1, 0.9)	(0,100)	(50,100)	(1.05, 1.5)
Parameter	$(\underline{b}, \bar{b})$	$(\underline{c}, \bar{c})$	ρ_F	ρ_P
Value	(0.23, 2.86)	(672, 744)	0.143	0.224
Parameter	$(\underline{p}, \bar{p})$	$(\underline{q}, \bar{q})$	z	$(\underline{\Delta}, \bar{\Delta})$
Value	(8.5, 168)	(4, 1762.5)	100	(0, 1)

5.8.2 Results of Experimentation

Our computational results were obtained using a computer consisting of a 2.67GHz dual core CPU with 2GByte RAM running OPL IDE 6.3 (which uses CPLEX 12.1 as the underlying solver). We considered two different LP bound values:

Bound I: Determined using the presolve routine of CPLEX before solving the LP relaxation at the root node, without generating any additional cuts via CPLEX.

Bound II: Determined using the presolve routine of CPLEX and the default cuts provided by CPLEX, before solving the LP at the root node.

- **Comparison of Formulations**

First, we compared the performances of several different problem formulations (see Table 5.5) without including any additional valid inequalities. The computational results obtained for these formulations on Dataset A are summarized in Table 5.6, where relative optimality gaps are calculated using the optimal objective value as a denominator. Several of the 30 problem instances for Dataset A were found to be infeasible, and hence, are excluded from the results presented in Table 5.6. In addition, from among the 23 feasible instances, Problem 23 required a much longer CPU time than that required by the others. To avoid it from dominating the calculation of average performance values, we present the results for Problem 23 separately in Table 5.7.

Table 5.5: Different problem formulations

Formulation	Constraints	Production Model	Graph Connectivity Constraint
V1	(5.1), (5.4)-(5.23)	Basic Model	Multiple Commodity Flow
V2	(5.4)-(5.23), (5.25), (5.26), (5.29)	Plant Location	Multiple Commodity Flow
V3	(5.1), (5.4)-(5.18), (5.23), (5.30)-(5.32)	Basic Model	MTZ-based
V4	(5.1), (5.4)-(5.18), (5.23), (5.33), (5.34)	Basic Model	Single Commodity Flow

Table 5.6: Average performance of different formulations for 22 problem instances of Dataset A

Formulation	No. of Non-zero Coefficients	Opt Gap for Bound I	Opt Gap for Bound II*	No. of Nodes	Average Time (sec)
V1	9,643	74.57%	26.35%	**42,529**	**71.7**
V2	13,621	78.49%	**21.35%**	56,492	359.9
V3	8,335	78.70%	26.91%	81,124	131.6
V4	**8,240**	**73.57%**	24.74%	51,247	102.4

*Problem 22 is excluded for this column since it was solved at the root node itself.

Table 5.7: Performance of different formulations for Problem No. 23 of Dataset A

Formulation	No. of Non-zero Coefficients	Opt Gap for Bound I	Opt Gap for Bound II	No. of Nodes	CPU Time (sec)
V1	8,788	**78.50%**	60.50%	41,230,428	56,414.4
V2	12,982	**78.50%**	**53.97%**	**33,974,978**	76,064.5
V3	**7,900**	80.35%	60.11%	*>39,444,900	*>48,466.9
V4	7,768	80.04%	59.70%	39,198,110	**48,900.7**

* Solver was stopped due to memory exhaustion.

Note that Formulation V1 required the least average CPU time for the 22 instances. For Problem 23, it is also very competitive in terms of the CPU time required, although V4 is slightly faster. Formulation V2 used the least number of nodes for Problem 23, and induced the tightest Bound II value for all instances. Also note that Formulation V4 yields the most compact model representation in view of the number of no-zero coefficients.

- **Effectiveness of Valid Inequalities**

Next, we evaluated the effectiveness of the valid inequalities proposed in Section 0 by adding them individually to Formulation V1. Table 5.8 specifies the different cut configurations investigated.

Table 5.8: Cut configurations

Name of Cut	Added Inequalities	Replaced Inequalities
Cut C1	(5.35)	(5.15)
Cut C2	(5.36)	(5.15)
Cut C3	(5.36)	
Cut C4	(5.36), (5.37)	
Cut P	(5.2), (5.3), (5.38)	(5.23)
Cut G	(5.39) or (5.40)*	
Cut X1	(5.41), (5.43), (5.44) and (5.45)*	
Cut X2	(5.46)	
Cut Z	(5.47)	
Cut S	(5.48)-(5.53)	

* The marked constraint is for the Plant Location Model.

The results obtained using these valid inequalities for Dataset A are summarized in Table 5.9 and Table 5.10, where, as before, the results for Problem No. 23 are summarized separately in Table 5.10. Note that the application of any of these inequalities by itself, except for Cut S, results in tightened LP bounds. Although the CPU time required to solve a problem instance does not always decrease after the inclusion of a cut, the effect is

Table 5.9: Average performances of various valid inequalities for 22 problem instances of Dataset A

Cut Added	No. of Non-zero Coefficients	Opt Gap for Bound I	Opt Gap for Bound II*	No. of Nodes	Average Time (sec)
N/A	9,643	74.57%	26.35%	42,529	71.7
C1	9,643	73.34%	26.19%	57,581	123.9
C2	9,643	73.33%	26.34%	61,947	141.2
C3	9,994	73.32%	26.41%	54,161	145.3
C4	10,481	73.26%	26.30%	66,615	116.4
P	10,616	63.34%	**23.33%**	35,019	75.8
G	10,066	63.18%	23.51%	**34,293**	**67.5**
X1	10,964	**42.91%**	24.79%	48,366	116.3
X2	10,966	**42.91%**	25.07%	63,333	148.2
Z	10,074	43.88%	23.83%	86,324	137.6
S	12,763	74.57%	48.92%	1,551,955	3068.7

*Problem 22 is excluded for this column since it was solved at the root node itself.

Table 5.10: Performances of various valid inequalities for Problem No. 23 of Dataset A

Cut Added	No. of Non-zero Coefficients	Opt Gap for Bound I	Opt Gap for Bound II	No. of Nodes	Time (sec)
N/A	8,788	78.50%	60.50%	41,230,428	56,414.4
C1	8,788	78.48%	60.21%	28,702,577	34,708.7
C2	8,788	78.48%	59.41%	42,052,821	50,877.0
C3	9,136	78.48%	59.49%	21,614,523	23,164.3
C4	9,616	78.46%	59.49%	33,366,570	40,655.7
P	9,748	71.07%	**54.10%**	19,085,757	22,452.7
G	9,208	71.07%	54.15%	*>34,548,700	*>39,748.3
X1	10,116	**70.28%**	57.58%	30,478,947	39,719.6
X2	10,116	**70.28%**	57.58%	30,478,947	38,583.7
Z	9,236	73.72%	58.42%	**13,885,268**	**15,886.3**
S	11,788	78.50%	68.36%	15,644,497	22,120.7

* Solver was stopped due to memory exhaustion.

positive for the hardest problem instance (i.e., Problem No. 23—see Table 5.10) for all the cuts except for Cut G, for which the runs exceeded the memory limitations.

We further investigated the simultaneous addition of multiple valid inequalities to the problem formulation. The various cut combinations used are depicted in Table 5.11. We applied these combinations on Formulation V1 and solved the problems in Dataset A. The results obtained are summarized in Table 5.12 and Table 5.13. Note that Combination 1 yielded the most favorable results, being robustly effective for all problem instances (including Problem No. 23).

Table 5.11: Combinations of cuts

Cut Combination	**Cuts Included**
Combination 1	Cut C3, Cut G, Cut X1
Combination 2	Cut C3, Cut G, Cut Z
Combination 3	Cut C4, Cut G, Cut X1
Combination 4	Cut P, Cut X1

Table 5.12: Average performances of various cut combinations on Formulation V1 for 22 instances of Dataset A

Cut Added	No. of Non-zero Coefficients	Opt Gap for Bound I	Opt Gap for Bound II*	No. of Nodes	Average Time (sec)
N/A	9,643	74.57%	26.35%	42,529	71.7
Combination 1	11,738	36.95%	22.26%	**19,271**	66.3
Combination 2	10,849	**36.35%**	**21.05%**	22,563	**49.6**
Combination 3	12,225	36.95%	22.08%	33,062	90.1
Combination 4	11,937	37.07%	21.88%	27,543	81.1

* Problem 22 is excluded for this column since it was solved at the root node itself.

Table 5.13: Performances of various cut combinations for Problem No. 23 of Dataset A

Cut Added	No. of Non-zero Coefficients	Opt Gap for Bound I	Opt Gap for Bound II	No. of Nodes	Average Time (sec)
N/A	8,788	78.50%	60.50%	41,230,428	56,414.4
Combination 1	10,884	65.42%	54.56%	**13,034,931**	15,323.3
Combination 2	10,004	**63.93%**	**51.59%**	*>28,596,200	*>32,622.3
Combination 3	11,364	65.42%	54.57%	13,760,222	**15,256.5**
Combination 4	11,076	65.46%	54.53%	23,011,362	31,527.4

* Solver was stopped due to memory exhaustion.

Furthermore, we applied Cut Combination 1 to Formulations V2, V3 and V4 as well. The results obtained are summarized in Table 5.14 and Table 5.15. Note that, comparing these results with the results in Table 5.6 and Table 5.7, the addition of Combination 1 significantly tightened the LP relaxation and reduced the numbers of nodes for all four problem formulations. Formulation V4 with Cut Combination 1 is the most efficient configuration on average. Formulation V3 with Combination 1 required the least CPU time for the 22 easier problem instances of Dataset A.

Table 5.14: Average performances of four different formulations along with Combination 1 for 22 instances of Dataset A

Configuration	No. of Non-zero Coefficients	Opt Gap at Bound I	Opt Gap at Bound II*	No. of Nodes	Average Time (sec)
Formulation V1 + Combination 1	11,738	36.95%	22.26%	**19,271**	66.3
Formulation V2 + Combination 1	19,069	**31.56%**	**19.42%**	24,116	223.2
Formulation V3 + Combination 1	10,430	40.35%	22.33%	23,132	**58.6**
Formulation V4 + Combination 1	10,261	38.25%	21.77%	26,670	70.0

* Problem 22 is excluded for this column.

Table 5.15: Performances of four different formulations along with Combination 1 for Problem No. 23 of Dataset A

Configuration	No. of Non-zero Coefficients	Opt Gap at Bound I	Opt Gap at Bound II	No. of Nodes	Average Time (sec)
Formulation V1 + Combination 1	10,884	65.42%	54.56%	13,034,931	15,323.3
Formulation V2 + Combination 1	18,573	**64.54%**	**51.75%**	33,718,150	109,653.4
Formulation V3 + Combination 1	9,996	67.64%	53.45%	13,501,309	17,376.9
Formulation V4 + Combination 1	9,864	67.64%	52.82%	**9,695,521**	**9,021.7**

- **Larger Problem Instances**

Next, we applied various model configurations to Dataset B, which consists of more difficult problem instances. To make the computations more manageable, we solved these problem instances to a specified optimality gap of 10%. Based on the results of a preliminary experimentation, Formulation V1 with Cut Combination 1 yielded the best performance. Hence, using this model configuration, we recorded the corresponding CPU time, t^* seconds, say, for each instance in Dataset B, and then used a corresponding time limit of max $\{60, 2t^*\}$ seconds for testing other model configurations in order to provide some comparative results.

In order to evaluate the efficacy of adding valid inequalities, we solved the 10 problem instances in Dataset B using Formulation V1, with and without Cut Combination 1. Since the optimal objective values were not always available, the relative optimality gap was calculated by regarding the best objective value obtained as the true optimal value. The results are summarized in Table 5.16. An infeasible problem instance (Problem No. 3) has been excluded from this comparison. Note that the addition of Cut Combination 1 leads to a significant reduction in CPU time. This observation is further illustrated in Figure 5.19,

where an arrow denotes that the solver terminated prematurely before an optimal gap of 10% was achieved.

Table 5.16: Average performances for nine problem instances of Dataset B with and without Cut Combination 1

Configuration	No. of Non-zero Coefficients	Opt Gap for Bound I	Opt Gap for Bound II	No. of Nodes	Average Time (sec)
Formulation V1	48,329	75.44%	36.22%	*>316,904	*>9,342.9
Formulation V1 +Cut Combination 1	58,946	47.36%	31.64%	160,177	6,693.9

* On Problem 7, the solver stopped prematurely due to memory exhaustion.

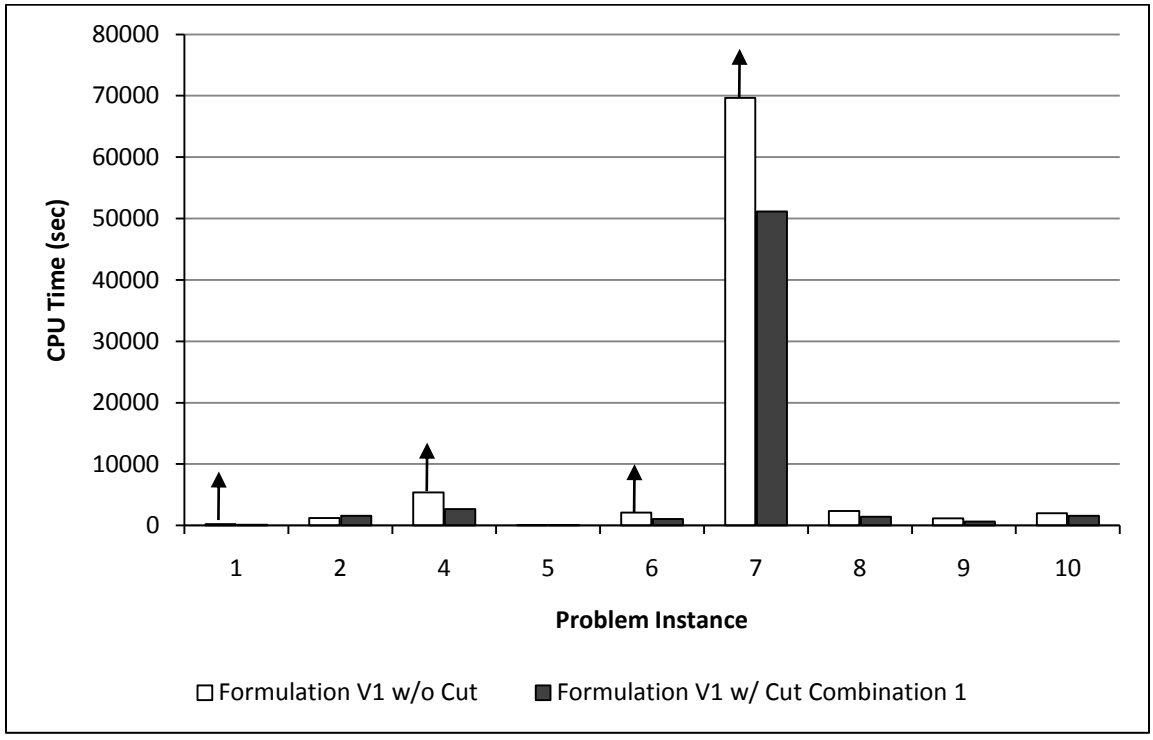


Figure 5.19: CPU time of problem instances in Dataset B

For comparative purposes, we also solved the problem instances of Dataset B with Formulations V2, V3, and V4 along with Cut Combination 1. The results are summarized in Table 5.17. Note that although Formulation V2 yielded tighter LP relaxations than that for Formulation V1, it is much larger in size, and therefore, required a longer average CPU time. Formulations V3 and V4 are more compact in model sizes than the other two, but they yield inferior values of Bound I and consume a longer average CPU time than that required by Formulation V1. To compare the CPU times in more detail, we have depicted the results for the individual nine instances in Figure 5.20. Note that the superiority of Formulation V1 (with Cut Combination 1) is more prominent for the harder problem instances. Formulation V3 is very competitive for easier problem instances.

Table 5.17: Average performances of three different formulations for nine problem instances of Dataset B.

Configuration	No. of Non-zero Coefficients	Opt Gap at Bound I	Opt Gap at Bound II	No. of Nodes	Average Time (sec)
Formulation V1 +Cut Combination 1	58,946	47.36%	31.64%	160,177	**6,693.9**
Formulation V2 +Cut Combination 1	85,057	**44.82%**	**30.54%**	**131,813**	12,016.6
Formulation V3 +Cut Combination 1	52,140	47.59%	31.65%	*>250,705	*>7,622.1
Formulation V4 + Combination 1	**51,390**	47.59%	31.62%	*>384,100	*>11,105.8

* On Problem 7, the solver stopped prematurely due to memory exhaustion.

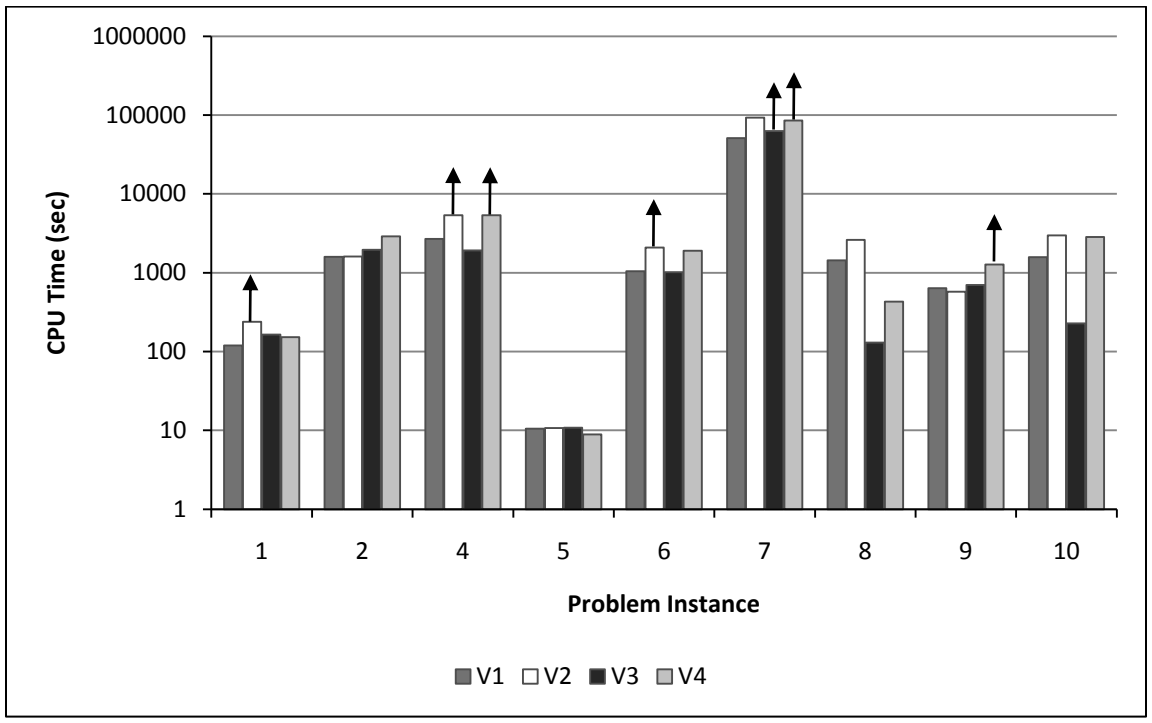


Figure 5.20: CPU time used by different formulations with Cut Combination 1, for the problems in Dataset B

5.8.3 Column Generation Method

To solve the PPMSP with the column generation method, we first tested the generation of initial feasible solutions by using the heuristic methods introduced in Section 5.6.2. For the Relaxation-Bounding (R-B) heuristic method, we used a target optimality gap of 10% and a time limit of 60 seconds in its relaxation step (i.e., Step 1) to reduce the CPU time. The results for the problem instances of Dataset A are summarized in Table 5.18, where the *heuristic gap* is defined as:

$$\frac{z_h - z^*}{z^*} \times 100\%, \tag{5.65}$$

where z_h is the objective value of the solution obtained by a heuristic method, and z^* is the optimal objective value. Three heuristic methods were implemented with Formulation V1

and Cut Combination 1. In addition, we considered the option of implementing the R-B heuristic method without adding Cut Combination 1. For a comparison, we also include in Table 5.18 the CPU time required to solve the problem to optimality by using Formulation V1 with Cut Combination 1, which is the best configuration found so far for a *Direct Solution*.

Table 5.18: Results of heuristic methods for 23 problem instances of Dataset A

Method	Maximum Heuristic Gap	Average Heuristic Gap	Maximum CPU Time (sec)	Average CPU Time (sec)
Period-wise heuristic	26.95%	6.01%	211.0	26.3
Level-wise heuristic	*56.44%	*23.91%	19.9	4.9
R-B Heuristic w/o Cut	**8.46%**	**2.28%**	73.5	19.9
R-B Heuristic w/Cut	22.91%	2.65%	82.9	14.1
Direct Solution	--	--	12,549.5	597.2

* Calculated based on 14 problem instances for which the heuristic obtained a feasible solution.

Note that the R-B heuristic method is the most effective of the three, as it yielded good initial solutions with relatively short CPU times. Furthermore, when the R-B heuristic method is used without any additional cuts, it yielded better solutions than those obtained with Cut Combination 1. Hence, for problem instances of Dataset A, we use this option to provide initial columns for the column generation method.

A comparative study of the decomposition schemes presented in Section 5.6.1 revealed the HMATSP2 scheme to give the most promising results. It required the least CPU time and number of iterations to converge, and yielded the best solutions. Next, we applied the column generation method with the HMATSP2 decomposition scheme to the problem instances of Dataset A. Subproblem heuristic was applied to reduce the CPU time. We consider two options for the starting strategy of the master program: either to start from a Phase-I linear program without initial columns, or to use the solution obtained from the R-B heuristic method as initial columns. In addition, we considered the inclusion of Cut

Combination 1 to the master program, as it tends to tighten the problem formulation. The results obtained are summarized in Table 5.19. Note that the CPU time reported includes both the time required to generate the columns and the time required to solve the final MIP. As the results suggest, the use of initial columns significantly reduces the CPU time, and also, improves the solution quality. Although the addition of Cut Combination 1 tends to require a higher number of iteration for the column generation procedure, it is very effective in reducing the overall CPU time and improving the solution quality. In conclusion, applying column generation method with initial columns and Cut Combination 1 is very efficient, as it reduces the average gap from an initial value of 2.28% (see Table 5.18) to 1.33% using only 38.3 seconds on average.

Table 5.19: Results of the column generation method using scheme HMATSP2 for 23 problem instances of Dataset A

Initial columns	Cuts Applied	Number of CG Iterations	Maximum Heuristic Gap	Average Heuristic Gap	CPU Time (sec)
N/A	N/A	9.3	52.45%	33.35%	1666.7
N/A	Combination 1	10.5	20.01%	6.05%	555.0
R-B heuristic	N/A	**8.4**	**5.38%**	1.51%	595.8
R-B heuristic	Combination 1	8.8	**5.38%**	**1.33%**	**38.3**

The convergence of our column generation method is quite fast in view of the number of iterations used. Nevertheless, we tested the use of BoxStep dual stabilization technique (Tran, Reinelt, and Bock, 2006) with Cut Combination 1 added to the master program. The results are summarized in Table 5.20, where initial columns were provided to both options. Note that the BoxStep method required more iterations and a longer CPU time due to its iterative adjustment of dual bounds. However, its final solution is slightly better than the default solution.

Table 5.20: Results of the column generation method using scheme HMATSP2 with and without dual stabilization technqiue for 23 problem instances of Dataset A

Dual Stabilization	Number of CG Iterations	Maximum Heuristic Gap	Average Heuristic Gap	Average CPU Time (sec)
N/A	**8.8**	5.38%	1.33%	**38.3**
BoxStep method	54.9	**4.20%**	**1.24%**	150.8

- **Larger Problem Instances**

Next, we applied the column generation method on Dataset B, which involves larger-sized problems. Since the optimal objective values were not always accessible, the heuristic gap was calculated by regarding the best lower bound obtained in Section 5.8.2 as the value of z^* in (5.65). As for Dataset A, we used the Relaxation-Bounding heuristic method to generate initial columns. To limit the CPU time required to solve the MIP in Step 2 of the R-B heuristic method, we applied a time limit of 1800 seconds and stopped the solver once a 1% optimality gap was reached. The results of R-B heuristic method are summarized in Table 5.21, along with the results of solving the problem by using Formulation V1 with Cut Combination 1 (Direct Solution). Note that, on average, the addition of Cut Combination 1 to the R-B heuristic method yielded better solutions than those obtained without adding cuts. We use this version on Dataset B to provide initial columns for the column generation method.

The results of our column generation method with the HMATSP2 decomposition scheme are presented in Table 5.22. At the end of column generation, a time limit of 3,600 seconds was applied for the solution of the master problem as an MIP. It is obvious that the addition of Cut Combination 1 improves the performance of the column generation method significantly. Consequently, the average relative difference between the solutions of column generation and those of direct solution is only 0.84%. If we further include the CPU time of the R-B heuristic method, the average CPU time of the column generation method is 490.7+2313.6=2804.3 seconds, which is significantly less than that of direct solution (i.e., 6693.9 seconds; see Table 5.16).

Table 5.21: Results of the Relaxation-Bounding heuristic method and direct solution by CPLEX for 9 problem instances of Dataset B

Method	Maximum Heuristic Gap	Average Heuristic Gap	Maximum CPU Time (sec)	Average CPU Time (sec)
R-B Heuristic w/o Cut	18.11%	14.32%	1,833.4	589.1
R-B Heuristic w/Cut	22.54%	11.66%	1,835.7	**490.7**
Direct Solution	11.11%	9.74%	51,155.2	6,693.9

Table 5.22: Results of the column generation method using scheme HMATSP2 with various valid inequalities for 9 problem instances of Dataset B

Cuts Applied	Number of CG Iterations	Maximum Heuristic Gap	Average Heuristic Gap	Average CPU Time (sec)
N/A	17.3	22.54%	11.60%	3168.7
Combination 1	**15.4**	**12.96%**	**10.09%**	**2313.6**

5.8.4 Local Search Methods

In Section 5.7, we have introduced two types of local search heuristic methods: (i) search by problem partitioning, and (ii) guided search. For the first type of local search, our experimental investigation has revealed that *search by product families* yields the best results. Therefore, we tested this strategy and the guided search method on problem instances of Dataset B. For both of these methods, we used the solution obtained from the Relaxation-Bounding Heuristic with Cut Combination 1 as the starting point. A time limit of 1800 seconds was imposed for the guided search method. The results are summarized in Table 5.23. Note that both of these methods improved upon the initial solutions obtained by using the R-B method (see Table 5.21); however, guided search generates better solutions. Compared with the column generation method (with Cut Combination 1), both of these method required less CPU time; but the solutions obtained are not as good as those by the column generation method (see Table 5.22).

Table 5.23: Results of two local search heuristic methods for 9 problem instances of Dataset B

Method	Maximum Heuristic Gap	Average Heuristic Gap	Maximum CPU Time (sec)	Average CPU Time (sec)
Search by Product Family	21.46%	11.34%	1741.7	230.7
Guided Search	22.54%	11.19%	1825.3	1620.4

Chapter 6

Summary, Conclusions and Future Research

This dissertation has been devoted to the development of novel approaches for some stochastic and deterministic scheduling problems. Our treatment of these scheduling problems has emphasized two important features: presence of stochastic job processing times and integration of scheduling with batching decisions for production on capacitated parallel machines. We have handled the first feature by addressing the following three aspects: (i) evaluation of a given schedule to determine both the expectation and variance of a performance measure; (ii) simultaneous minimization of expectation and variability of a performance measure through the use of conditional-value-at-risk; (iii) generation of scenarios that ascertain a lower bound for the problem by using the scenario-based mathematical programming approach. To address the second feature, we considered the primal pharmaceutical manufacturing scheduling problem, which comprises production in batches on parallel bays(machines) of limited capacity, sequence-dependent setup time/cost, and carryover setup (from one period to another).

In Chapter 2, we have considered the problem of determining both the expectation and variance of various performance measures of a given schedule when the job processing times are stochastic variables. Our method relies on the use of mixture models to

approximate a variety of distribution types. We have applied the Expectation-Maximization algorithm of Dempster et al. (1977) to derive mixture models of processing time distributions. Since the job sequencing decisions are known a priori, we utilized these mixture models of job processing times to calculate distributions of intermediate random variables in order to derive the expectation and variance of various scheduling performance measures. To make our method more computationally efficient, we have adapted a mixture reduction method to control the number of mixture components used at the intermediate steps. We have applied our method to two different scheduling problems: the job shop makespan minimization problem and the single machine total weighted tardiness scheduling problem. We compared the performance of our method with that of the Monte-Carlo method. The results show the efficacy of our mixture approximation method. Our method generates fairly accurate results while requiring significantly less CPU times. The proposed method offers a good compromise between the Monte Carlo method, which requires extensive effort, and the use of simple normal approximation, which produces lower-quality results.

In Chapter 3, we have introduced and demonstrated, for the first time in the literature, the use of conditional-value-at-risk (CVaR) as a criterion for stochastic scheduling problems in order to obtain risk-averse solutions. This criterion has the tendency of minimizing both the expectation and variance of a performance measure simultaneously, which is an attractive feature in the scheduling area as most of the literature in this area considers the expectation and variance of a performance measure separately. Also, the CVaR has an added advantage of maintaining a linear objective function. We have developed a scenario-based mixed integer programming formulation to minimize CVaR for the general scheduling problem involving various performance measures, and employed a decomposition-based approach for its solution. Furthermore, valid inequalities were incorporated to strengthen the relaxed master problem of this decomposition scheme. We have demonstrated the use of the proposed approach on the single machine total weighted tardiness scheduling problem. Our computational investigation has revealed the efficacy of

the proposed decomposition approach and the effectiveness of using the CVaR as an optimization criterion for scheduling problems. Besides providing an exact approach to solve our stochastic scheduling problem, we have also developed an efficient heuristic method to enable the use of CVaR for large-sized problems. To that end, we modified the Dynasearch method of Grosso et al. (2004) to minimize CVaR for a stochastic scheduling problem. Furthermore, we have extended the application of CVaR to a parallel-machine total weighted tardiness problem. The use of CVaR appears to be quite promising for simultaneously controlling both the expected value and variability of a performance measure in a stochastic scheduling environment.

Our focus in Chapter 4, has been to determinate a lower bound for stochastic scheduling problems. Due to the long CPU time required to solve a scenario-based formulation, the method of Sample Average Approximation tends to be quite time-consuming. The new method that we have developed for scenario generation is computationally competitive, and it assures attainment of an exact lower bound. Our approach is based on discretization of random parameter distributions of job processing times. We have used the idea of Recursive Stratified Sampling to partition the probability space to enable the conditional expectations in each region to yield scenario-wise parameter values. These scenarios are, then, used to formulate a two-stage stochastic program, which yields a lower bound for the original stochastic problem. We have provided theoretical basis of our bounding approach for both the expectation and CVaR objectives. Our discrete bounding method generates exact lower bounds, as against the probabilistic bounds generated by Sample Average Approximation. We have also presented results of our numerical experimentation that demonstrate superiority of our method over the Sample Average Approximation method in terms of the bound value obtained and the CPU time required.

In Chapter 5, we have addressed an integrated problem of batching and scheduling of jobs on parallel capacitated machines that is encountered in the primary manufacturing sector of a pharmaceutical supply chain. We have developed a comprehensive mathematical programming model that accommodates various realistic features of this problem. These

features include batch production, sequence-dependent setup time/cost, and carryover setups (from one period to another). We have further derived several valid inequalities that exploit the embedded subproblem structure. We have also considered an alternative formulation (termed the Plant Location model) based on the lot-sizing perspective of the problem. Noting the resemblance of the campaign sequencing subproblem to the high multiplicity asymmetric traveling salesman problem (HMATSP), we have adapted various ideas from the HMATSP to enforce the connectivity of the sequencing graph. Due to the complexity of this problem, we also explored the possibility of applying column generation technique for its solution. Various schemes of problem decomposition were considered, along with the use of dual stabilization technique to improve convergence of the column generation procedure. We also employed an advance-start strategy to further enhance the performance of the column generation method. Our comprehensive computational investigation on a testbed of data that mimics real-life problem instances obtained from Boehringer-Ingelheim's facility in Petersburg, Virginia reveals the effectiveness of using the proposed column generation method on real-life size problems.

There are several directions that can be followed for future research.

1. Our method of mixture approximation is an alternative method to compute both the expectation and variance of a schedule. Even though we have demonstrated its application on two problems, yet its application to other scheduling problems and performance measures seems quite promising. The key question is to seek a compromise between the number of components and the accuracy desired.
2. We have successfully demonstrated, for the first time in the literature, the use of CVaR to simultaneously minimize both the expectation and variance of a performance measure. Until now, these measures have been optimized separately. It seems very promising to further investigate its use in the scheduling area.
3. We have provided several useful insights regarding the integrated scheduling and lot-sizing problem. It seems worthwhile to develop other methods, besides the column generation method, for the solution of this difficult problem.

Appendix A

Approximating Expectation using Kernel Density Estimation

Consider k independent samples of random variable $X \sim f_X(x)$:

$$\{X_i : i = 1, \dots, k\}$$

The PDF of its Gaussian kernel density estimation, $\tilde{X}$, can be written as:

$$f_{\tilde{X}}(x) = \sum_{i=1}^{k} \frac{1}{k\sigma} \varphi\left(\frac{x - X_i}{\sigma}\right).$$

Suppose that $g(x)$ is performance measure of interests, and we use $\tilde{X}$ instead of X to determine its value. The corresponding error is

$$E\left[g\left(\tilde{X}\right)\right] - E[g(X)] = \int_{-\infty}^{+\infty} \left(f_{\tilde{X}}(x) - f_X(x)\right) g(x) \mathrm{d}x \,.$$

Since the PDF of $\tilde{X}$ depends on the random sample values $\{X_i\}$, the resulting error is also a random variable whose expectation can be written as follows:

$$E_{\{X_i\}}\left[E\left[g\left(\tilde{X}\right)\right] - E[g(X)]\right] = \int_{-\infty}^{+\infty} \left(E_{\{X_i\}}[f_{\tilde{X}}(x)] - f_X(x)\right) g(x) \mathrm{d}x \,.$$

We have

$$\begin{aligned} & E_{\{X_i\}}[f_{\tilde{X}}(x)] \\ &= E_{\{X_i\}}\left[\sum_{i=1}^{k} \frac{1}{k\sigma} \varphi\left(\frac{x - X_i}{\sigma}\right)\right] \\ &= E\left[\frac{1}{\sigma} \varphi\left(\frac{x - X_i}{\sigma}\right)\right] = \frac{1}{\sigma} \int_{-\infty}^{+\infty} \varphi\left(\frac{x - y}{\sigma}\right) f_X(y) \mathrm{d}y. \end{aligned}$$

Applying transformation $t = \frac{x-y}{\sigma}$ and $y = x - \sigma t$ to the above equation, we further have

$$E_{\{X_i\}}[f_{\tilde{X}}(x)] = \int_{-\infty}^{+\infty} \varphi(t) f_X(x - \sigma t) \mathrm{d}t.$$

Hence,

$$\begin{aligned} & E_{\{X_i\}}\left[E\left[g\left(\tilde{X}\right)\right] - E[g(X)]\right] \\ &= \int_{-\infty}^{+\infty} \left[\int_{-\infty}^{+\infty} \varphi(t) f_X(x - \sigma t) \mathrm{d}t - f_X(x)\right] g(x) \mathrm{d}x \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \varphi(t) [f_X(x - \sigma t) - f_X(x)] \mathrm{d}t \, g(x) \mathrm{d}x \end{aligned} \tag{A.1}$$

Assuming the existence of derivatives $f_X'(x)$, $f_X''(x)$, and $f_X^{(3)}(x)$, we apply Taylor's theorem as follows:

$$f_X(x - \sigma t) - f_X(x) = -\sigma t \cdot f_X'(x) + \frac{\sigma^2 t^2}{2!} f_X''(x) - \frac{\sigma^3 t^3}{3!} f_X^{(3)}(\xi),$$

where $\xi \in [x - \sigma t, x]$.

Since

$$\int_{-\infty}^{+\infty} t \cdot \varphi(t) \mathrm{d}t = 0, \qquad \int_{-\infty}^{+\infty} t^2 \cdot \varphi(t) \mathrm{d}t = 1,$$

we further have:

$$\begin{aligned} & \int_{-\infty}^{+\infty} \varphi(t) [f_X(x - \sigma t) - f_X(x)] \mathrm{d}t \\ &= \int_{-\infty}^{+\infty} \varphi(t) \left[-\sigma t \cdot f_X'(x) + \frac{\sigma^2 t^2}{2!} f_X''(x) - \frac{\sigma^3 t^3}{3!} f_X^{(3)}(\xi)\right] \mathrm{d}t \\ &= \frac{\sigma^2}{2!} f_X''(x) - \frac{\sigma^3}{3!} L(x), \end{aligned}$$

where $L(x) \equiv \int_{-\infty}^{+\infty} t^3 \varphi(t) f_X^{(3)}(\xi) \mathrm{d}t$.

Consequently,

$$\begin{aligned} & E_{\{X_i\}}\left[E\left[g\left(\tilde{X}\right)\right] - E[g(X)]\right] \\ &= \frac{\sigma^2}{2!} \int_{-\infty}^{+\infty} f_X''(x) g(x) \mathrm{d}x - \frac{\sigma^3}{3!} \int_{-\infty}^{+\infty} L(x) g(x) \mathrm{d}x. \end{aligned}$$

Therefore, convergence of the expected error with diminishing values of σ depends on the finiteness of the terms $\int_{-\infty}^{+\infty} f_X''(x)g(x)\mathrm{d}x$ and $\int_{-\infty}^{+\infty} L(x)g(x)\mathrm{d}x$. In case that the derivatives of $f_X(x)$ do not exist, the convergence condition need to be derived separately.

Suppose that Y is a random variable, which follows the Gaussian kernel distribution $\varphi(.)$, and that it is independent of X. We have

$$Z \equiv X + \sigma Y \sim f_{Z(z)} = \int_{-\infty}^{+\infty} \varphi(t) f_X(z - \sigma t)\mathrm{d}t.$$

In view of Equation (A.1), we can re-write the expected error as follows:

$$E_{\{X_i\}}\left[E\left[g\left(\tilde{X}\right)\right] - E[g(X)]\right] = E[g(X + \sigma Y) - g(X)].$$

Hence, its convergence to zero is intuitive with diminishing values of σ.

Appendix B

Simplification of Problem SP

Define two non-negative second stage variables $z^{+}_{\boldsymbol{\theta}(\omega)}$ and $z^{-}_{\boldsymbol{\theta}(\omega)}$, such that

$$z^{+}_{\boldsymbol{\theta}(\omega)} - z^{-}_{\boldsymbol{\theta}(\omega)} = \mathbf{q}\big(\boldsymbol{\theta}(\omega)\big)^{T}\mathbf{x}, \qquad \forall \boldsymbol{\theta}(\omega) \in \Theta.$$

The problem can be re-written as:

$$\begin{aligned}
\textbf{SP}: \quad \text{Minimize} \quad & \mathcal{H}[\mathbf{h}^{\boldsymbol{T}}\mathbf{y}_{\boldsymbol{\theta}} + z^{+}_{\boldsymbol{\theta}} - z^{-}_{\boldsymbol{\theta}}] \\
\text{subject to:} \quad & \mathbf{x} \in \Gamma, \mathbf{A}\mathbf{x} = \mathbf{b} \\
& \mathbf{E}\big(\boldsymbol{\theta}(\omega)\big)\mathbf{x} + \mathbf{R}\mathbf{y}_{\boldsymbol{\theta}(\omega)} = \mathbf{g}\big(\boldsymbol{\theta}(\omega)\big), && \forall \boldsymbol{\theta}(\omega) \in \Theta \\
& \mathbf{c}\big(\boldsymbol{\theta}(\omega)\big)^{T}\mathbf{x} - z^{+}_{\boldsymbol{\theta}(\omega)} + z^{-}_{\boldsymbol{\theta}(\omega)} = 0, && \forall \boldsymbol{\theta}(\omega) \in \Theta \\
& \mathbf{y}_{\boldsymbol{\theta}(\omega)} \in \mathbb{R}^{n_2}_{+}, && \forall \boldsymbol{\theta}(\omega) \in \Theta \\
& z^{+}_{\boldsymbol{\theta}(\omega)}, z^{-}_{\boldsymbol{\theta}(\omega)} \geq 0, && \forall \boldsymbol{\theta}(\omega) \in \Theta.
\end{aligned}$$

Let $\hat{\mathbf{h}} = \begin{bmatrix} \mathbf{h} \\ 1 \\ -1 \end{bmatrix}, \hat{\mathbf{y}}_{\boldsymbol{\theta}} = \begin{bmatrix} \mathbf{y}_{\boldsymbol{\theta}} \\ z^{+}_{\boldsymbol{\theta}} \\ z^{-}_{\boldsymbol{\theta}} \end{bmatrix}, \hat{\mathbf{E}}(.) = \begin{bmatrix} \mathbf{E}(.) \\ \mathbf{c}(.)^{T} \end{bmatrix}, \hat{\mathbf{R}} = \begin{bmatrix} \mathbf{R} & 0 & 0 \\ 0 & -1 & 1 \end{bmatrix}, \hat{\mathbf{g}}(.) = \begin{bmatrix} \mathbf{g}(.) \\ 0 \end{bmatrix},$

$\hat{n}_2 = n_2 + 2.$

We have the following simplified form:

$$\begin{aligned}
\textbf{SP}: \quad \text{Minimize} \quad & \mathcal{H}\big[\hat{\mathbf{h}}^{T}\hat{\mathbf{y}}_{\boldsymbol{\theta}}\big] \\
\text{subject to:} \quad & \mathbf{x} \in \Gamma, \mathbf{A}\mathbf{x} = \mathbf{b} \\
& \hat{\mathbf{E}}\big(\boldsymbol{\theta}(\omega)\big)\mathbf{x} + \hat{\mathbf{R}}\hat{\mathbf{y}}_{\boldsymbol{\theta}(\omega)} = \hat{\mathbf{g}}\big(\boldsymbol{\theta}(\omega)\big), && \forall \boldsymbol{\theta}(\omega) \in \Theta \\
& \hat{\mathbf{y}}_{\boldsymbol{\theta}(\omega)} \in \mathbb{R}^{\hat{n}_2}_{+}, && \forall \boldsymbol{\theta}(\omega) \in \Theta.
\end{aligned}$$

Appendix C
Additional Decomposition Schemes for Column Generation

C.1 HMATSP2 Decomposition Scheme

Sets:

$\Lambda^{k,t}$: Index set of the columns corresponding to bay k and period t, $\forall k \in M, t \in T$.

Parameters:

$(\_)_\iota$: Value of subproblem variable "_" fixed in the ι^{th} column of bay k and period t, $\forall k \in M, t \in T, \iota \in \Lambda^{k,t}$. Note that indices k and t are implicitly indicated by the enclosed variable "_".

Variables:

$Q_\iota^{k,t}$: Binary variable indicating the inclusion/exclusion of the ι^{th} column of bay k and period t, $\forall k \in M, t \in T, \iota \in \Lambda^{k,t}$, and relaxed to take continuous values.

$v^{k,t}$: Dual variable associated with the Convexity Constraint (C.1), $\forall k \in M, t \in T$.

$\mu^{k,t}$: Dual variable associated with the Time Capacity Constraint (C.2), $\mu^{k,t} \leq 0$, $\forall k \in M$, $t \in T$.

χ_g^{k,t_2}: Dual variable associated with Constraint (C.3), $\forall k \in M, t_2 \in T, g \in F^k$.

ψ_f^{k,t_1}: Dual variable associated with Constraint (C.4), $\psi_f^{k,t_1} \geq 0$, $\forall k \in M, t_1 \in T, f \in F^k$.

$\xi_f^{k,t}$: Dual variable associated with Constraint (C.5), $\xi_f^{k,t} \leq 0$, $\forall k \in M, t \in T, f \in F^k$.

$\nu_f^{k,t}$: Dual variable associated with Constraint (C.6), $\nu_f^{k,t} \geq 0$, $\forall k \in M, t \in T, f \in F^k$.

$\phi_f^{k,t}$: Dual variable associated with Constraint (C.7), $\phi_f^{k,t} \leq 0$, $\forall k \in M, t \in T, f \in F^k$.

The problem decomposition is as follows:

HMATSP2-Master:

$$\text{Minimize} \sum_{t\in T}\sum_{f\in F}\sum_{i\in N_f} h_{f,i} I_{f,i}^t + \sigma \sum_{k\in M}\sum_{f\neq g\in F^k} s_{f,g}^k \left(\sum_{t\in T}\sum_{\iota\in\Lambda^{k,t}} Q_\iota^{k,t} \left(X_{f,g}^{k,t}\right)_\iota + \sum_{t_1<t_2\in\hat{T}} \gamma_{f,g}^{k,t_1,t_2} \right)$$

subject to: Constraints (5.1)-(5.3), (5.5), (5.6), (5.11), (5.12), (5.18)

$$\sum_{\iota\in\Lambda^{k,t}} Q_\iota^{k,t} = 1, \qquad \forall k \in M, t \in T \tag{C.1}$$

$$\sum_{f\in F^k}\sum_{i\in N_f\cap N^k} p_{f,i}^k W_{f,i}^{k,t} + \sum_{f\neq g\in F^k} s_{f,g}^k \sum_{\iota\in\Lambda^{k,t}} Q_\iota^{k,t}\left(X_{f,g}^{k,t}\right)_\iota + \alpha^{k,t} + \beta^{k,t} \leq c^{k,t}, \qquad \forall k \in M, t \in T \tag{C.2}$$

$$\sum_{\iota\in\Lambda^{k,t_2}} Q_\iota^{k,t_2}\left(X_{0,g}^{k,t_2}\right)_\iota = \sum_{f\in F^k}\sum_{\substack{t_1\in\hat{T};\\ t_1<t_2}} \gamma_{f,g}^{k,t_1,t_2}, \qquad \forall k \in M, t_2 \in T, g \in F^k \tag{C.3}$$

$$\sum_{\iota\in\Lambda^{k,t_1}} Q_\iota^{k,t_1}\left(X_{f,0}^{k,t_1}\right)_\iota \geq \sum_{g\in F^k}\sum_{\substack{t_2\in T,\\ t_2>t_1}} \gamma_{f,g}^{k,t_1,t_2}, \qquad \forall k \in M, t_1 \in T, f \in F^k \tag{C.4}$$

$$\sum_{i\in N_f\cap N^k} W_{f,i}^{k,t} \leq l_f^{k,t} \sum_{\iota\in\Lambda^{k,t}} Q_\iota^{k,t}\left(R_f^{k,t}\right)_\iota, \qquad \forall k \in M, t \in T, f \in F^k \tag{C.5}$$

$$\sum_{i\in N_f\cap N^k} W_{f,i}^{k,t} \geq \sum_{\iota\in\Lambda^{k,t}} Q_\iota^{k,t}\left(R_f^{k,t}\right)_\iota, \qquad \forall k \in M, t \in T, f \in F^k \tag{C.6}$$

$$\sum_{i\in N_f\cap N^k} W_{f,i}^{k,t} \le m_f^{k,t} \sum_{\iota\in\Lambda^{k,t}} Q_\iota^{k,t}\left(Z_f^{k,t}\right)_\iota,$$

$$\forall k\in M, t\in T, f\in F^k \tag{C.7}$$

$$Q_\iota^{k,t} \ge 0, \qquad \forall k\in M, t\in T, \iota\in\Lambda^{k,t}$$

$$I_{f,i}^t \ge 0, G_{f,i}^t \ge 0, \qquad \forall t\in T, f\in F, i\in N_f$$

$$P_{f,i}^{k,t} \ge 0, W_{f,i}^{k,t}\in\mathbb{Z}^+, \qquad \forall t\in T, f\in F, i\in N_f, k\in M^i$$

$$\gamma_{f,g}^{k,t_1,t_2}\in\{0,1\}, \qquad \forall k\in M, t_1<t_2\in\hat{T}, f,g\in F^k$$

$$\alpha^{k,t}\ge 0, \beta^{k,t}\ge 0, \qquad \forall k\in M, t\in T.$$

HMATSP2-Subproblem, $\forall k\in M, t\in T$:

Minimize $\sum_{f\ne g\in F^k} s_{f,g}^k\left(\sigma-\mu^{k,t}\right)X_{f,g}^{k,t} - \upsilon^{k,t}$

$$+\sum_{f\in F^k}\left(\xi_f^{k,t} l_f^{k,t} R_f^{k,t} + \nu_f^{k,t} R_f^{k,t} + \phi_f^{k,t} m_f^{k,t} Z_f^{k,t} - \chi_f^{k,t} X_{0,f}^{k,t} - \psi_f^{k,t} X_{f,0}^{k,t}\right)$$

subject to: Constraints (5.9), (5.10), (5.16), (5.18), (5.19)-(5.22)

$$\sum_{\forall g\in F^k} X_{0,g}^{k,\mathrm{t}} \le 1 \tag{C.8}$$

$$\sum_{\forall f\in F^k} X_{f,0}^{k,\mathrm{t}} \le 1 \tag{C.9}$$

$$R_f^{k,t} \le m_f^{k,t} Z_f^{k,t}, \qquad \forall f\in F^k \tag{C.10}$$

$$\sum_{f\in F^k} \min_{i\in N_f\cap N^k}\left\{p_{f,i}^k\right\} R_f^{k,t} + \sum_{f\ne g\in F^k} s_{f,g}^k X_{f,g}^{k,t} \le c^{k,t} \tag{C.11}$$

$$R_f^{k,t}\in\mathbb{Z}^+, Z_f^{k,t}\in\{0,1\}, \qquad \forall f\in F^k$$

$$X_{f,g}^{k,t}\in\mathbb{Z}^+, \qquad \forall f\ne g\in\hat{F}^k.$$

Note that Constraints (C.8) and (C.9) are aggregated versions of Constraints (5.7) and (5.8), respectively. Constraint (C.10) is based on Constraint (5.24), which is induced by (5.14) and (5.15), and dictates that $Z_f^{k,t}=1$ if $R_f^{k,t}>0$. Constraint (C.11) is a relaxed version of the Time Capacity Constraint (5.4). If the optimal objective value of the subproblem is negative,

the column index ι is increased by 1 and the corresponding values of $R_f^{k,t}$, $Z_f^{k,t}$ and $X_{f,g}^{k,t}$ variables are recorded as coefficients of the new column for the master problem.

C.2 B&S Decomposition Scheme

We present the formulations for this decomposition scheme as follows:

Sets:

Λ^k: Index set of the columns corresponding to bay k, $\forall k \in M$.

Parameters:

$(\_)_\iota$: Value of subproblem variable "$\_$" fixed in the ι^{th} column of bay k, $\forall k \in M, \iota \in \Lambda^k$. Note that index k is implicitly indicated by the enclosed variable "$\_$".

Variables:

Q_ι^k: Binary variable indicating the inclusion/exclusion of the ι^{th} column of bay k, $\forall k \in M$, $\iota \in \Lambda^k$, and relaxed to take continuous values.

v^k: Dual variable associated with the Convexity Constraint (C.12), $\forall k \in M$.

$\vartheta_{f,i}^{k,t}$: Dual variable associated with the Constraint (C.13), $\vartheta_{f,i}^{k,t} \leq 0$, $\forall t \in T, f \in F, i \in N_f$, $k \in M^i$.

The problem decomposition is as follows:

B&S-Master:

$$\text{Minimize} \sum_{t\in T}\sum_{f\in F}\sum_{i\in N_f} h_{f,i} I_{f,i}^t + \sigma \sum_{k\in M}\sum_{f\neq g\in F^k} s_{f,g}^k \sum_{\iota\in\Lambda^k} Q_\iota^k \left[\sum_{t\in T}\left(X_{f,g}^{k,t}\right)_\iota + \sum_{t_1<t_2\in\hat{T}}\left(\gamma_{f,g}^{k,t_1,t_2}\right)_\iota\right]$$

subject to: Constraints (5.1), (5.2)

$$\sum_{\iota\in\Lambda^k} Q_\iota^k = 1, \qquad \forall t \in T, k \in M \tag{C.12}$$

$$P_{f,i}^{k,t} \leq q_{f,i}^k \sum_{\iota\in\Lambda^k} Q_\iota^k \left(W_{f,i}^{k,t}\right)_\iota, \qquad \forall t \in T, f \in F, i \in N_f, k \in M^i \tag{C.13}$$

$$Q_\iota^k \geq 0, \qquad \forall k \in M, \iota \in \Lambda^k$$
$$I_{f,i}^t \geq 0, G_{f,i}^t \geq 0, \qquad \forall t \in T, f \in F, i \in N_f$$
$$P_{f,i}^{k,t} \geq 0, \qquad \forall t \in T, f \in F, i \in N_f, k \in M^i.$$

B&S-Subproblem, $\forall k \in M$:

Minimize $\sum_{f \neq g \in F^k} \sigma s_{f,g}^k \left(\sum_{t \in T} X_{f,g}^{k,t} + \sum_{t_1 < t_2 \in \hat{T}} \gamma_{f,g}^{k,t_1,t_2} \right) - \upsilon^k + \sum_{f \in F^k} \sum_{i \in N_f \cap N^k} q_{f,i}^k \sum_{t \in T} \vartheta_{f,i}^{k,t} W_{f,i}^{k,t}$

subject to: Constraints(5.4)-(5.22)

$$W_{f,i}^{k,t} \in \mathbb{Z}^+, \qquad \forall t \in T, f \in F^k, i \in N_f$$
$$R_f^{k,t} \in \mathbb{Z}^+, Z_f^{k,t} \in \{0,1\}, \qquad \forall t \in T, f \in F^k$$
$$X_{f,g}^{k,t} \in \mathbb{Z}^+, \qquad \forall t \in T, f \neq g \in \hat{F}^k$$
$$\gamma_{f,g}^{k,t_1,t_2} \in \{0,1\}, \qquad \forall t_1, t_2 \in \hat{T}, f, g \in F^k; t_1 < t_2$$
$$\alpha^{k,t} \geq 0, \beta^{k,t} \geq 0 \qquad \forall t \in T$$
$$\lambda_{f,g}^{k,t,u} \geq 0, \qquad \forall t \in T, u, g \in F^k, f \in \hat{F}^k; f \neq g.$$

C.3 Knapsack Decomposition Scheme

We present the formulations for this decomposition scheme as follows:

Sets:

$\Lambda^{k,t}$: Index set of the columns corresponding to bay k and period t, $\forall k \in M, t \in T$.

Parameters:

$(\_)_\iota$: Value of subproblem variable "$\_$" fixed in the ι^{th} column of bay k and period t, $\forall k \in M, t \in T, \iota \in \Lambda^{k,t}$. Note that indices k and t are implicitly indicated by the enclosed variable "$\_$".

Variables:

$Q_\iota^{k,t}$: Binary variable indicating the inclusion/exclusion of the ι^{th} column of bay k and

period t, $\forall k \in M, t \in T, \iota \in \Lambda^{k,t}$, and relaxed to take continuous values.

$v^{k,t}$: Dual variable associated with the Convexity Constraint (C.14), $\forall k \in M, t \in T$.

$\vartheta_{f,i}^{k,t}$: Dual variable associated with the Constraint (C.15), $\vartheta_{f,i}^{k,t} \leq 0$, $\forall t \in T, f \in F, i \in N_f$, $k \in M^i$.

$\mu^{k,t}$: Dual variable associated with the Time Capacity Constraint (C.16), $\mu^{k,t} \leq 0$, $\forall k \in M, t \in T$.

$\xi_f^{k,t}$: Dual variable associated with Constraint (C.17), $\xi_f^{k,t} \leq 0$, $\forall k \in M, t \in T, f \in F^k$.

$\nu_f^{k,t}$: Dual variable associated with Constraint (C.18), $\nu_f^{k,t} \geq 0$, $\forall k \in M, t \in T, f \in F^k$.

$\phi_f^{k,t}$: Dual variable associated with Constraint (C.19), $\phi_f^{k,t} \leq 0$, $\forall k \in M, t \in T, f \in F^k$.

The problem decomposition is as follows:

Knapsack-Master:

$$\text{Minimize} \sum_{t \in T} \sum_{f \in F} \sum_{i \in N_f} h_{f,i} I_{f,i}^t + \sigma \sum_{k \in M} \sum_{f \neq g \in F^k} s_{f,g}^k \left(\sum_{t \in T} X_{f,g}^{k,t} + \sum_{t_1 < t_2 \in \hat{T}} \gamma_{f,g}^{k,t_1,t_2} \right)$$

subject to: Constraints (5.1), (5.2), (5.5)-(5.12), (5.16)-(5.22)

$$\sum_{\iota \in \Lambda^{k,t}} Q_\iota^{k,t} = 1, \qquad \forall k \in M, t \in T \tag{C.14}$$

$$P_{f,i}^{k,t} \leq q_{f,i}^k \sum_{\iota \in \Lambda^{k,t}} Q_\iota^{k,t} \left(W_{f,i}^{k,t}\right)_\iota, \qquad \forall t \in T, f \in F, i \in N_f, k \in M^i \tag{C.15}$$

$$\sum_{f \in F^k} \sum_{i \in N_f \cap N^k} p_{f,i}^k \sum_{\iota \in \Lambda^{k,t}} Q_\iota^{k,t} \left(W_{f,i}^{k,t}\right)_\iota + \sum_{f \neq g \in F^k} s_{f,g}^k X_{f,g}^{k,t} + \alpha^{k,t} + \beta^{k,t} \leq c^{k,t}, \qquad \forall t \in T, k \in M \tag{C.16}$$

$$\sum_{i \in N_f \cap N^k} \sum_{\iota \in \Lambda^{k,t}} Q_\iota^{k,t} \left(W_{f,i}^{k,t}\right)_\iota \leq l_f^{k,t} R_f^{k,t}, \qquad \forall k \in M, t \in T, f \in F^k \tag{C.17}$$

$$\sum_{i\in N_f\cap N^k}\sum_{\iota\in\Lambda^{k,t}} Q_\iota^{k,t}\left(W_{f,i}^{k,t}\right)_\iota \geq R_f^{k,t},$$

$$\forall k\in M, t\in T, f\in F^k \qquad \text{(C.18)}$$

$$\sum_{i\in N_f\cap N^k}\sum_{\iota\in\Lambda^{k,t}} Q_\iota^{k,t}\left(W_{f,i}^{k,t}\right)_\iota \leq m_f^{k,t} Z_f^{k,t},$$

$$\forall k\in M, t\in T, f\in F^k \qquad \text{(C.19)}$$

$$Q_\iota^{k,t}\geq 0, \qquad \forall k\in M, t\in T, \iota\in\Lambda^{k,t}$$

$$I_{f,i}^t \geq 0, G_{f,i}^t\geq 0, \qquad \forall t\in T, f\in F, i\in N_f$$

$$P_{f,i}^{k,t}\geq 0, \qquad \forall t\in T, f\in F, i\in N_f, k\in M^i$$

$$R_f^{k,t}\in\mathbb{Z}^+, Z_f^{k,t}\in\{0,1\}, \qquad \forall t\in T, f\in F^k$$

$$X_{f,g}^{k,t}\in\mathbb{Z}^+, \qquad \forall t\in T, f\neq g\in\hat{F}^k$$

$$\gamma_{f,g}^{k,t_1,t_2}\in\{0,1\}, \qquad \forall t_1,t_2\in\hat{T}, f,g\in F^k; t_1<t_2$$

$$\alpha^{k,t}\geq 0, \beta^{k,t}\geq 0 \qquad \forall k\in M, t\in T$$

$$\lambda_{f,g}^{k,t,u}\geq 0, \qquad \forall t\in T, u, g\in F^k, f\in\hat{F}^k; f\neq g.$$

Knapsack-Subproblem, $\forall k\in M, t\in T$:

Minimize $\sum_{f\in F^k}\sum_{i\in N_f\cap N^k}\left(q_{f,i}^k\vartheta_{f,i}^{k,t}-p_{f,i}^k\mu^{k,t}-\xi_f^{k,t}-\nu_f^{k,t}-\phi_f^{k,t}\right)W_{f,i}^{k,t}-\upsilon^k$

subject to:

$$\sum_{f\in F^k}\sum_{i\in N_f\cap N^k} p_{f,i}^k W_{f,i}^{k,t}\leq c^{k,t}$$

$$W_{f,i}^{k,t}\in\mathbb{Z}^+, \qquad \forall f\in F^k, i\in N_f\cap N^k$$

C.4 LSP Decomposition Scheme

We present the formulations for this decomposition scheme as follows:

Sets:

Λ_f: Index set of the columns corresponding to product family f, $\forall f \in F$.

Parameters:

$(\_)_\iota$: Value of subproblem variable "$\_$" fixed in the ι^{th} column of family f, $\forall f \in F$. Note that index f is implicitly indicated by the enclosed variable "$\_$".

Variables:

Q_ι^f: Binary variable indicating the inclusion/exclusion of the ι^{th} column of family f, $\forall f \in F$, and relaxed to take continuous values.

v^f: Dual variable associated with the Convexity Constraint (C.20), $\forall f \in F$.

$\mu^{k,t}$: Dual variable associated with the Time Capacity Constraint (C.21), $\mu^{k,t} \leq 0$, $\forall k \in M, t \in T$.

$\xi_f^{k,t}$: Dual variable associated with Constraint (C.22), $\xi_f^{k,t} \leq 0$, $\forall k \in M, t \in T, f \in F^k$.

$\nu_f^{k,t}$: Dual variable associated with Constraint (C.23), $\nu_f^{k,t} \geq 0$, $\forall k \in M, t \in T, f \in F^k$.

$\phi_f^{k,t}$: Dual variable associated with Constraint (C.24), $\phi_f^{k,t} \leq 0$, $\forall k \in M, t \in T$, $f \in F^k$.

The problem decomposition is as follows:

LSP-Master:

$$\text{Minimize} \sum_{t\in T}\sum_{f\in F}\sum_{i\in N_f} h_{f,i} \sum_{\iota\in\Lambda_f} Q_\iota^f \left(I_{f,i}^t\right)_\iota + \sigma \sum_{k\in M}\sum_{f\neq g\in F^k} s_{f,g}^k \left(\sum_{t\in T} X_{f,g}^{k,t} + \sum_{t_1<t_2\in\hat{T}} \gamma_{f,g}^{k,t_1,t_2}\right)$$

subject to: Constraints (5.5) -(5.12), (5.16) -(5.22)

$$\sum_{\iota\in\Lambda_f} Q_\iota^f = 1, \qquad \forall f \in F \qquad \text{(C.20)}$$

$$\sum_{f\in F^k}\sum_{i\in N_f\cap N^k} p_{f,i}^k \sum_{\iota\in\Lambda_f} Q_\iota^f \left(W_{f,i}^{k,t}\right)_\iota \leq c^{k,t}, \qquad \forall t \in T, k \in M \qquad \text{(C.21)}$$

$$\sum_{i\in N_f\cap N^k}\sum_{\iota\in\Lambda_f} Q_\iota^f\left(W_{f,i}^{k,t}\right)_\iota \le l_f^{k,t}R_f^{k,t}, \qquad \forall k\in M, t\in T, f\in F^k \tag{C.22}$$

$$\sum_{i\in N_f\cap N^k}\sum_{\iota\in\Lambda_f} Q_\iota^f\left(W_{f,i}^{k,t}\right)_\iota \ge R_f^{k,t}, \qquad \forall k\in M, t\in T, f\in F^k \tag{C.23}$$

$$\sum_{i\in N_f\cap N^k}\sum_{\iota\in\Lambda_f} Q_\iota^f\left(W_{f,i}^{k,t}\right)_\iota \le m_f^{k,t}Z_f^{k,t}, \qquad \forall k\in M, t\in T, f\in F^k \tag{C.24}$$

$$Q_\iota^f \ge 0, \qquad \forall f\in F, \iota\in\Lambda_f$$

$$R_f^{k,t}\in\mathbb{Z}^+, Z_f^{k,t}\in\{0,1\}, \qquad \forall k\in M, t\in T, f\in F^k$$

$$X_{f,g}^{k,t}\in\mathbb{Z}^+, \qquad \forall k\in M, \forall t\in T, f\ne g\in\hat{F}^k$$

$$\gamma_{f,g}^{k,t_1,t_2}\in\{0,1\}, \qquad \forall k\in M, t_1,t_2\in\hat{T}, f,g\in F^k; t_1<t_2$$

$$\alpha^{k,t}\ge 0, \beta^{k,t}\ge 0, \qquad \forall k\in M, t\in T$$

$$\lambda_{f,g}^{k,t,u}\ge 0, \qquad \forall k\in M, t\in T, u,g\in F^k, f\in\hat{F}^k; f\ne g.$$

LSP-Subproblem, $\forall f\in F$:

Minimize $\sum_{t\in T}\sum_{i\in N_f} h_{f,i}I_{f,i}^t - \sum_{t\in T}\sum_{i\in N_f}\sum_{k\in M^i}\left(p_{f,i}^k\mu^{k,t}+\xi_f^{k,t}+\nu_f^{k,t}+\phi_f^{k,t}\right)W_{f,i}^{k,t} - \upsilon^f$

subject to: Constraints (5.1)-(5.3)

$$\sum_{i\in N_f\cap N^k} p_{f,i}^k W_{f,i}^{k,t}\le c^{k,t}, \qquad \forall k\in M, t\in T; f\in F^k$$

$$I_{f,i}^t\ge 0, G_{f,i}^t\ge 0, \qquad \forall t\in T, i\in N_f$$

$$P_{f,i}^{k,t}\ge 0, \qquad \forall t\in T, i\in N_f, k\in M^i$$

$$W_{f,i}^{k,t}\in\mathbb{Z}^+, \qquad \forall t\in T, i\in N_f, k\in M^i.$$

C.5 LSP2 Decomposition Scheme

We present the formulations for this decomposition scheme as follows:

Sets:

Λ_f: Index set of the columns corresponding to product family f, $\forall f\in F$.

Parameters:

$(\_)_\iota$: Value of subproblem variable "$\_$" fixed in the ι^{th} column of family f, $\forall f \in F$. Note that index f is implicitly indicated by the enclosed variable "$\_$".

Variables:

Q_ι^f: Binary variable indicating the inclusion/exclusion of the ι^{th} column of family f, $\forall f \in F, \iota \in \Lambda_f$, and relaxed to take continuous values.

v^f: Dual variable associated with the Convexity Constraint (C.25), $\forall f \in F$.

$\mu^{k,t}$: Dual variable associated with the Time Capacity Constraint (C.26), $\mu^{k,t} \leq 0$, $\forall k \in M, t \in T$.

$\tilde{\chi}_g^{k,t}$: Dual variable associated with Constraint (C.27), $\forall k \in M, t \in T, g \in F^k$.

$\tilde{\psi}_f^{k,t}$: Dual variable associated with Constraint (C.28), $\forall k \in M, t \in T, f \in F^k$.

$\tilde{\xi}_u^{k,t}$: Dual variable associated with Constraint (C.29), $\forall k \in M, t \in T, u \in F^k$.

$\tilde{v}_u^{k,t}$: Dual variable associated with Constraint (C.30), $\forall k \in M, t \in T, u \in F^k$.

The problem decomposition is as follows:

LSP2-Master:

$$\text{Minimize} \sum_{t\in T}\sum_{f\in F}\sum_{i\in N_f} h_{f,i} \sum_{\iota\in\Lambda_f} Q_\iota^f \left(I_{f,i}^t\right)_\iota + \sigma \sum_{k\in M}\sum_{f\neq g\in F^k} s_{f,g}^k \left(\sum_{t\in T} X_{f,g}^{k,t} + \sum_{t_1<t_2\in\hat{T}} \gamma_{f,g}^{k,t_1,t_2}\right)$$

subject to: Constraints (5.5) -(5.8), (5.11)-(5.19), (5.22)

$$\sum_{\iota\in\Lambda_f} Q_\iota^f = 1, \qquad \forall f \in F \tag{C.25}$$

$$\sum_{f\in F^k}\sum_{i\in N_f\cap N^k} p_{f,i}^k \sum_{\iota\in\Lambda_f} Q_\iota^f \left(W_{f,i}^{k,t}\right)_\iota \leq c^{k,t}, \qquad \forall k \in M, t \in T \tag{C.26}$$

$$\sum_{f\in\hat{F}^k\setminus\{g\}} X_{f,g}^{k,t} = \sum_{\iota\in\Lambda_g} Q_\iota^g \left(R_g^{k,t}\right)_\iota, \qquad \forall k \in M, t \in T, g \in F^k \tag{C.27}$$

$$\sum_{g\in\hat{F}^k\setminus\{f\}} X_{f,g}^{k,t} = \sum_{\iota\in\Lambda_f} Q_\iota^f \left(R_f^{k,t}\right)_\iota, \qquad \forall k \in M, t \in T, f \in F^k \tag{C.28}$$

$$\sum_{g\in F^k} \lambda_{0,g}^{k,t,u} = \sum_{\iota\in\Lambda_u} Q_\iota^u \left(Z_u^{k,t}\right)_\iota, \qquad \forall k \in M, t \in T, u \in F^k \tag{C.29}$$

$$\sum_{f\in \hat{F}^k\setminus\{u\}} \lambda_{f,u}^{k,t,u} = \sum_{\iota\in\Lambda_u} Q_\iota^u \left(Z_u^{k,t}\right)_\iota, \qquad \forall k \in M, t \in T, u \in F^k \tag{C.30}$$

$$Q_\iota^f \geq 0, \qquad \forall f \in F, \iota \in \Lambda_f$$

$$R_f^{k,t} \in \mathbb{Z}^+, \qquad \forall k \in M, t \in T, f \in F^k$$

$$X_{f,g}^{k,t} \in \mathbb{Z}^+, \qquad \forall k \in M, t \in T, f \neq g \in \hat{F}^k$$

$$\gamma_{f,g}^{k,t_1,t_2} \in \{0,1\}, \qquad \forall k \in M, t_1, t_2 \in \hat{T}, f, g \in F^k; t_1 < t_2$$

$$\alpha^{k,t} \geq 0, \beta^{k,t} \geq 0, \qquad \forall k \in M, t \in T$$

$$\lambda_{f,g}^{k,t,u} \geq 0, \qquad \forall t \in T, u, g \in F^k, f \in \hat{F}^k; f \neq g.$$

LSP2-Subproblem, $\forall f \in F$:

$$\text{Minimize} \sum_{t\in T}\sum_{i\in N_f} h_{f,i} I_{f,i}^t - \sum_{t\in T}\sum_{\substack{k\in M;\\ f\in F^k}} \left[\left(\tilde{\chi}_f^{k,t} + \tilde{\psi}_f^{k,t}\right) R_f^{k,t} + \left(\tilde{\xi}_f^{k,t} + \tilde{v}_f^{k,t}\right) Z_f^{k,t}\right] - \upsilon^f$$

subject to: Constraints (5.1)-(5.3), (5.13)-(5.16)

$$\sum_{i\in N_f\cap N^k} p_{f,i}^k W_{f,i}^{k,t} \leq c^{k,t}, \qquad \forall k \in M, t \in T; f \in F^k$$

$$R_f^{k,t} \in \mathbb{Z}^+, Z_f^{k,t} \in \{0,1\}, \qquad \forall k \in M, t \in T; f \in F^k$$

$$I_{f,i}^t \geq 0, G_{f,i}^t \geq 0, \qquad \forall t \in T, i \in N_f$$

$$P_{f,i}^{k,t} \geq 0, \qquad \forall t \in T, i \in N_f, k \in M^i$$

$$W_{f,i}^{k,t} \in \mathbb{Z}^+, \qquad \forall t \in T, i \in N_f, k \in M^i.$$

C.6 UCLSP Decomposition Scheme

We present the formulations for this decomposition scheme as follows:

Sets:

Λ_f: Index set of the columns corresponding to product family f, $\forall f \in F$.

Parameters:

$(\_)_\iota$: Value of subproblem variable "$\_$" fixed in the ι^{th} column of family f, $\forall f \in F$. Note that index f is implicitly indicated by the enclosed variable "$\_$".

Variables:

Q_ι^f: Binary variable indicating the inclusion/exclusion of the ι^{th} column of family f, $\forall f \in F$, $\iota \in \Lambda_f$, and relaxed to take continuous values.

v^f: Dual variable associated with the Convexity Constraint (C.31), $\forall f \in F$.

$\tilde{\vartheta}_f^{k,t}$: Dual variable associated with Constraint (C.32), $\tilde{\vartheta}_f^{k,t} \leq 0$, $\forall k \in M, t \in T, f \in F^k$.

$\tilde{\varsigma}_f^{k,t}$: Dual variable associated with Constraint (C.33), $\tilde{\varsigma}_f^{k,t} \geq 0$, $\forall k \in M, t \in T, f \in F^k$.

$\tilde{\xi}_u^{k,t}$: Dual variable associated with Constraint (C.34), $\forall k \in M, t \in T, u \in F^k$.

$\tilde{v}_u^{k,t}$: Dual variable associated with Constraint (C.35), $\forall k \in M, t \in T, u \in F^k$.

The problem decomposition is as follows:

UCLSP-Master:

$$\text{Minimize} \sum_{t\in T}\sum_{f\in F}\sum_{i\in N_f} h_{f,i} \sum_{\iota\in\Lambda_f} Q_\iota^f \left(I_{f,i}^t\right)_\iota + \sigma \sum_{k\in M}\sum_{f\neq g\in F^k} s_{f,g}^k \left(\sum_{t\in T} X_{f,g}^{k,t} + \sum_{t_1<t_2\in\hat{T}} \gamma_{f,g}^{k,t_1,t_2}\right)$$

subject to: Constraints (5.3) -(5.14), (5.17)-(5.19), (5.22)

$$\sum_{\iota\in\Lambda_f} Q_\iota^f = 1, \qquad \forall f \in F \tag{C.31}$$

$$\sum_{i\in N_f\cap N^k} W_{f,i}^{k,t} \leq m_f^{k,t} \sum_{\iota\in\Lambda_u} Q_\iota^f \left(Z_f^{k,t}\right)_\iota, \qquad \forall k \in M, t \in T, f \in F^k \tag{C.32}$$

$$R_f^{k,t} \geq \sum_{\iota\in\Lambda_u} Q_\iota^f \left(Z_f^{k,t}\right)_\iota, \qquad \forall k \in M, t \in T, f \in F^k \tag{C.33}$$

$$\sum_{g\in F^k} \lambda_{0,g}^{k,t,u} = \sum_{\iota\in\Lambda_u} Q_\iota^u \left(Z_u^{k,t}\right)_\iota, \qquad \forall k \in M, t \in T, u \in F^k \tag{C.34}$$

$$\sum_{f\in\hat{F}^k\setminus\{u\}} \lambda_{f,u}^{k,t,u} = \sum_{\iota\in\Lambda_u} Q_\iota^u \left(Z_u^{k,t}\right)_\iota, \qquad \forall k \in M, t \in T, u \in F^k \tag{C.35}$$

$$Q_{\iota}^{f} \geq 0, \qquad \forall f \in F, \iota \in \Lambda_f$$

$$P_{f,i}^{k,t} \geq 0, W_{f,i}^{k,t} \in \mathbb{Z}^{+}, \qquad \forall t \in T, f \in F, i \in N_f, k \in M^i$$

$$X_{f,g}^{k,t} \in \mathbb{Z}^{+}, \qquad \forall k \in M, t \in T, f \neq g \in \hat{F}^k$$

$$\gamma_{f,g}^{k,t_1,t_2} \in \{0,1\}, \qquad \forall k \in M, t_1, t_2 \in \hat{T}, f, g \in F^k; t_1 < t_2$$

$$\alpha^{k,t} \geq 0, \beta^{k,t} \geq 0, \qquad \forall k \in M, t \in T$$

$$\lambda_{f,g}^{k,t,u} \geq 0, \qquad \forall t \in T, u, g \in F^k, f \in \hat{F}^k; f \neq g.$$

UCLSP-Subproblem, $\forall f \in F$:

Minimize $\sum_{t \in T} \sum_{i \in N_f} h_{f,i} I_{f,i}^{t} - \sum_{t \in T} \sum_{\substack{k \in M; \\ f \in F^k}} \left(\tilde{\vartheta}_f^{k,t} + \tilde{\varsigma}_f^{k,t} + \tilde{\xi}_f^{k,t} + \tilde{v}_f^{k,t}\right) Z_f^{k,t} - \upsilon^f$

subject to: Constraints (5.1)

$$G_{f,i}^{t} \leq \omega_{f,i}^{t} \sum_{k \in M^i} Z_f^{k,t}, \qquad \forall t \in T, i \in N_f$$

$$Z_f^{k,t} \in \{0,1\}, \qquad \forall k \in M, t \in T; f \in F^k$$

$$I_{f,i}^{t} \geq 0, G_{f,i}^{t} \geq 0, \qquad \forall t \in T, i \in N_f.$$

Bibliography

Armentano, V. A., França, P. M., and de Toledo, F. M. B. (1999). A network flow model for the capacitated lot-sizing problem. *Omega*, *27*(2), 275-284.

Artzner, P., Delbaen, F., Eber, J.-M., and Heath, D. (1999). Coherent measures of risk. *Mathematical Finance*, *9*, 203-228.

Atamtürk, A., and Muñoz, J. C. (2004). A study of the lot-sizing polytope. *Mathematical Programming*, *99*(3), 443-465.

Ayhan, H., and Lennon Olsen, T. (2000). Scheduling of multi-class single-server queues under nontraditional performance measures. *Operations Research*, *48*(3), 482-489.

Baker, T., and Muckstadt, Jr., J. A. (1989). *The CHES problems*. New Providence, NJ: Chesapeake Decision Sciences, Inc.

Balasubramanian, J., and Grossmann, I. E. (2003). Scheduling optimization under uncertainty--an alternative approach. *Computers & Chemical Engineering*, *27*(4), 469-490.

Barany, I., Van Roy, T. J., and Wolsey, L. A. (1984). Strong formulations for multi-item capacitated lot sizing. *Management Science*, *30*, 1255-1261.

Bazaraa, M. S., Sherali, H. D., and Shetty, C. M. (2006). *Nonlinear Programming: Theory and Algorithms* (3rd ed.). Wiley-Interscience.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*(1), 238-252.

Billington, P. J., McClain, J. O., and Thomas, L. J. (1983). Mathematical programming approaches to capacity-constrained MRP systems: review, formulation and problem reduction. *Management Science*, *29*, 1126-1141.

Birge, J. R., and Louveaux, F. (1997). *Introduction to Stochastic Programming*. Springer.

Birge, J. R., and Wallace, S. W. (1986). Refining bounds for stochastic linear programs with linearly transformed independent random variables. *Operations Research Letters*, *5*(2), 73-77.

Birge, J. R., and Wets, R. J.-B. (1986). Designing approximation schemes for stochastic optimization problems, in particular for stochastic programs with recourse. *Stochastic Programming 84 Part I*, Mathematical Programming Studies (Vol. 27, pp. 54-102). Springer Berlin Heidelberg.

Bitran, G. R., and Yanasse, H. H. (1982). Computational complexity of the capacitated lot size problem. *Management Science*, *28*(10), 1174-1186.

Bruneau, P., Gelgon, M., and Picarougne, F. (2010). Parsimonious reduction of Gaussian mixture models with a variational-Bayes approach. *Pattern Recognition*, *43*(3), 850-858.

Clark, C. E. (1961). The greatest of a finite set of random variables. *Operations Research*, *9*(2), 145-162.

Congram, R. K., Potts, C. N., and van de Velde, S. L. (2002). An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal of Computing*, *14*(1), 52-67.

Cosmadakis, S. S., and Papadimitriou, C. H. (1984). The Traveling Salesman Problem with Many Visits to Few Cities. *SIAM Journal on Computing*, *13*(1), 99-108.

Daniels, R. L., and Kouvelis, P. (1995). Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science*, *41*(2), 363-376.

Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Operations Research*, *2*(4), 393-410.

Davenport, A. (2002). A survey of techniques for scheduling with uncertainty. Retrieved from http://www.eil.utoronto.ca/profiles/chris/gz/uncertainty-survey.ps

De, P., Ghosh, J. B., and Wells, C. E. (1992). Expectation-variance analysis of job sequences under processing time uncertainty. *International Journal of Production Economics*, *28*(3), 289-297.

Degraeve, Z., and Jans, R. (2007). A new Dantzig-Wolfe reformulation and branch-and-price algorithm for the capacitated lot-sizing problem with setup times. *Operations Research*, *55*(5), 909-920.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*(1), 1-38.

Diaby, M., Bahl, H. C., Karwan, M. H., and Zionts, S. (1992). A Lagrangean relaxation approach for very-large-scale capacitated lot-sizing. *Management Science*, *38*(9), 1329-1340.

Dodin, B. (1996). Determining the optimal sequences and the distributional properties of their completion times in stochastic flow shops. *Computers & Operations Research*, *23*(9), 829-843.

Dogan, M. E., and Grossmann, I. E. (2006). A decomposition method for the simultaneous planning and scheduling of single-stage continuous multiproduct plants. *Industrial & Engineering Chemistry Research*, *45*(1), 299-315.

Edirisinghe, N. C. P., and Ziemba, W. T. (1994). Bounds for two-stage stochastic programs with fixed recourse. *Mathematics of operations research*, *19*(2), 292-313.

Eppen, G. D., and Martin, R. K. (1987). Solving multi-item capacitated lot-sizing problems using variable redefinition. *Operations Research*, *35*(6), 832-848.

Fandel, G., and Stammen-Hegene, C. (2006). Simultaneous lot sizing and scheduling for multi-product multi-level production. *International Journal of Production Economics*, *104*(2), 308-316.

Federgruen, A., and Tzur, M. (1991). A simple forward algorithm to solve general dynamic lot sizing models with n periods in O(n log n) or O(n) time. *Management Science*, *37*(8), 909-925.

Fleischmann, B. (1994). The discrete lot-sizing and scheduling problem with sequence-dependent setup costs. *European Journal of Operational Research*, *75*(2), 395-404.

Florian, M., and Klein, M. (1971). Deterministic production planning with concave costs and capacity constraints. *Management Science*, *18*(1), 12-20.

Frauendorfer, K., and Kall, P. (1988). A solution method for SLP recourse problems with arbitrary multivariate distributions-the independent case. *Problems of Control and Information Theory*, *17*, 177-205.

Gavish, B., and Graves, S. (1978). *The travelling salesman problem and related problems*. Working Paper, .

Geoffrion, A. M., and Graves, G. W. (1974). Multicommodity distribution system design by benders decomposition. *Management Science*, *20*(5), 822-844.

Ghoniem, A., and Sherali, H. D. (2009). Complementary column generation and bounding approaches for set partitioning formulations. *Optimization Letters*, *3*(1), 123-136.

Glover, F., and Sherali, H. D. (2005). Some classes of valid inequalities and convex hull characterizations for dynamic fixed-charge problems under nested constraints. *Annals of Operations Research*, *140*(1), 215-233.

Graves, S. C. (1982). Using Langrangean techniques to solve hierarchical production planning problems. *Management Science*, *28*(3), 260-275.

Grigoriev, A., and van de Klundert, J. (2006). On the high multiplicity traveling salesman problem. *Discrete Optimization*, *3*(1), 50-62.

Grosso, A., Della Croce, F., and Tadei, R. (2004). An enhanced Dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem. *Operations Research Letters*, *32*, 68-72.

Gupta, D., and Magnusson, T. (2005). The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times. *Computers & Operations Research*, *32*(4), 727-747.

Haase, K., and Kimms, A. (2000). Lot sizing and scheduling with sequence-dependent setup costs and times and efficient rescheduling opportunities. *International Journal of Production Economics*, *66*(2), 159-169.

Hindi, K. S. (1995). Solving the single-item, capacitated dynamic lot-sizing problem with startup and reservation costs by tabu search. *Computers & Industrial Engineering*, *28*(4), 701-707.

van Hoesel, S., Wagelmans, A., and Moerman, B. (1994). Using geometric techniques to improve dynamic programming algorithms for the economic lot-sizing problem and extensions. *European Journal of Operational Research*, *75*, 312-331.

Hürlimann, W. (2002). Analytical bounds dor two Value-at-Risk functionals. *ASTIN Bulletin*, *32*(2), 235-265.

Jung, Y., Nagasawa, H., and Nishiyama, N. (1990). Bicriterion single-stage scheduling to minimize both the expected value and the variance of the total flow time. *Journal of Japan Industrial Management Association*, *39*, 76–82.

Kalymon, B. A. (1972). A decomposition algorithm for arborescence inventory systems. *Operations Research*, *20*(4), 860-874.

Kang, S., Malik, K., and Thomas, L. J. (1999). Lot-sizing and scheduling on parallel machines with sequence-dependent setup costs. *Management Science*, *45*(2), 273-289.

Karrer, B., and Newman, M. E. J. (2009). Random acyclic networks. *Physical Review Letters*, *102*(12), 128701(4pages).

Kaut, M., and Wallace, S. W. (2003). Evaluation of Scenario-Generation Methods for Stochastic Programming. *World Wide Web, Stochastic Programming E-print Series*, *14*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi =10.1.1.15.6773

Kimms, A. (1997). Demand shuffle - A method for multilevel proportional lot sizing and scheduling. *Naval Research Logistics*, *44*(4), 319-340.

Kleywegt, A. J., Shapiro, A., and Homem-de-Mello, T. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, *12*(2), 479-502.

Koulamas, C. (1996). Single-machine scheduling with time windows and earliness/tardiness penalties. *European Journal of Operational Research*, *91*(1), 190-202.

Kouvelis, P., Daniels, R. L., and Vairaktarakis, G. (2000). Robust scheduling of a two-machine flow shop with uncertain processing times. *IIE Transactions*, *32*(5), 421-432.

Law, A., and Kelton, W. D. (1999). *Simulation Modeling and Analysis* (3rd ed.). McGraw-Hill Science/Engineering/Math.

Leung, J. M. Y., Magnanti, T. L., and Vachani, R. (1989). Facets and algorithms for capacitated lot sizing. *Mathematical Programming*, *45*(2), 331-359.

Loparic, M., Marchand, H., and Wolsey, L. A. (2003). Dynamic knapsack sets and capacitated lot-sizing. *Mathematical Programming*, *95*(1), 53-69.

Mak, W.-K., Morton, D. P., and Wood, R. K. (1999). Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, *24*, 47-56.

McDaniel, D., and Devine, M. (1977). A modified Benders' partitioning algorithm for mixed integer programming. *Management Science*, *24*(3), 312-319.

McKay, K. N., Safayeni, F. R., and Buzacott, J. A. (1988). Job-Shop Scheduling Theory: What Is Relevant? *Interfaces*, *18*(4), 84-90.

McLachlan, G., and Peel, D. (2000). *Finite Mixture Models* (1st ed.). Wiley-Interscience.

Meyr, H. (2002). Simultaneous lot-sizing and scheduling on parallel machines. *European Journal of Operational Research*, *139*(2), 277-292.

Miller, A. J., Nemhauser, G. L., and Savelsbergh, M. W. P. (2000). On the capacitated lot-sizing and continuous 0-1 knapsack polyhedra. *European Journal of Operational Research*, *125*(2), 298-315.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM*, *7*(4), 326-329.

Nagasawa, H., and Shing, C. (1998). Interactive decision system in stochastic multiobjective scheduling to minimize the expected value and variance of total flow time. *Journal-Operations Research Society of Japan*, *41*(2), 261-278.

Nemhauser, G. L., and Wolsey, L. A. (1999). *Integer and Combinatorial Optimization* (2nd ed.). New York, NY: Wiley.

Nielsen, F., and Nock, R. (2009). Clustering multivariate normal distributions. *Emerging Trends in Visual Computing* (pp. 164-174). Retrieved from http://dx.doi.org.ezproxy.lib.vt.edu:8080/10.1007/978-3-642-00826-9_7

Ogryczak, Wl., and Ruszczynski, A. (2002). Dual stochastic dominance and related mean-risk models. *SIAM Journal on Optimization*, *13*(1), 60-78.

Öncan, T., AltInel, I. K., and Laporte, G. (2009). A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, *36*(3), 637-654.

Pearson, K. (1894). Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, *185*, 71-110.

Pinedo, M. (2001). *Scheduling: Theory, Algorithms, and Systems* (2nd ed.). Prentice Hall.

Pochet, Y. (1988). Valid inequalities and separation for capacitated economic lot sizing. *Operations Research Letters*, *7*(3), 109-115.

Porter, R. B., and Gaumnitz, J. E. (1972). Stochastic dominance vs. Mean-variance portfolio analysis: an empirical evaluation. *The American Economic Review*, *62*(3), 438-446.

Portougal, V., and Trietsch, D. (1998). Makespan-related criteria for comparing schedules in stochastic environments. *The Journal of the Operational Research Society*, *49*(11), 1188-1195.

Rockafellar, R. T., and Uryasev, S. (2000). Optimization of Conditional Value-at-Risk. *Journal of Risk*, *2*, 21-41.

Ross, S. M., and Pekoz, E. A. (2007). *A Second Course in Probability*. pekozbooks.

Runnalls, A. R. (2007). Kullback-Leibler approach to gaussian mixture reduction. *IEEE Transactions on Aerospace and Electronic Systems*, *43*(3), 989-999.

Sabuncuoglu, I., and Bayiz, M. (2000). Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research*, *126*(3), 567-586.

Salmond, D. J. (1988). *Mixture Reduction Algorithms for Uncertain Tracking*. Retrieved from http://stinet.dtic.mil/oai/oai?&verb=getRecord&metadataPrefix=html&identifier=ADA197641

Sarin, S. C., Nagarajan, B., Jain, S., and Liao, L. (2007). Analytic evaluation of the expectation and variance of different performance measures of a schedule on a single machine under processing time variability. *Journal of Combinatorial Optimization*, *17*(4), 400-416.

Sarin, S. C., Sherali, H. D., and Bhootra, A. (2005). New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints. *Operations Research Letters*, *33*(1), 62-70.

Sarin, S. C., Sherali, H. D., and Yao, L. (2011). New formulation for the high multiplicity asymmetric traveling salesman problem with application to the Chesapeake problem. *Optimization Letters*, *5*(2), 259-272.

Shah, N. (2004). Pharmaceutical supply chains: Key issues and strategies for optimisation. *Computers & Chemical Engineering*, *28*(6-7), 929-941.

Sherali, H. D., and Lunday, B. (2010). On generating maximal nondominated Benders cuts. *Annals of Operations Research*.

Sherali, H. D., Sarin, S. C., and Tsai, P.-F. (2006). A class of lifted path and flow-based formulations for the asymmetric traveling salesman problem with and without precedence constraints. *Discrete Optimization*, *3*(1), 20-32.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis* (1st ed.). Chapman and Hall/CRC.

Skutella, M., and Uetz, M. (2005). Stochastic Machine Scheduling with Precedence Constraints. *SIAM Journal on Computing*, *34*(4), 788.

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.

Tebboth, J. R. (2001). *A Computational Study of Dantzig-Wolfe Decomposition*. UK.: University of Buckingham.

Tran, V. H., Reinelt, G., and Bock, H. G. (2006). BoxStep methods for crew pairing problems. *Optimization and Engineering*, *7*(1), 33-46.

Vieira, G. E., Herrmann, J. W., and Lin, E. (2003). Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling*, *6*(1), 39-62.

Wagelmans, A., van Hoesel, S., and 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*, 145-156.

Wagner, H. M., and Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management Science*, *5*(1), 89-96.

Wang, W., and Ahmed, S. (2008). Sample average approximation of expected value constrained stochastic programs. *Operations Research Letters*, *36*(5), 515-519.

Williams, J. L. (2003). *Gaussian Mixture Reduction for Tracking Multiple Maneuvering Targets in Clutter*. Retrieved from http://stinet.dtic.mil/oai/oai?&verb=getRecord&metadataPrefix=html&identifier=ADA415317

Wong, R. T. (1980). Integer programming formulations of the traveling salesman problem. *Proceedings of the IEEE international conference of circuits and computers* (pp. 149-152). New York.

Zangwill, W. I. (1969). A Backlogging Model and a Multi-Echelon Model of a Dynamic Economic Lot Size Production System-A Network Approach. *Management Science*, *15*(9), 506-527.