OPTIMIZATION PROBLEMS IN OPERATIONS MANAGEMENT: APPLICATIONS IN MULTI-PERIOD DECISION MAKING, TELECOMMUNICATIONS AND SCHEDULING

APPROVED BY SUPERVISORY COMMITTEE:

[illegible]

Dr. Chelliah Sriskandarajah, Co-Chair

MWDawande

Dr. Milind Dawande, Co-Chair

Rakesh Gupta

Dr. Rakesh Gupta

[illegible]

Dr. Kathryn E. Stecke

Dedicated to my loving Mother Chandra and late loving Father R.B.

OPTIMIZATION PROBLEMS IN OPERATIONS MANAGEMENT: APPLICATIONS IN MULTI-PERIOD DECISION MAKING, TELECOMMUNICATIONS AND SCHEDULING

by

SANJEEWA A NARANPANAWE, B.S., M.S.

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN MANAGEMENT SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

December 2005

ACKNOWLEDGMENTS

I would like to start by expressing my sincere gratitude to my advisors, Prof. Chelliah Sriskandarajah and Prof. Milind Dawande for their guidance and advice throughout my stay at UT Dallas. Without their support, I would never have been successful in my endeavor. They were always helpful and willing to answer any question I had in my research work. I relied heavily on both of them for advice and guidance regarding my work and personal life.

I am thankful to Prof. Suresh Sethi for his help and guidance during the past four years. I am also thankful to Prof. Kathryn Stecke for her kindness in serving on my dissertation committee, and all the helpful advice given to me during my years at UT Dallas.

I'm grateful to Prof. Rakesh Gupta for his guidance with my research. He always found time for me and help me in my work. I am also grateful to Prof. Srinagesh Gaverneni of Cornell University for introducing me to some interesting research problems.

I also liked to thank Prof. Shun-Chen Niu for his advice and guidance. He always welcomed me into his office whenever I had a question. My thanks extend to Prof. Holly Lutze and Prof. Metin Çakanyildirim for their advice and help. A special thanks to Prof. Divakar Rajamani for his help and advice.

A special thanks to all my fellow doctoral students at UT Dallas; Abdulla, Annabelle, Daisy, Emre, Gan, Gocken, Jenny, Manoj, Priyantha, Sanjay, Sandy, Siroung and Wendy, who made my student life at UT Dallas enjoyable.

I would also like to extend a very special thank you to Babara, Sophia and LeeAnn, for their help during the past years.

I'm very grateful to my family, who's support and encouragement I could always count on. I'm grateful to my late Father, who encouraged me to achieve higher things in life and taught me the value of education. I'm thankful to my dear Mother who has always given me unconditional love. I'm grateful to my sister Shyama and her husband Shantha, my brothers and sisters-in-law; Rohan and wife Vipuli, Ajith and wife Marian, Rajeendra and wife Samanthi, for the support and encouragement during my very long student life. Finally, a very special thanks to my nephews and nieces; Chamindra, Heshan, Shanaka, Wyomi, Tharinda, Nimesh and Nimeshika, who all inspired me greatly.

December 2005.

OPTIMIZATION PROBLEMS IN OPERATIONS MANAGEMENT:

APPLICATIONS IN MULTI-PERIOD DECISION MAKING,

TELECOMMUNICATIONS AND SCHEDULING

Publication No. ____________________

Sanjeewa A Naranpanawe., Ph.D.

The University of Texas at Dallas, 2005

Supervising Professors: Dr. Chelliah Sriskandarajah, Dr. Milind Dawande

Optimization plays a key role in industrial applications. The problems investigated in this research span multiple industries in Production Operations Management, Scheduling and Telecommunications. Most of the optimization problems encountered are NP-hard and therefore difficult to solve. We provide below a brief description of the results obtained for the specific problems considered in this study.

In the first problem discussed in Chapters two and three, we present a structural and computational investigation of a new class of weak forecast horizons – minimal forecast horizons under the assumption that future demands are integer multiples of a positive real number. Apart from being appropriate in most practical instances, the discreteness assumption offers a significant reduction in the length of a minimal

forecast horizon over the one using the classical notion of continuous future demands. We provide several conditions under which a discrete-demand forecast horizon is also a continuous-demand forecast horizon. We also show that the increase in the cost resulting from using a discrete minimal forecast horizon instead of the classical minimal forecast horizon is modest.

In Chapter four, we consider the problem of traffic grooming in all-optical networks with the objective of throughput maximization. We present an integer programming formulation which addresses this objective while constraining the number of optical transceivers at each node, the link load, and the capacity of each lightpath. Based on the structural properties of the problem we develop an heuristic algorithm based on a column generation technique. The algorithm is easy to implement, requires a modest amount of CPU time and provides high quality solutions. To ascertain the quality of solutions obtained by our column generation based algorithm, we present an alternative formulation which allows us to develop an upper bound using a Lagrangian relaxation technique. An extensive computational study is presented to justify our claims.

The final problem discussed in Chapter five of this study is a two-stage blocking flow-shop scheduling problem with a material handling constraint. We develop heuristic algorithms and a meta-heuristic search method to solve this problem. Such problems are common in production system design, service facilities design, or specialty jobs such as petrochemical processing. The absence of intermediate buffers between the stages here causes the blocking of jobs when downstream machines are occupied. The objective is to minimize the makespan. We show that this problem is unary

NP-hard. We then explore the special structural features of this problem and develop two problem-specific solution construction heuristics for different job processing time scenarios. We show that these heuristics can speed up solution evolution rate by providing good starting points for a genetic algorithm, particularly when the problem is large and computational efficiency is paramount.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This study focuses on the the application of optimization techniques in difficult practical problems. Most discrete optimization problems we encounter in practice require unique solution approaches. The approaches depend on problem structures as well as the managerial implications involved. Every problem discussed in this study can be modeled using pure or mixed integer programs. However, it is very difficult to obtain optimal solutions to these problems since most are NP-hard. Therefore, heuristic and meta-heuristic methods become very useful in practice. In this study we obtain optimal solutions where possible, and heuristic solutions where optimality is difficult to achieve.

The first problem deals with computing forecast horizons in multi-period decision problems. In Chapter two, we discuss the viability of integer programming as a tool to compute forecast horizons. In Chapter three, we extend the discussion and introduce a new type of weak forecast horizon - a forecast horizon subject to discrete future demand. The second problem focusses on traffic routing in optical networks with wavelength division multiplexing. The problem is first formulated as an integer program. We then develop an algorithm based on column generation to solve this problem. The third problem looks at a problem of production scheduling under flow-shop environment subject to blocking. We apply meta-heuristic search methods, specifically genetic algorithms, as a solution approach for this problem. A brief introduction to these problems is provided below.

1.1 Computing Forecast Horizons subject to Discrete Future Demand

In a multi-period decision making environment, the need for a robust decision making process requires managers to evaluate the role that future business data (e.g., costs, demands) play in determining current decisions. Managers involved in production and operations decisions typically use forecasts for the first several periods to evaluate current decisions. The signals obtained from forecast data, despite their inaccuracies, cannot be ignored and are routinely incorporated into current decisions. Here, a fundamental issue is the extent of the impact of future data on current decisions. In this context, the concept of a *forecast horizon* has been widely studied in the operations management/research literature (Chand et al. [20]).

We consider the problem of determining forecast horizons in the standard Dynamic Lot-Sizing Model (DLS). The DLS involves time-varying deterministic demands, and is concerned with determining appropriate production lot sizes in the presence of setup costs for production and holding costs for carrying inventory from one period to the next. Properties of optimal solutions of the DLS problem have been extensively studied in the literature. Since the initial inventory is zero, and backlogs are not allowed, production must take place in the first period; in each production period, the quantity produced is the sum of the demand over an integer number of periods [103].

A *decision horizon* refers to the number of the next few periods, say τ, for which decisions must be made in the current period. An integer T is referred to as a *forecast horizon* corresponding to the decision horizon τ if the data (i.e., demands and costs) beyond period T does not influence the optimal decisions for the first τ periods in any N-period problem with $N \geq T+1$. For a given decision horizon, the importance of a forecast horizon T, if it exists, is immediate: decisions for the first τ periods of some optimal solution of the T-period problem are also optimal for every instance of the problem with more than T periods. Consequently, there is no need to forecast data beyond period T. Also, if T is a forecast horizon for a given decision horizon, then so is every integer $T^{'} \geq T+1$. Thus, obtaining the *minimal* forecast horizon for a given decision horizon becomes relevant.

One of the earliest studies of forecast horizons in operations management is Wagner and Whitin [103][102], who analyzed a problem of determining lot sizes over time in the presence of setup costs, holding costs, and deterministic non-stationary demands. This problem is known as the *dynamic lot-size* (DLS) problem. Furthermore, the concept of forecast horizons has been extended to a wide variety of dynamic decision problems; e.g., capacity expansion (Rajagopalan [87][88]), cash management (Chand and Morton [21], Sethi [94]), inventory management (Bean et al. [15], Chand and Morton [22], Federgruen and Tzur [40], [41]), machine replacement (Chand and Sethi [23], Nair and Hopp [77]), plant location (Bastian and Volkmer [14], Daskin et al. [32]), production planning (Johnson and McClain [60], Kleindorfer and Lieber [64], Aronson et al. [4]), sequencing and scheduling (Remapala [90], Blocher and Chand [18]). For an extensive classified bibliography of the forecast horizon literature, see Chand et al. [20].

Given the strong properties associated with a forecast horizon, the issue of its existence is fundamental. However, for a wide variety of problems, the existence question has not been answered satisfactorily. It is possible that, a forecast horizon may not exist for a dynamic decision problem [95]. For the DLS model with constant initial demand, Chand et al. [24] established the existence of a forecast horizon (and, hence, a minimal forecast horizon). They also characterized demands d_1, (d_1, d_2), and (d_1, d_2, d_3) for which $T = 1, 2, 3$, respectively, are forecast horizons. For larger values of T, this characterization becomes exceedingly complex. A typical approach in the literature has been to propose procedures for identifying a forecast horizon under the assumption that it exists (see e.g., Chand and Morton [21, 22], Lundin and Morton [68], Bensoussan et al. [16]).

As a solution approach for computing forecast horizons, integer programming (IP) has been largely ignored by the research community. However, the modeling and structural advantages of this approach coupled with the recent significant developments in computational integer programming make for a strong case for its use in practice. We elaborate on these advantages below.

1. Recall that, given the demands for the first T periods, T is a forecast horizon for a given decision horizon if, *there exists* at least one common optimal solution to the decision horizon, for all problem instances with length geT. Computationally, this requirement of the existence of an optimal solution can become challenging in the presence of multiple optimal solutions. Existing algorithms that use necessary and sufficient conditions for an integer T to be forecast horizon may potentially need to investigate all these optimal solutions. This

enumeration of multiple optimum solutions becomes especially time consuming if T is *not* a forecast horizon.

As we see later in chapters two and three, necessary and sufficient conditions for a given integer T to be a forecast horizon can be posed as a feasibility problem for a linear integer program. The existence requirement noted above is implicitly modeled using variables and linear constraints. The search for a feasible integer solution, by state-of-the-art integer programming approaches, involves the use of powerful techniques such as linear implications over an integer domain (e.g., cutting planes), strong linear relaxations (e.g., linear programming), and heuristics. For modest values of T, the feasibility problems can be solved efficiently (Section 2.5).

2. Over the past 15 years, there has been a phenomenal progress in computational integer programming. The success of frameworks such as branch-and-cut [52][83] and branch-and-price [12] has lead to a revival of an area that for a long time had fallen out of favor. Today, several powerful implementations exist both in industry (e.g, CPLEX[1], LINDO[2], Xpress-MP[3]) and academia (MINTO [79], MIPO [10]) and are being widely used to solve a variety of large-scale, real-world problems.

3. The constraint-based nature of integer programming makes it preferable as a single framework to address several commonly occurring variants in a dynamic decision-making environment. For instance, consider the following variants of

[1]CPLEX is a trademark of ILOG, Inc.
[2]LINDO is a trademark of LINDO Systems, Inc.
[3]Xpress-MP is a trademark of Dash Optimization Ltd.

the classical DLS problem: (i) a DLS problem in the presence of a warehouse-capacity limit that prevents the ending inventory at any period from exceeding the limit, and (ii) a two-product DLS problem in the presence of individual and joint setup costs for the two products. It is straightforward to extend the integer programming approach for these variants.

The classical notion of a forecast horizon, addressed in the literature places no restrictions on the future data. The dynamic lot-size model of Wagner and Whitin [103] accommodates the possibility that a future demand could take any non-negative value. In practice, the context of the problem being investigated often allows us to specify additional characteristics of future demands. Typically, demand realizations obey a well-defined granularity that prohibits them from assuming any arbitrary values. For example, for a car manufacturer that needs to consider only integer valued demands, a forecast horizon for all non-negative demands is possibly an overestimate resulting in unnecessary forecasting. To illustrate this idea, consider the constant initial demand (set equal to one without loss of generality) dynamic lot-size problem studied in [24]. For the situation when the holding cost is one and the setup cost is 21, the minimal forecast horizon is 42. An interesting question is the impact, on the length of the minimal forecast horizon, of the knowledge that future demands are integer valued. It turns out (details are provided later in Section 4.3) that the minimal forecast horizon for the integer valued future demand scenario is only 32, a reduction of 23.8% in the length of the forecast horizon. As forecasting well into the future can be time consuming, expensive, and unreliable, the benefits of this reduction are substantial. The reduction in the length of the minimal forecast horizon is not restricted only to the case of integer valued future demands; it is a typical outcome whenever future demands are restricted to be an integer multiple of some fixed, positive real number. As the granularity of the discretization becomes finer, the length of the minimal forecast horizon increases and eventually coincides with that of the classical forecast horizon. This study formally introduces the concept of forecast horizons subject to discrete future demands and compares them, both structurally and computationally, to classical forecast horizons.

1.2 A Traffic Grooming Algorithm for Wavelength Routed Optical Networks

Optical fiber-based networks are expected to play a major part in the delivery of telecommunication services in the future (Cinkler, [28]). The proliferation of such networks has prompted the development of sophisticated techniques to allow increasing amounts of traffic to be transported on the same physical network. In the early phases of this evolution, Wavelength Division Multiplexing (WDM) was employed to allow multiple wavelengths of light to be transmitted on the same optical fiber. WDM networks use Optical Add Drop Multiplexes (OADMs) at the nodes, so as to be able to add and drop signals of specific wavelengths. Due to technological constraints, the OADMs performed these add/drop operations in the electrical domain (i.e., optical signals were converted to electrical signals before adding/dropping traffic). For such networks, each node requires expensive optical transmitters and receivers (also referred to as transponders/transceivers) thereby motivating network designers to choose designs that minimize the number of transponders used.

The current generation of commercial WDM systems incorporates optical cross-connects (OXCs) in addition to the OADMs at each node and increasingly utilize mesh (and not ring) based topologies (Zhu and Mukherjee, [110]). OXCs are switches capable of directing specific wavelengths of light from input ports to output ports entirely in the optical domain (i.e., without converting light signals to electrical signals). These nodes are generically referred to as wavelength routed switches or wavelength routers (Dutta and Rouskas, [37]) and the networks are referred to as all-optical networks. The essential mechanism of transport in all-optical networks is a *lightpath*, which is a communication channel established between two nodes in the network of OXCs that uses the same wavelength of light for its entire path (Dutta and Rouskas, [37]). Currently, each such wavelength of light is capable of carrying upto 10 Gbps, while 40 Gbps capable systems are on the verge of being introduced (Dutta and Rouskas, [37]). In order for such a high data-carrying capacity to be utilized efficiently, a number of lower-rate traffic streams must be multiplexed onto a single lightpath. This multiplexing process gives rise to the concept of traffic grooming, which consists of techniques that combine lower-speed traffic requests into available wavelengths in order to meet network design goals such as cost minimization or throughput maximization (Modino et al., [73] and Zhu and Mukherjee, [110]). In this study, we examine the objective of throughput maximization for such all-optical networks.

Traffic demands (or connection requests) for such optical networks can be characterized by a source (origin), a destination (termination point) and a granularity (bandwidth requested). Each such demand needs to be routed on a physical path in the network. All nodes other than the source and destination of each such path are referred to as intermediate nodes. The bandwidth of each such demand is typically one of OC-48, OC-12, OC-3 and OC-1[4]. Each such physical path can correspond to one single lightpath (in the case of *single-hop* networks) or could be a series of lightpaths (in the case of *multi-hop* networks). A major advantage of single-hop networks is that the signal travels in an all-optical medium and only a single wavelength is required to establish a connection. In multi-hop networks, a single connection between two nodes can sequentially utilize multiple lightpaths and therefore necessitates the ability to change/convert the wavelength of the transmission at network nodes. Wavelength conversion can be accomplished in the electrical domain but then the advantages of all-optical transmissions are lost. All-optical wavelength converters are available but are (currently) prohibitively expensive (Chu et al., [27]). Therefore, in this study we consider only single-hop networks, i.e., networks in which wavelength conversion is not allowed.

1.3 A Blocking Flowshop Scheduling Problem with a Material Handling Constraint

Blocking occurs in many flowshops that have limited or no buffer capacity between two successive processing stages or machining centers. When the buffer is full or when there is no space to discharge the completed job, the upstream processing stage is not allowed to release that job and thus, blocking occurs. This study develops heuristic algorithms which schedule jobs through processing systems such as paint shops, or specialty chemicals plants with special material binding constraints. Such systems with no buffers between processing stages are often characterized under the class

[4]An OC-1 is the lowest capacity optical communication channel which has a capacity of 51.84 Mbps. An OC-n can carry n*51.84 Mbps.

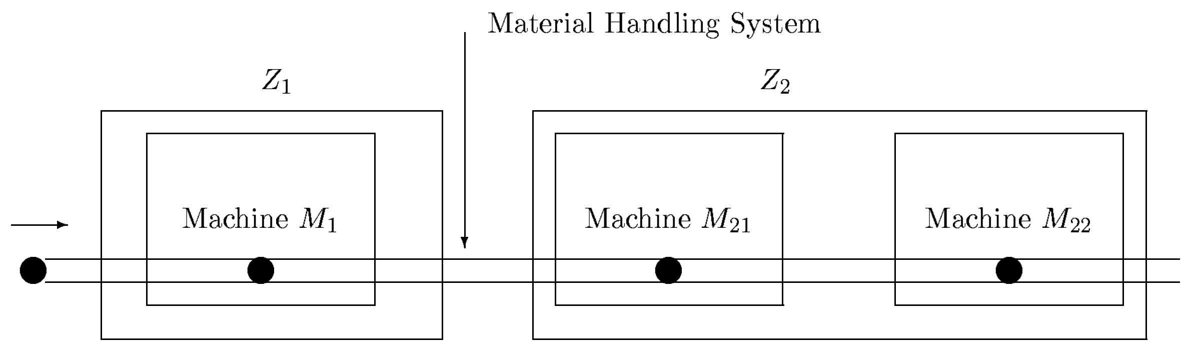

Figure 1.1: A Two-Stage Blocking Flowshop.

the "blocking flowshop." A blocking flowshop consists of a set of $k \geq 2$ processing stages $[Z_1, Z_2, \ldots, Z_k]$ with stage Z_i having $m_i \geq 1$ parallel machines. There are n jobs $\{Jj \mid 1 \leq j \leq n\}$ ready to be processed. A function $\pi(Jj) = [p_{j1}, p_{j2}, \ldots, p_{jk}]$ denotes the various tasks or equivalently the processing times required by Jj in stages $[Z_1, Z_2, \ldots, Z_k]$, respectively. This notation implies that job Jj will first be processed in Z_1 for p_{j1} units of time and then in Z_2 for p_{j2} units of time and so on, until it finally completes its processing in Z_k for p_{jk} units of time. Task p_{ji} may be processed on any of the m_i identical parallel machines $\{M_{i1}, M_{i2}, \ldots, M_{im_i}\}$ in Z_i.

For the classification of scheduling problems and notation, we follow Graham et al. [51], Lawler, et al. [66]. A problem is represented by the form $\alpha|\beta|\gamma$, which in our case is $Fk|blocking, m_1, m_2, \ldots, m_k|C_{max}$, where $\alpha = Fk$ represents the flowshop environment with k processing stages, $\beta = \{blocking, m_1, m_2, \ldots, m_k\}$ indicates the blocking restriction and the number of machines at each stage. $\gamma = C_{max}$ is the optimality criterion, i.e., minimization of finish time (makespan) is the goal. C_{max} is defined as follows. Let C_i be the finish time of job Ji under schedule S, then $C_{max} = \max_i\{C_i\}$.

Sequencing dissimilar jobs through a blocking flowshop quickly becomes surprisingly complex. We specifically consider the case of a blocking flowshop with $k = 2$ and operating with a special material handling restriction. This is a variation of the following problem: $F2|blocking, m_1 = 1, m_2 = 2|C_{max}$. Furthermore, we consider the case when Z_1 has only one machine (M_1) and Z_2 has two identical but sequentially laid machines (M_{21} and M_{22}). A single material handling system (like a conveyor) moves all the jobs between the machines, the path of every job being $M_1 \rightarrow M_{21} \rightarrow M_{22}$ (see Figure 1.1). Each job is first processed on M_1, then on either M_{21} or M_{22}. In other words, a job requires processing by M_1, and then by either M_{21} or M_{22} but not both. The material handling restriction acts as follows. A job completed on M_1 may move to M_{22} *through* M_{21} only when machine M_{21} is free. For this reason, a completed job on M_1 is blocked if machine M_{21} is processing a job even if M_{22} is free. We characterize this particular problem structure by $F2|blocking, MH, m_1 = 1, m_2 = 2|C_{max}$, where MH denotes the above special material handling restriction or constraint.

1.4 Organization of the Dissertation

In chpters two and three, we consider the problem of computing forecast horizons in dynamic decesion problems. Specifically we consider the Dynamic Lot-size problems introduced by Wagner and Whitin [103]. In chapter two, we investigate the practicality and usefulness of integer programming in computing forecast horizons. We formulate well known necessary, and necessary and sufficient conditions given in [68] and [21], as integer programming models. Also we introduce a new weak forecast horizon; forecast horizon subject to discrete future demand. In Chapter three, discrete forecast horizon is analyzed more thoroughly. A column generation based heuristic is introduced to the problem of traffic grooming in optical networks is discussed in Chapter four. Also a Lagrangian relaxation based upper bound procedure in used to compare the heuristic solution. In chapter five, we study the problem of flowshop scheduling under material handling constraint. In this chapter, a number of heuristics is proposed to solve this problem. Further analysis of this problem is done by using a hybrid-Genetic Algorithm (GA) search method. Finally chapter six concludes the study. The chapter also outline the future research directions.

CHAPTER 2

INTEGER PROGRAMS FOR COMPUTING MINIMAL FORECAST HORIZONS

2.1 Synopsis

In this chapter, we develop integer programs (IP) for computing minimal forecast horizons for the classical dynamic lot-sizing problem (DLS). The research community has thus far failed to evaluate the usefulness of Integer programming as a viable solution approach in this area. We feel that the modelling and structural advantages of the IP approach coupled with the recent significant developments in computational integer programming make for a strong case for its use in practice. Our objectives for this chapter are three fold: (i) Formulate the well-known Lundin and Morton [68] sufficiency conditions and characterizations (i.e., necessary and sufficient conditions) for forecast horizons as feasibility/optimality questions in 0-1 mixed integer programs. (ii) Develop integer pogramming based approach for evaluating the necessary and sufficient conditions of Chand and Morton [22]; and (iii) Through the use of an extensive computational study, establish the effectivness of IP-based approaches. We start by formulating the DLS problem.

T	:	the problem horizon, i.e., the number of periods.
k	:	setup cost for production.
h	:	holding cost per unit per period.
d_j	:	demand in period j, $j = 1, 2, ..., T$.
Q_j	:	quantity produced in period j, $j = 1, 2, ..., T$.
X_j	:	$\begin{cases} 1 & \text{if a setup is required in period } j; \\ 0 & \text{otherwise; } j = 1, 2, ..., T. \end{cases}$
I_j	:	inventory at the end of period j, $j = 1, 2, ..., T$.

Problem DLS-T

$$\text{Minimize } \sum_{j=1}^{T} [kX_j + hI_j]$$

subject to:

$$\begin{aligned} Q_j &\leq (\sum_{r=j}^{T} d_r)X_j \; 1 \leq j \leq T \\ I_j &= I_{j-1} + Q_j - d_j \; 1 \leq j \leq T \\ I_0, I_T &= 0 \\ I_j, Q_j &\geq 0 \\ X_j &\in \{0, 1\} \end{aligned}$$

2.2 Forecast Horizons for Dynamic Lot-Sizing

Definition: Let $(Q_1^F)^*$ be the first period production quantity in an optimum solution of an F-period problem with d_i as the demand for period i, $i = 1, 2, ..., F$. Period F is a *forecast horizon* if for every $N > F$ and all vectors of demands d_i, $i = F + 1, ..., N$, there exists at least one optimal solution with $(Q_1^F)^*$ as the first period production quantity. The number of periods of demand covered by $(Q_1^F)^*$ is the *decision horizon*.

2.3 Sufficiency Conditions (Lundin and Morton [68])

First, we present an IP formulation that identifies a planning horizon that satisfies the following sufficient conditions proved in Lundin and Morton [68].

A period F is a forecast horizon if an optimal solution with first period production quantity, say, $(Q_1^F)^*$ to the F-period problem that has its last setup in period L satisfies the following condition: every S-period problem with $L-1 \le S \le F$ has an optimal solution with first period production quantity equal to $(Q_1^F)^*$.

The period L is called the last *production point*; period $L-1$, at the end of which inventory is zero, is called the last *regeneration point.* The set of periods from $L-1$ to F is called the *regeneration set.* Wagner and Whitin [103] had shown earlier that any N-period problem with $N \ge F+1$ has a production point in $L,\ldots,F+1$. Thus, if every S-period problem with $L-1 \le S \le F$ has an optimal solution with first period production quantity equal to $(Q_1^F)^*$, then it must be that $(Q_1^F)^*$ is an optimal first period production for every N-period problem, $N \ge F+1$.

We formulate an integer program that determines whether a given time horizon T satisfies the sufficiency conditions above. In order to do this, note that we need predetermined optimal costs for problems with planning horizons ranging from 1 to T. We use the following additional notation.

Q_j^t = Quantity produced in period j in a t-period problem.

$$X_j^t = \begin{cases} 1 & \text{if a setup is required in period } j \text{ for a } t\text{-period problem;} \\ 0 & \text{otherwise.} \end{cases}$$

I_j^t = Inventory carried over from period j to $j+1$ in a t-period problem.

$$R_j^T = \begin{cases} 1 & \text{if period } j \text{ is in the regeneration set of the } T\text{-period problem;} \\ 0 & \text{otherwise.} \end{cases}$$

M_t = The optimal cost for a t-period problem.

Problem DLS-T-SC: *Find a feasible solution to the following set of constraints*

$$\sum_{j=1}^{t} kX_j^t + \sum_{j=1}^{t} hI_j^t = M_t \qquad 1 \le t \le T \tag{2.1}$$

$$(\sum_{r=j}^{t} d_r)X_j^t \ge Q_j^t \qquad 1 \le j \le t, 1 \le t \le T \tag{2.2}$$

$$I_j^t + Q_{j+1}^t - d_{j+1}^t - I_{j+1}^t = 0 \qquad 0 \le j \le t-1, 1 \le t \le T \tag{2.3}$$

$$R_T^T = 1 \tag{2.4}$$

$$R_{T-1}^T = 1 \tag{2.5}$$

$$1 - R_j^T \le \frac{1}{T} \sum_{r=j+2}^{T} X_r^T \qquad 1 \le j \le T-2 \tag{2.6}$$

$$Q_1^t - Q_1^{t+1} \le (\sum_{r=1}^{T} d_r)(1 - R_t^T) \qquad 1 \le t \le T-1 \tag{2.7}$$

$$Q_1^t - Q_1^{t+1} \ge -(\sum_{r=1}^{T} d_r)(1 - R_t^T) \qquad 1 \le t \le T-1 \tag{2.8}$$

$$I_0^t = 0$$

$$I_t^t = 0$$

$$I_j^t, Q_j^t, X_j^t \ge 0$$

$$X_j^T, R_j^T \in \{0, 1\}$$

Constraints (2.1), (2.2), and (2.3) ensure that only optimal solutions are considered. The optimal costs $M_t, t = 1, ..., T$, are obtained by solving the corresponding t-period standard DLS problem (Section 2.2). Constraints (2.4) and (2.5) ensure that periods T and $T-1$ are always in the regeneration set of the T-period problem. Constraint (2.6) determines whether a period, other than T and $T-1$, is in the regeneration set. Constraints (2.7) and (2.8) enforce that, for a forecast horizon, the first period production quantities are all equal for planning horizons in the regeneration set.

Next, we formulate an integer program for identifying the smallest planning horizon that satisfies these sufficiency conditions for a forecast horizon. We assume that an upper bound, denoted by T^u, on the length of a forecast horizon is known. We define the following additional notation:

$$R_j^t = \begin{cases} 1 & \text{if period } j \text{ is in the regeneration set of the } t\text{-period problem;} \\ 0 & \text{otherwise.} \end{cases}$$

$$W_t = \begin{cases} 1 & \text{if } t \text{ is the minimal forecast horizon;} \\ 0 & \text{otherwise.} \end{cases}$$

$$Y_j^t = \begin{cases} 1 & \text{if } t \text{ is the minimal forecast horizon and } j \text{ belongs to the} \\ & \text{regeneration set of the } t\text{-period problem;} \\ 0 & \text{otherwise.} \end{cases}$$

Problem DLS-MT-SC: *Find an optimal solution to the following integer program*

$$\text{Minimize} \sum_{k=1}^{T^u} kW_k$$

subject to:

$$\sum_{j=1}^{t} kX_j^t + \sum_{j=1}^{t} hI_j^t = M_t \qquad 1 \le t \le T^u \tag{2.9}$$

$$(\sum_{r=j}^{t} d_r)X_j^t \ge Q_j^t \qquad 1 \le j \le t, 1 \le t \le T^u \tag{2.10}$$

$$I_j^t + Q_{j+1}^t - d_{j+1}^t - I_{j+1}^t = 0 \qquad 0 \le j \le t-1, 1 \le t \le T^u \tag{2.11}$$

$$R_t^t = 1 \qquad 1 \le t \le T^u \tag{2.12}$$

$$R_{t-1}^t = 1 \qquad 1 \le t \le T^u \tag{2.13}$$

$$1 - R_j^t \le \frac{1}{T^u} \sum_{r=j+2}^{t} X_r^t \qquad 1 \le j \le t-2 \tag{2.14}$$

$$\sum_{t=1}^{T^u} W_t = 1 \tag{2.15}$$

$$Y_j^t \le 0.5R_j^t + 0.5W_t \qquad 1 \le j \le t, 1 \le t \le T^u \tag{2.16}$$

$$Y_j^t \ge R_j^t + W_t - 1 \qquad 1 \le j \le t, 1 \le t \le T^u \tag{2.17}$$

$$Q_1^t - Q_1^{t+1} \le (\sum_{r=1}^{T^u} d_r)(1 - Y_j^t) \qquad 1 \le j \le t, 1 \le t \le T^u - 1 \tag{2.18}$$

$$Q_1^t - Q_1^{t+1} \ge -(\sum_{r=1}^{T^u} d_r)(1 - Y_j^t) \qquad 1 \le j \le t, 1 \le t \le T^u - 1 \tag{2.19}$$

$$I_0^t = 0 \qquad 1 \le t \le T^u$$

$$I_t^t = 0 \qquad 1 \le t \le T^u$$

$$I_j^t, Q_j^t, X_j^t \ge 0$$

$$X_j^t, R_j^t, Y_j^t \in \{0, 1\}$$

The objective function minimizes the length of the forecast horizon. Constraints (2.9)-(2.14) play the same role they did in the previous formulation. Constraint (2.15) ensures that only one planning horizon is chosen as a forecast horizon. Constraints (2.16) and (2.17) are a linearization of the constraint $Y_j^t = R_j^t W_t$, and enforce that it is only necessary to evaluate regeneration sets of the planning horizon corresponding to the minimum forecast horizon. Constraints (2.18) and (2.19) check whether all the members in the regeneration set have the same optimal first period production quantity.

2.4 Necessary and Sufficiency Conditions (Chand and Morton [22])

Chand and Morton [22] were the first to develop a procedure for determining minimal forecast horizons. Their approach was designed around minimizing the regeneration set defined in Lundin and Morton [68]. In [68], the regeneration set was defined to consist of all the periods from the period before the last production point to the final period (i.e., periods $L(T)-1$ to T). Chand and Morton recognized that the actual regeneration set (i.e, the set of periods that could potentially be regeneration points in a longer horizon problem) could actually be much smaller than that defined in [68]. They defined the concept of a minimal regeneration set and developed a procedure for identifying it. Once the minimal regeneration set has been identified, it only remains to test whether all the planning horizons in it have a common optimal first period production quantity. The following four steps follow their procedure. Steps 1-3 identify the minimal regeneration set, which we denote by $R(T)$; Step 4 checks if all the planning horizons in set $R(T)$ have a common first period production quantity for at least one optimal solution. We begin by initializing set $R(T)=\{\emptyset\}$.

Step 1: Let M_T^s be the optimal cost of a T-period problem with the additional restriction that period $s+1$ is the last production point. This cost can be computed using the following integer program.

Problem: DLS-T-MTS

$$
\begin{aligned}
\text{Minimize} \sum_{j=1}^{T} (kX_j + hI_j) & \\
\textit{subject to:} & \\
(\sum_{r=j}^{T} d_r)X_j \geq Q_j & \qquad 1 \leq j \leq T \\
I_j + Q_{j+1} - d_{j+1} - I_{j+1} = 0 & \qquad 0 \leq j \leq T-1 \\
X_{s+1} = 1 & \qquad (2.20) \\
\sum_{r=s+2}^{T} X_r = 0 & \qquad (2.21) \\
I_0 = 0 & \\
I_T = 0 & \\
I_j, Q_j \geq 0 & \qquad 1 \leq j \leq T \\
X_j \in \{0,1\} & \qquad 1 \leq j \leq T
\end{aligned}
$$

The formulation is similar to Problem DLS-T (Section 2.2), except for Constraints (2.20) and (2.21). These two conditions ensure that the last setup is in period $s+1$. Thus, period $s+1$ is the last production point; the objective function is the lowest cost for the T-period problem with the additional restriction that $s+1$ is the last production point.

Step 2: Identify the maximum period s such that $(s+1)$ is the last production point (i.e., s corresponds to the the maximum value of $L(T)-1$ over all optimal solutions) for a T-period problem using the following integer program. We define the following.

$$
R_s = \begin{cases} 1 & \text{if period } s+1 \text{ is the maximum last production point over all} \\ & \text{optimum solutions of the } T\text{-period problem;} \\ 0 & \text{otherwise.} \end{cases}
$$

Problem: DLS-T-MLT

$$\text{Maximize} \sum_{s=0}^{T-1} sR_s$$

$$\textit{subject to:}$$

$$M_T \geq M_T^s R_s \qquad 0 \leq s \leq T-1 \tag{2.22}$$

$$\sum_{s=0}^{T-1} R_s = 1 \tag{2.23}$$

$$R_s \in \{0,1\} \qquad 0 \leq s \leq T-1$$

Note that $M_T \leq M_T^s, s = 0, ..., T-1$. The reverse constraints (2.22) along with the Constraint (2.23) and the objective function ensure the selection of only that optimal solution (of the T-period problem) whose last regeneration point is maximum. It is easy to see that Problem DLS-T-MLT is always feasible; let $l^*(T)$ denote its optimum solution value, and $R(T) = \{l^*(T)\}$.

Step 3: Compute the ratios $R_{l^*}^q$ and $R_{l^*}^T$ as follows:

$$R_{l^*}^q = \frac{M_T^q - M_T^{l^*}}{(q - l^*(T))h} \quad l^*(T) < q \leq T-1$$

$$R_{l^*}^T = \frac{M_T + k - M_T^{l^*}}{(T - l^*(T))h}$$

Find v, the largest period that satisfies the condition:

$$R_{l^*}^v = \min_{q \in \{l^*(T)+1, \ldots T\}} R_{l^*}^q$$

Include v in the regeneration set, i.e., $R(T) = R(T) \cup \{v\}$. If $v \leq T-1$, then set $l^*(T) = v$ and go back to Step 3. Otherwise add T to the regeneration set, i.e., $R(T) = R(T) \cup \{T\}$ and proceed to Step 4.

Step 4: Check whether all the planning horizons in the minimal regeneration set $R(T)$ have a common first period production quantity in at least one optimal solution.

Problem DLS-T-MRT: *Find a feasible solution to the following set of constraints*

$$\begin{aligned}
\sum_{j=1}^{t} kX_j^t + \sum_{j=1}^{t} hI_j^t = M_t \qquad & t \in R(T) \\
(\sum_{r=j}^{t} d_r)X_j^t \geq Q_j^t \qquad & 1 \leq j \leq t,\ t \in R(T) \\
I_j^t + Q_{j+1}^t - d_{j+1} - I_{j+1}^t = 0 \qquad & 0 \leq j \leq t-1,\ t \in R(T) \\
Q_1^t - Q_1^s = 0 \qquad & s, t \in R(T), s \neq t \qquad (2.24) \\
I_0^t = 0 \qquad & t \in R(T) \\
I_t^t = 0 \qquad & t \in R(T) \\
I_j^t, Q_j^t \geq 0 \qquad & 1 \leq j \leq t,\ t \in R(T) \\
X_j^t \in \{0,1\} \qquad & 1 \leq j \leq t,\ t \in R(T)
\end{aligned}$$

The first three sets of constraints are similar to those in Problem DLS-T-SC (Section 2.3), except that these constraints are imposed only on the planning horizons in the minimal regeneration set $R(T)$. Constraints (2.24) check for the existence of a common optimal first period production quantity for all the planning horizons in the minimal regeneration set.

For a given value of T, Steps 1-4 establish whether or not T is a forecast horizon. The minimal forecast horizon is the smallest value of T, $1 \leq T \leq T^u$ for which Problem DLS-T-MRT above has a feasible solution.

2.5 Computational Results

Our main purpose in implementing the formulations discussed in this chapter is to test the practicability of the approach for realistic problems. For the dynamic lot-sizing problem, the literature on computational results of forecast horizon procedures is limited. Chand et al. [24] report a graphical illustration of minimal forecast horizons for the unit demand case. For non-stationary demand, Chand and Morton [22] provide results for several data patterns and illustrate the benefit of computing minimal forecast horizons by comparing with results from an earlier report of Lundin and

Morton [68]. These studies do not report details such as the computing platforms and the computing times. However, since our intention is to demonstrate an alternative approach and not to compare based on measures such as computing times, the description of the test bed used in these papers is sufficient for our purpose.

The integer programming formulations were solved using the MINTO (version 8.1) programming library [79]. All the computations were carried out on a Pentium IV computer (2.4 GHz, 512 MB RAM) running Red Hat Linux 7.2.

2.5.1 Stationary demand

Our setup is the same as that in Chand et al. [24]: unit demand in each period; the holding cost, $h = 1$; the setup cost, k, varies from 10 to 25 in unit increments.

For stationary demand, the minimum forecast horizons and the corresponding computation times are provided in the the second and third columns of Table 2.5.1. As mentioned in Chand et al. [24], the relationship between the setup cost and the minimal forecast horizon is difficult to characterize precisely. As a function of the setup cost k, the minimum forecast horizon has local peaks where it equals approximately twice the setup cost; between two adjacent peaks, corresponding to setup costs of, say, k_1 and k_2, the minimum forecast horizon is smallest at approximately the average setup cost $\frac{k_1+k_2}{2}$.

2.5.2 Non-stationary demand

We use the setup designed by Chand and Morton [22]. Here, the demands are manipulated over two dimensions. The first one is the rate of growth of the demands. In order to capture flat, increasing, and decreasing natures of the demand, the function $D_{t+1} = D_0(G)^t$ is used to compute the mean demand in each period. The parameter G captures the character of the growth in demand over time. Three values of this growth parameter are considered: 1.000 (flat), 1.005 (increasing), and 0.995 (decreasing). Once the mean is fixed, the actual demand in a period is generated using

Table 2.1: Minimum Forecast Horizons and the required computational times for unit initial demand.

k	$F_{min}(k)$	Time (sec.)
10	20	1.01
11	17	0.92
12	13	0.86
13	18	1.03
14	24	1.73
15	30	34.58
16	26	7.39
17	21	1.16
18	21	1.42
19	28	8.53
20	35	216.63
21	42	402.11
22	37	245.83
23	31	179.32
24	25	10.45
25	32	203.71

the function, $D_{t+1} = D_0(G)^t + SD_0\xi$, where ξ is the standard Normal variate. The parameter S is a measure of the variability associated with the demand takes three values: 0.15, 0.50, and 1.15. At any time, if a demand generated is less than zero, it is set to zero. We thus have nine possible combinations (three values of G and three values of S) of the two demand-related parameters. The base demand rate, D_0 was set to 10, holding cost h was set to 1. The setup cost k was allowed to take values 8 values: 10, 15, 20, 30, 50, 75, 100, and 150. For each value of k and each combination of the 9 demand parameters, we generated 11 instances. The total number of instances in this test bed is, therefore, 9 x 8 x 11 = 792.

For non-stationary data, the minimum, maximum, and median minimal forecast horizons along with the computation times (in seconds) are reported in Tables 2.5.2-2.5.2. Figure 3.3 plots the relationship between the setup cost and the median forecast horizons from two different procedures: (i) the smallest forecast horizon satisfying the sufficiency conditions of Section 2.3 and (ii) the minimal forecast horizon (Section 3.2). For each value of the setup cost, the 11 instances with $G = 1.000$ and $S = 0.15$ are used for the plot. It is interesting to note that the smallest forecast horizons satisfying the sufficiency conditions of Section 2.3 are reasonably close to the minimal forecast horizons. Since the computational effort for verifying the sufficiency conditions is lesser than that for the necessary and sufficient conditions, the sufficiency conditions could be used for a reasonable and quick approximation of minimal forecast horizons.

Table 2.2: Minimum forecast horizons for non-stationary demand: flat demand pattern with the growth parameter $G = 1.000$.

k	S=0.15				S=0.50				S=1.15			
	Min. Forecast Horizons				Min. Forecast Horizons				Min. Forecast Horizons			
	Min.	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**
10	2	6	3	0.56	2	6	3	0.21	2	7	3	0.16
15	9	14	8	1.02	2	9	5	0.92	2	8	4	0.73
20	4	23	11	1.11	2	9	4	1.03	2	7	4	0.89
30	3	20	13	1.78	6	15	8	1.52	2	13	6	1.32
50	11	27	19	2.42	5	21	17	1.96	8	14	6	1.61
75	12	24	15	3.52	7	22	15	2.51	6	11	8	1.94
100	15	25	21	35.62	9	17	14	14.24	5	16	9	22.12
150	20	35	28	142.51	16	32	24	125.71	7	28	16	122.16

Table 2.3: Minimum forecast horizons for non-stationary demand: increasing demand pattern with the growth parameter $G = 1.005$.

k	S=0.15				S=0.50				S=1.15			
	Min. Forecast Horizons				Min. Forecast Horizons				Min. Forecast Horizons			
	Min.	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**
10	2	8	5	0.83	2	5	3	0.61	2	6	4	0.43
15	3	15	7	0.99	2	5	3	0.78	2	6	4	0.59
20	3	11	8	1.31	2	8	5	1.04	2	7	5	0.81
30	5	14	6	2.17	6	19	14	1.41	4	12	7	0.95
50	6	19	12	3.12	5	23	16	1.94	5	11	8	1.48
75	13	29	17	6.23	8	25	15	5.12	4	15	6	3.82
100	16	26	21	121.52	9	22	16	103.42	3	17	13	89.41
150	18	31	25	452.92	12	27	19	326.12	2	23	16	173.4

Figure 3.4 is a plot of the median forecast horizon as a function of the setup cost for the three values of S (0.15, 0.50, 1.15) and G set equal to 1.005. For each of the three values of S, the median minimal forecast horizon typically increases with the setup cost. For a given setup cost, the median forecast horizon is typically lower for higher values of S. While a precise mathematical explanation seems difficult to prove, we offer the following intuitive explanation: a higher value of the demand variability parameter S indicates a larger fluctuation in the demand thereby increasing

Table 2.4: Minimum forecast horizons for non-stationary demand:decreasing demand pattern with the growth parameter G set to 0.995.

k	S=0.15				S=0.50				S=1.15			
	Min. Forecast Horizons				Min. Forecast Horizons				Min. Forecast Horizons			
	Min.	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**
10	4	8	5	1.13	2	6	4	1.04	4	7	6	0.82
15	4	14	8	1.52	2	7	5	1.31	2	6	5	0.93
20	6	16	12	1.89	4	12	6	1.42	7	9	6	1.18
30	5	15	10	2.41	7	13	8	1.72	8	13	10	1.39
50	8	25	17	6.27	7	21	14	2.18	6	24	17	1.92
75	14	28	18	22.17	6	22	17	3.62	13	25	16	2.04
100	18	34	24	153.82	12	28	25	24.73	14	23	18	11.04
150	23	36	27	843.23	10	35	25	632.68	10	28	24	428.22

Table 2.5: Forecast horizons from two different procedures: (i) the smallest forecast horizon satisfying the sufficiency conditions of Section 2.3 and (ii) the minimal forecast horizon (Section 3.2).

k	Smallest Forecast Horizon Satisfying Sufficient Conditions				Min. Forecast Horizon			
	Min.	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**
10	2	4	3	0.11	2	4	3	0.05
15	3	13	6	0.03	3	13	6	0.64
20	3	17	8	0.06	3	17	7	1.34
30	7	23	11	1.11	4	18	9	2.29
50	10	20	16	2.84	8	20	12	3.18
75	12	23	19	3.06	12	23	15	3.94
100	18	22	19	2.92	15	22	17	35.32
150	22	34	28	55.39	16	32	24	192.44

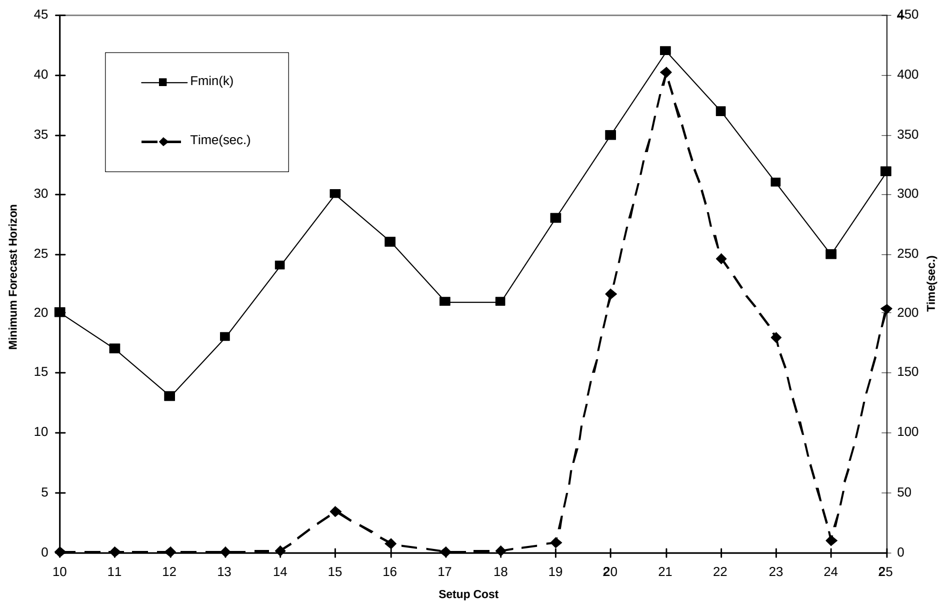

Figure 2.1: Relationship between the setup cost and the median forecast horizons from two different procedures: (i) the smallest forecast horizon satisfying the sufficiency conditions of Section 2.3 and (ii) the minimal forecast horizon (Section 3.2).

the probability of a large demand, and consequently reducing the forecast horizon. With S fixed at 0.50, Figure 3.5 presents the relationship between the median minimal forecast horizon and the setup cost for the three values of the growth parameter, G: 0.995, 1.000, and 1.005. As with Figure 3.4, the minimal forecast horizons typically increase with the setup cost. The other observation is that the median forecast horizon is larger for smaller values of G. When G is large, the demands get increasingly larger in the later periods and as a result, there is a better chance of a smaller forecast horizon.

Before concluding this section, it is useful to highlight managerial insights and guidelines for practitioners gained from our computational results:

- Integer programming offers a practicable and easily generalizable alternative approach for computing forecast horizons. Most state-of-the-art integer programming solvers can be applied directly without any significant customization.

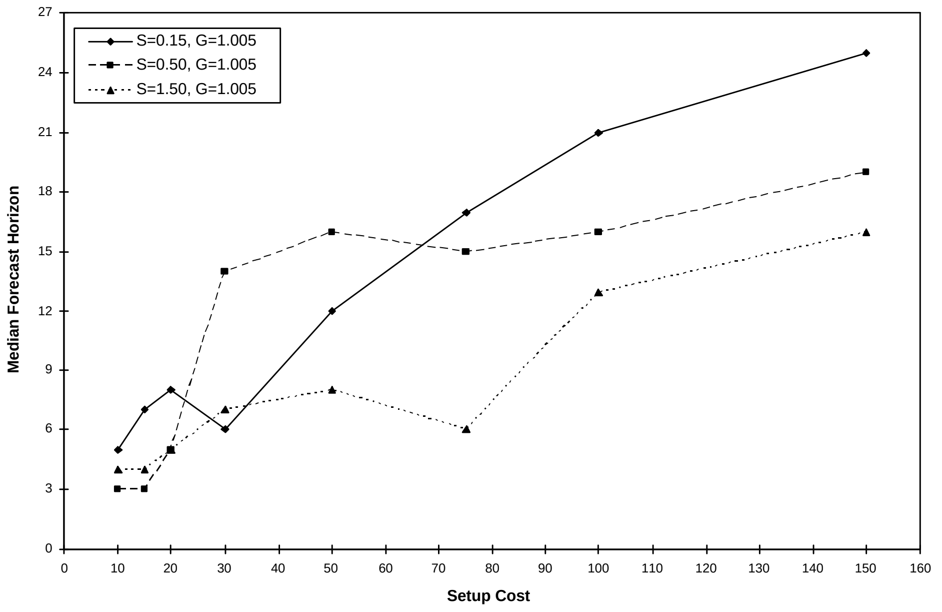

Figure 2.2: The median forecast horizon as a function of setup cost and demand variability.

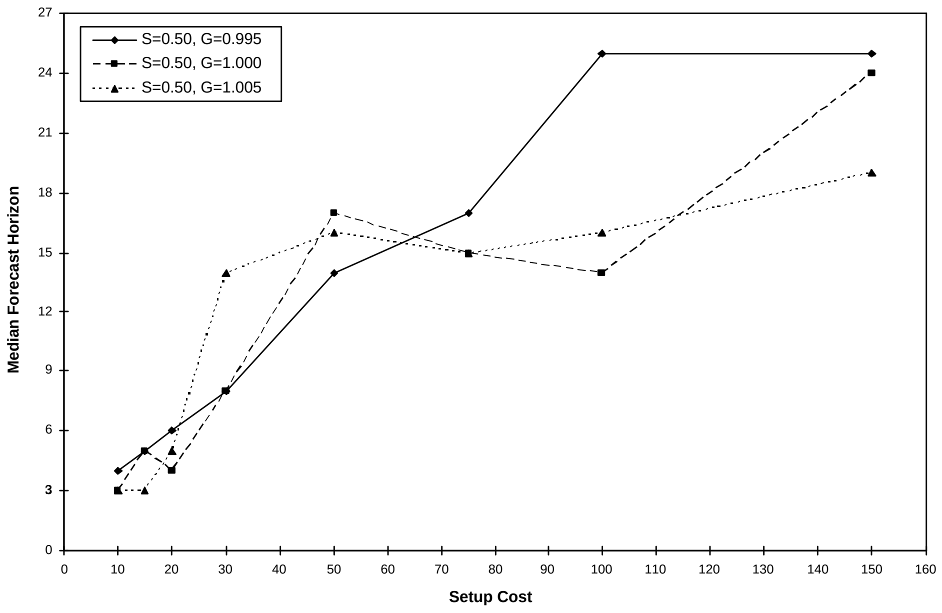

Figure 2.3: The median forecast horizon as a function of setup cost and demand growth.

- For the dynamic lot-sizing problem, the sufficiency conditions of Lundin and Morton [68] seem to offer an acceptable approximation of minimal forecast horizons and involve relatively low computational effort as compared to that involved in verifying the exact characterization of Chand and Morton [22]. The sufficiency conditions can, therefore, be used for a quick estimate of the minimal forecast horizon.

- For the dynamic lot-sizing problem, the length of a minimal forecast horizon for a fixed holding cost typically (i) increases with the setup cost, (ii) decreases with the variability in the demand, and (iii) decreases as the rate of growth of demand increases over time.

2.6 Concluding Remarks

Our main purpose in this chapter was to demonstrate integer programming as a viable option for computing minimal forecast horizons for dynamic decision problems. To this end, we outline our approach using the dynamic lot-sizing problem. We formulate two well-known results, a sufficient condition for an integer to be a forecast horizon and a necessary and sufficient condition for an integer to be a minimal forecast horizon, as feasibility/optimality problems in mixed 0-1 programs. Numerical results on a large number of instances with both stationary and non-stationary demand, demonstrate the solvability of the formulations with a modest amount of computing effort.

In the next chapter, we continue the discussion of computing forecast horizons. We introduce and explore the structural properties of a new type of *weak* forecast horizons.

CHAPTER 3

DISCRETE FORECAST HORIZONS: ANALYSIS AND COMPUTATIONS

3.1 Synopsis

In this chapter, from a computational point of view, we emphasize our use of 0-1 mixed integer programming. While the forecast horizon literature is extensive, the use of integer programming to compute forecast horizons has been limited. The recent significant developments in computational integer programming coupled with the modeling and structural simplicity of the integer programming approach make a strong case for its use in computing forecast horizons. Additionally, the structured and constraint-based nature of integer programming makes it easy to incorporate additional constraints and to address multi-product variants of the DLS problem. We illustrate this simplicity on two significant variants of the classical DLS problem – a capacity-constrained warehouse scenario and a two product DLS problem with joint and individual fixed costs – and demonstrate the computation of discrete forecast horizons for these variants. Here, it it worth noting that most existing procedures to identify minimal forecast horizons depend heavily on the structural properties of the problem being studied. Therefore, the extension of such procedures to accommodate significant variants is typically difficult. Our characterization of discrete forecast horizons, together with the integer programming approach allows us to address new and important variants in a relatively straightforward manner. Thus, we believe that our approach has the potential to significantly expand the domains over which forecast horizon concepts can be investigated.

Before we proceed with the details of our analysis of DLS models with discrete future demands, we summarize the major contributions of this paper:

1. For a dynamic decision problem, we introduce the concept of a forecast horizon in the presence of discrete future demand. Specifically, we define FH^q as a forecast horizon assuming future demands to be an integer multiple of some number $q \in \Re_+$. The classical forecast horizon, which does not place any restriction on future demand, is denoted by FH^0. The minimal forecast horizons under these two cases (i.e., with and without the discretization) are denoted by FH^q_{min} and FH^0_{min}, respectively. Clearly, $\mathrm{FH}^q_{min} \leq \mathrm{FH}^0_{min}$. To illustrate the structural connections between a discrete and continuous forecast horizon, we prove several conditions under which FH^q equals FH^0. We also examine the cost implications of using FH^q_{min} as an alternative for FH^0_{min}.

2. We characterize a necessary and sufficient condition for an integer T to be a minimal forecast horizon when the future demands are discrete. These conditions are a significant relaxation from those detailed in [22], and are widely applicable due to their non-reliance on specific structural properties.

3. We demonstrate the feasibility and effectiveness of integer programming (IP) as an approach for computing discrete forecast horizons. Using a detailed computational study, we illustrate the impact of assuming discrete demand, and the usefulness of the theoretical properties developed in this Chapter.

4. We illustrate, via a representative computational study, the easy adaptability of the integer programming approach to two variants of the dynamic lot-size model under discrete future demand. To the best of our knowledge, these are the first known forecast horizon computation procedures for the two variants we discuss.

Before considering the possibility of discrete demands, we summarize some important forecast horizon results developed with the assumption of continuous demands. Wagner and Whitin [103] showed that if period T is a production period in a T-period problem, $T \geq 2$, then T is a forecast horizon. Let period L be the last production period in an optimal solution to a T-period problem. Lundin and Morton [68] extended that result to say that T is a forecast horizon if all the problems with horizon $L-1, L, L+1, \ldots, T$ have a common optimal first-period production quantity. The set $\{L-1, L, L+1, \ldots, T\}$ is referred to as the *regeneration set.* Chand and Morton [22] proposed a characterization for a minimal regeneration set, developed a procedure for identifying it, and proved the following necessary and sufficient condition: T *is a forecast horizon if and only if every problem horizon in the minimal regeneration set has a common optimal first-period production quantity.* This stream of analysis did not, in any manner, restrict the possible values of future demands.

The rest of this chapter is organized as follows. Section 3.2 formally introduces the concept of an FH^q, and develops the necessary and sufficient conditions for a problem horizon T to be an FH^q. Section 3.3 discusses conditions under which an FH^q implies an FH^0. We also analyze the impact on cost of the discreteness assumption. Section 3.3.3 discusses the results of a detailed computational study aimed at accessing the usefulness of the IP approach, and quantifies the impact of the discreteness assumption. Discrete forecast horizons for two variants of the dynamic lot-size model are discussed in Section 3.4. Section 3.5 provides a summary of the chapter.

3.2 Forecast Horizons for Discrete Demands

Imposing discreteness on the future demands is essentially an attempt to exploit practical restrictions on the values of potential demands. Forecast horizons obtained by incorporating such additional restrictions are known as *weak forecast horizons* (see [17]). This section investigates an important type of weak minimal forecast horizon – a minimal forecast horizon assuming future demands to be integer multiples of q where $q \in \Re^+$.

There are two important reasons for the study of FH^q. First, it is typical for practical demand realizations to obey significant and known restrictions on the future demands. The most elementary restriction is the granularity with which these demands are measured. Most businesses have a basic minimal unit for measuring demand.

For example, an oil refinery may quantify demand in the thousands of gallons, while demand is integral for a tire manufacturer. In both these cases, the business is not interested in considering demands that fall in between these discrete values. Thus, a discrete forecast horizon suffices in these cases and there is no need to consider a traditional forecast horizon. This seemingly minor restriction on future demands has the ability to significantly reduce the length of a minimal forecast horizon, and consequently reduce forecasting expense. Second, such a discretization in the future demands allows us to develop new necessary and sufficient conditions for minimal forecast horizons; these conditions can be easily posed as optimality/feasibility problems for 0-1 integer programs. An important benefit of this approach is that it does not depend explicitly on the structural properties of the forecast horizon problem being studied. Consequently, it becomes relatively easy to extend the approach to investigate a wider class of forecast horizon problems.

In order to compute FH^q_{min}, we first need necessary and sufficient conditions for a period T to be an FH^q. As before, the demands in the first T periods are known and assumed to be $d_1, d_2, \ldots, d_T$, respectively. Let $\mathcal{Q}_1(\alpha)$ be the *set* of optimal first-period production quantities for a $(T+1)$-period problem with a demand of α for period $T+1$. Chand and Morton [21] prove the following result:

The optimal production plan in the first T periods of any N-period problem with $N \geq T+1$ can be found by solving the $(T+1)$-period problem with demands $d_1, d_2, \ldots, d_T$ in the first T periods and some appropriately selected demand $\alpha \in \Re_+$ in period $T+1$.

This result holds for arbitrary (i.e., nonnegative real-valued) future demands and it is more general than is needed here. For the discrete demand case, the only additional observation, which follows trivially, is that it is enough for α to be an integer multiple of q. Clearly, if $\alpha > \frac{k}{h}$, then $\mathcal{Q}_1(\alpha) = \mathcal{Q}_1(0)$. The necessary and sufficient conditions for T to be a forecast horizon can be stated as follows.

Theorem 3.2.1 *T is a forecast horizon assuming that future demands are integer multiples of q (i.e., T is an FH^q) if, and only if, $\cap_{\alpha \in \{0,q,2q,\ldots,\overline{n}q\}} \mathcal{Q}_1(\alpha) \neq \emptyset$, where $\overline{n}$ is* *the smallest integer such that* $\overline{n}q \geq \frac{k}{h}$.

Proof: T is clearly not an FH^q if $\cap_{\alpha \in \{0,q,2q,...,\overline{n}q\}} \mathcal{Q}_1(\alpha) = \emptyset$: there is no common optimal first-period decision for the demands $0, q, 2q, ..., \overline{n}q$ in period $T+1$. To prove sufficiency, assume that $\cap_{\alpha \in \{0,q,2q,...,\overline{n}q\}} \mathcal{Q}_1(\alpha) \neq \emptyset$. We know from [21] that every instance of an N-period problem with $N \geq T+1$ has the same first-period production as a $(T+1)$-period problem with an appropriately chosen $T+1$-period demand α. We consider two cases: (i) if $\alpha > \frac{k}{h}$, then $\mathcal{Q}_1(\alpha) = \mathcal{Q}_1(0)$ and the result follows immediately, and (ii) if $\alpha \leq \frac{k}{h}$, the result follows from the hypothesis. □

Given that T is an FH^q, let $\chi^q = \cap_{\alpha \in \{0,q,2q,...,\overline{n}q\}} \mathcal{Q}_1(\alpha) \neq \emptyset$. Then, χ^q is the set of first-period decisions guaranteed to be optimal for every N-period problem, $N \geq T+1$, and all vectors of demands $d_i \equiv 0 \pmod{q}, i = T+1, ..., N$. In order to formulate an integer program to test whether T is an FH^q, we use the following additional notation:

$M_{T+1}(\alpha)$	:	the optimal cost for a $(T+1)$-period problem when the demand in period $T+1$ is α. Note that $\alpha = 0$ is essentially the T-period problem.
Λ_q	:	the set of demands for period $T+1$, indexed by α, $\Lambda_q \in \{n \times q; n = 0, 1, 2, ...\overline{n}\}$.
X_j^α	:	$\begin{cases} 1 & \text{if a setup is required in period } j \text{ in the problem with period } T+1 \text{ demand } \alpha; \\ 0 & \text{otherwise.} \end{cases}$
Q_j^α	:	the quantity produced in period j in the problem with period $T+1$ demand α.
I_j^α	:	the inventory carried over from period j to $j+1$ in the problem with period $T+1$ demand α.

Problem DLS-T-NSC: *Find a feasible solution to the following set of constraints:*

$$\sum_{j=1}^{T+1} kX_j^\alpha + \sum_{j=1}^{T+1} hI_j^\alpha = M_{T+1}(\alpha) \qquad \alpha \in \Lambda_q \tag{3.1}$$

$$(\sum_{r=j}^{T+1} d_r)X_j^\alpha \geq Q_j^\alpha \qquad 1 \leq j \leq T+1,\ \alpha \in \Lambda_q \tag{3.2}$$

$$I_{j-1}^\alpha + Q_j^\alpha - d_j - I_j^\alpha = 0 \qquad 1 \leq j \leq T,\ \alpha \in \Lambda_q \tag{3.3}$$

$$I_{j-1}^\alpha + Q_j^\alpha - \alpha - I_j^\alpha = 0 \qquad j = T+1,\ \alpha \in \Lambda_q \tag{3.4}$$

$$Q_1^\alpha - Q_1^{\alpha'} = 0 \qquad \alpha \neq \alpha'; \alpha, \alpha' \in \Lambda_q \tag{3.5}$$

$$I_0^\alpha, I_{T+1}^\alpha = 0 \qquad \alpha \in \Lambda_q$$

$$I_j^\alpha, Q_j^\alpha \geq 0 \qquad 1 \leq j \leq T+1,\ \alpha \in \Lambda_q$$

$$X_j^\alpha \in \{0,1\} \qquad 1 \leq j \leq T+1,\ \alpha \in \Lambda_q$$

Constraints (4.3) ensure that only those solutions with cost equal to the optimal cost are considered. Constraints (4.5) enforce the standard DLS restraint that production cannot occur without a setup. Constraints (4.6) and (3.4) enforce the conservation of material flows. Constraints (3.5) force that all demand values, $\alpha \in \Lambda_q$, have a common optimal first-period production quantity.

The minimal forecast horizon FH_{min}^q is the smallest value of T for which the formulation above has a feasible solution.

Example 1: We illustrate the computation of discrete minimal forecast horizons FH_{min}^q for $q = 1, 1/2$ and $1/4$, and compare them with the continuous forecast horizon FH_{min}^0. The demand vector for the first 23 periods is

$$(14, 15, 12, 12, 12, 19, 5, 5, 18, 17, 13, 11, 7, 12, 6, 8, 15, 19, 7, 8, 11, 12, 12).$$

Figure 3.1 shows the minimal forecast horizons as the setup cost k varies from 10 to 60; without loss of generality, the holding cost $h = 1$. For $k = 20$, observe that FH_{min}^1 $< \mathrm{FH}_{min}^{1/2} < \mathrm{FH}_{min}^{1/4} = \mathrm{FH}_{min}^0$. Also, note that the discrete forecast horizon $\mathrm{FH}_{min}^{1/4}$ is identical to FH_{min}^0. □

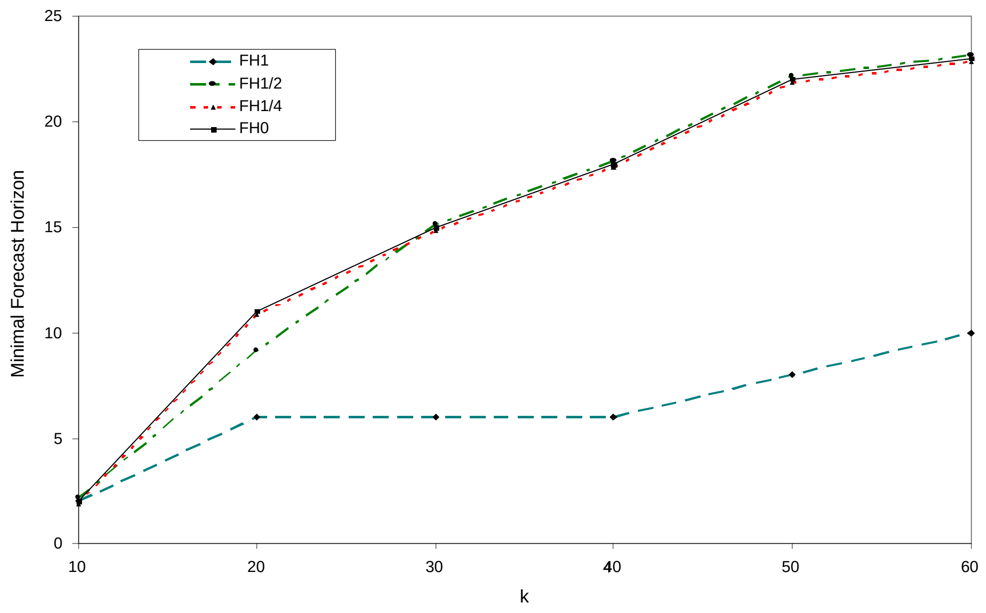

Figure 3.1: The impact of discretization on the length of the minimal forecast horizon as a function of the setup cost k. The initial demand data is as described in Example 1.

3.3 Relationship Between FH^q and FH^0

If T is an FH^0, then it follows that T is an FH^q for any $q > 0$. Therefore, we have FH$^q_{min} \leq$ FH$^0_{min}$. It is easy to see that the converse does not always hold; in Example 1 above, for $k = 20$, FH$^{1/2}_{min} <$ FH$^0_{min}$. Next, we examine conditions under which FH$^q_{min}$ $=$ FH$^0_{min}$. Intuitively, there are two considerations that enable us to derive such conditions: (i) the behavior of the optimal cost with respect to the granularity q of a discrete forecast horizon, and (ii) the sensitivity of the last regeneration period in an optimal solution to perturbations in future data. Next, we illustrate results under both cases. We start by defining additional notation required for the results in this section.

P(α)	:	$(T+1)$-period DLS problem with demand d_i in period $i, i=1,...,T$ and demand α in period $T+1$.
$C(Q_1=x,\alpha)$	:	optimal cost of P(α) with the first-period production fixed at x.
$C_l(\alpha)$	:	optimal cost of P(α) with the last regeneration period l.
$l(\alpha)$	:	last regeneration period of an optimal solution of P(α).
$L(\alpha)$	:	largest last regeneration period among all optimal solutions of P(α).
P$(\alpha, L(\alpha))$	:	Problem P(α) with the last regeneration period set to $L(\alpha)$.
$\mathcal{R}(\alpha)$	:	$\{(l(\alpha)+1, l(\alpha)+2, ..., L(\alpha+q)\}$.
$\Gamma(\alpha, j)$	:	$\frac{C_j(0)-C_{l(\alpha)}(0)}{(j-l(\alpha))h}, j \geq l(\alpha)+1$.
$\hat{\mathcal{Q}}_1(L(\alpha))$	:	set of optimal first-period production quantities for Problem P(α) with the last regeneration period set to $L(\alpha)$.
M_l	:	optimal cost of an l-period problem, $l \leq T+1$.
$\gamma(l, T+1, \alpha)$	:	cost of producing in period $l+1$, for periods $l+1, ..., T+1$ for P(α).

3.3.1 Sufficient conditions under which FH$^q \Rightarrow$ FH^0

For $\alpha \in \Lambda_q$, define $S_\alpha = \{d_1, d_1+d_2, ..., \sum_{i=1}^T d_i, \sum_{i=1}^T d_i + \alpha\}$. Then, S_α is the set of all possible first-period production quantities in an optimum solution to Problem P(α). Recall from Section 3 that $\chi^q = \cap_{\alpha\in\{0,q,2q,...,\overline{n}q\}} \mathcal{Q}_1(\alpha) \neq \emptyset$ if T is an FH^q.

Theorem 3.3.1 *Let period T be an FH^q. If*

$$\min_{x\in S_\alpha \setminus \chi^q} C(Q_1=x,\alpha) \geq M_{T+1}(\alpha+q)\ \forall\ \alpha \in \Lambda_q$$

then period T is an FH^0.

Proof: For Problem P$(\alpha+\delta), 0<\delta<q$, let an optimal first-period production be $x_1 \notin \chi^q$. Clearly, $x_1 \in S_{\alpha+\delta}$. Thus, $x_1 \in S_{\alpha+\delta} \backslash \chi^q$. Then,

$$M_{T+1}(\alpha+\delta)=C(Q_1=x_1, \alpha+\delta) \geq C(Q_1=x_1, \alpha).$$

Now, $C(Q_1=x_1, \alpha) \geq \min\limits_{x \in S_\alpha \backslash \chi^q} C(Q_1=x, \alpha) \geq M_{T+1}(\alpha+q)$. The last inequality follows from the hypothesis. Thus, if $x_1 \notin \chi^q$, then the cost of the optimal solution to Problem P$(\alpha+\delta), 0<\delta<q$, is at least as large as $M_{T+1}(\alpha+q)$. Further, an optimal solution to Problem P$(\alpha+q)$ is feasible for Problem P$(\alpha+\delta)$. Also, since T is an FH^q, there exists a solution of Problem P$(\alpha+q)$ with first-period production in χ^q. The result follows. $\square$

Theorem 3.3.1 is related to the following result in [103]: if period T is a production period in a T-period problem, $T \geq 2$, then T is a forecast horizon. We offer the following explanation: for any $(T+1)$-period problem P$(\alpha+\delta), 0<\delta \leq q, \alpha \in \Lambda_q$, if an optimal first-period production $x_1 \notin \chi^q$, then the proof of Theorem 3.3.1 shows that $M_{T+1}(\alpha+\delta) \geq M_{T+1}(\alpha+q)$. The reverse inequality, $M_{T+1}(\alpha+\delta) \leq M_{T+1}(\alpha+q)$, follows trivially. Thus, $M_{T+1}(\alpha+\delta)=M_{T+1}(\alpha+q)$. This can happen if, and only if, period $T+1$ is a production period. Thus, we have a situation where period $T+1$ is a production period in a $(T+1)$-period problem. We can, therefore, conclude that $T+1$ is a forecast horizon. Since T is an FH^q, the additional knowledge that the T-period problem has an optimal first period production quantity in χ^q helps reduce the length of the forecast horizon to T.

The following result was used in [22] without a formal proof. Since we could not locate a proof in the literature, we provide it here for completeness.

Theorem 3.3.2 *Consider two problem instances $P(\alpha_1)$ and $P(\alpha_2)$ with $\alpha_2>\alpha_1$ and holding cost $h>0$. Given an optimal solution to Problem $P(\alpha_1)$ with last regeneration period $l(\alpha_1)$, there exists an optimal solution to Problem $P(\alpha_2)$ with a last regeneration period $l(\alpha_2) \geq l(\alpha_1)$.*

Proof: The optimal costs of Problems P(α_1) and P(α_2) can be stated as follows:

$$M_{T+1}(\alpha_1) = M_{l(\alpha_1)} + \gamma(l(\alpha_1), T+1, \alpha_1) \tag{3.6}$$

$$M_{T+1}(\alpha_2) = M_{l(\alpha_2)} + \gamma(l(\alpha_2), T+1, \alpha_2) \tag{3.7}$$

A feasible solution to P(α_1) can be obtained from an optimal solution to Problem P(α_2) by producing α_1 units, instead of α_2, in the last period. Let the cost of this solution be $C'(\alpha_1)$. Then,

$$C'(\alpha_1) = M_{l(\alpha_2)} + \gamma(l(\alpha_2), T+1, \alpha_1) \tag{3.8}$$

Similarly, a feasible solution to P(α_2) can be obtained by substituting the optimal solution for problem P(α_1) and producing α_2 units, instead of α_1, in the last period. Let the cost of this solution be $C'(\alpha_2)$. Then,

$$C'(\alpha_2) = M_{l(\alpha_1)} + \gamma(l(\alpha_1), T+1, \alpha_2). \tag{3.9}$$

From (3.9) and (3.6):

$$\begin{aligned} C'(\alpha_2) - M_{T+1}(\alpha_1) &= (\alpha_2 - \alpha_1)(T - l(\alpha_1))h \\ C'(\alpha_2) &= M_{T+1}(\alpha_1) + (\alpha_2 - \alpha_1)(T - l(\alpha_1))h \qquad (3.10) \end{aligned}$$

From (3.7) and (3.8):

$$\begin{aligned} M_{T+1}(\alpha_2) - C'(\alpha_1) &= (\alpha_2 - \alpha_1)(T - l(\alpha_2))h \\ M_{T+1}(\alpha_2) &= C'(\alpha_1) + (\alpha_2 - \alpha_1)(T - l(\alpha_2))h \qquad (3.11) \end{aligned}$$

It follows from definition that $C'(\alpha_2) \geq M_{T+1}(\alpha_2)$. Substituting from (3.10) and (3.11), we have

$$\begin{aligned} M_{T+1}(\alpha_1) + (\alpha_2 - \alpha_1)(T - l(\alpha_1))h &\geq C'(\alpha_1) + (\alpha_2 - \alpha_1)(T - l(\alpha_2))h \\ \Rightarrow M_{T+1}(\alpha_1) + (\alpha_2 - \alpha_1)(T - l(\alpha_1))h &\geq M_{T+1}(\alpha_1) + (\alpha_2 - \alpha_1)(T - l(\alpha_2))h \\ \Rightarrow (\alpha_2 - \alpha_1)(T - l(\alpha_1))h &\geq (\alpha_2 - \alpha_1)(T - l(\alpha_2))h \\ \Rightarrow (\alpha_2 - \alpha_1)(l(\alpha_2) - l(\alpha_1))h &\geq 0 \\ \Rightarrow l(\alpha_2) &\geq l(\alpha_1). \end{aligned}$$

□

Corollary 3.3.3 *$L(\alpha)$, the largest last regeneration period among all optimal solutions of Problem $P(\alpha)$, is non-decreasing in α.*

Proof: From Theorem 3.3.2 it follows that if $l(\alpha_1) = L(\alpha_1)$, then $l(\alpha_2) \geq L(\alpha_1)$. Therefore, $L(\alpha_2) \geq l(\alpha_2) \geq L(\alpha_1)$. $\square$

Theorem 3.3.4 *Let period T be an FH^q with an optimal common first-period production $x^* \in \chi^q$. For any $\alpha \in \Lambda_q$, let $l(\alpha)$ denote the last regeneration period of an optimal solution of Problem $P(\alpha)$ with first-period production x^*. If $\Gamma(\alpha, j) \geq \alpha + q, \forall j \in \mathcal{R}(\alpha)$, then x^* is also a common optimal first-period production for Problems $P(\alpha + \delta), 0 \leq \delta \leq q$.*

Proof: $\Gamma(\alpha, j) = \frac{C_j(0) - C_{l(\alpha)}(0)}{(j - l(\alpha))h} \geq \alpha + q \geq \alpha + \delta \; \forall j \in \mathcal{R}(\alpha)$. Using $C_j(\alpha + \delta) = C_j(0) + (\alpha + \delta)[T + 1 - (j + 1)]h$ and $C_{l(\alpha)}(\alpha + \delta) = C_{l(\alpha)}(0) + (\alpha + \delta)[T + 1 - (l(\alpha) + 1)]h$, we get,

$$\left\{C_j(\alpha + \delta) - (\alpha + \delta)(T - j)h\right\} - \left\{C_{l(\alpha)}(\alpha + \delta) - (\alpha + \delta)(T - l(\alpha))h\right\} \geq (\alpha + \delta)(j - l(\alpha))h$$

Simplifying, we have $C_j(\alpha + \delta) \geq C_{l(\alpha)}(\alpha + \delta)$. By Corollary 3.3.3, we have

$$l(\alpha) \leq L(\alpha) \leq L(\alpha + \delta) \leq L(\alpha + q).$$

Therefore, $L(\alpha + \delta) \in \mathcal{R}(\alpha)$ and we have

$$M_{T+1}(\alpha + \delta) = C_{L(\alpha+\delta)}(\alpha + \delta) \geq C_{l(\alpha)}(\alpha + \delta)$$

Therefore, $l(\alpha)$ remains an optimal last regeneration period for P$(\alpha + \delta)$. Consequently, at least one optimal solution to every problem instance P$(\alpha + \delta)$, $0 \leq \delta \leq q$ has a first-period production $Q_1^* = x^*$. The result follows. $\square$.

Corollary 3.3.5 *If period T is an FH^q and $\Gamma(\alpha, j) \geq \alpha + q, \forall j \in \mathcal{R}(\alpha)$ for all points $\alpha \in \Lambda_q$, then period T is an FH^0.*

Proof: Follows immediately from Theorem 3.3.4. □

Example 2: Corollary 3.3.5 can be illustrated using the following numerical example. Let the setup cost $k = 25$, and the holding cost $h = 1$. The demand for the first few periods is the same as in Example 1. Using the integer program DLS-T-NSC in Section 3, it can be shown that $T = 15$ is an $\text{FH}^{1/2}_{min}$ (i.e., $q = \frac{1}{2}$). The hypothesis of Corollary 3.3.5 (i.e., the condition $\Gamma(\alpha, j) \geq \alpha + q, \forall j \in \mathcal{R}(\alpha)$ for all points $\alpha \in \Lambda_q$) holds for $q = \frac{1}{2}$. Table 3.3.1 shows the relevant computations. Thus, $T = 15$ is an FH^0.

Table 3.1: Computations to illustrate Corollary 2, Section 4.1. The relevant problem data is provided in Example 2.

α	$\min_{j\in R(\alpha)} \Gamma(\alpha, j)$	$\alpha + q$	α	$\min_{j\in R(\alpha)} \tau(\alpha, j)$	$\alpha + q$	α	$\min_{j\in R(\alpha)} \Gamma(\alpha, j)$	$\alpha + q$
0.00	6.25	0.50	8.50	15.50	9.00	17.00	30.50	17.50
0.50	6.25	1.00	9.00	15.50	9.50	17.50	30.50	18.00
1.00	6.25	1.50	9.50	15.50	10.00	18.00	30.50	18.50
1.50	6.25	2.00	10.00	15.50	10.50	18.50	30.50	19.00
2.00	6.25	2.50	10.50	15.50	11.00	19.00	30.50	19.50
2.50	6.25	3.00	11.00	15.50	11.50	19.50	30.50	20.00
3.00	6.25	3.50	11.50	15.50	12.00	20.00	30.50	20.50
3.50	6.25	4.00	12.00	15.50	12.50	20.50	30.50	21.00
4.00	6.25	4.50	12.50	15.50	13.00	21.00	30.50	21.50
4.50	6.25	5.00	13.00	15.50	13.50	21.50	30.50	22.00
5.00	6.25	5.50	13.50	15.50	14.00	22.00	30.50	22.50
5.50	6.25	6.00	14.00	15.50	14.50	22.50	30.50	23.00
6.00	15.50	6.50	14.50	15.50	15.00	23.00	30.50	23.50
6.50	15.50	7.00	15.00	15.50	15.50	23.50	30.50	24.00
7.00	15.50	7.50	15.50	30.50	16.00	24.00	30.50	24.50
7.50	15.50	8.00	16.00	30.50	16.50	24.50	30.50	25.00
8.00	15.50	8.50	16.50	30.50	17.00	25.00	30.50	25.50

For Problem P(α), our next result characterizes the situation when a positive perturbation in α, the demand in period $T+1$, leaves the last regeneration period unchanged. This result then leads to another sufficiency condition (Corollary 3) for an FH^q to be an FH^0. Before we present the result, we discuss two simple cases where the existence of a positive perturbation is trivial.

Let period T be an FH^q. As before, let $l(\alpha)$ denote the last regeneration period of an optimal solution of Problem P(α).

1. $\underline{\alpha = 0 \text{ and } l(\alpha) = T-1}$: As mentioned earlier, Problem P(0) is a T-period

problem. If $l(0) = T - 1$, then period T is a production period. Consequently, T remains a production period for any N-period problem, $N \geq T + 1$ [103]. Therefore, period T is an FH^0.

2. $\underline{\alpha > 0 \text{ and } l(\alpha) = T}$: For $\alpha > 0$, Problem P(α) is a $(T+1)$-period problem. If $l(\alpha) = T$, then $T + 1$ is a production period. Therefore, in the optimal solution to any problem P(α'), $\alpha' \geq \alpha$, period $T + 1$ remains a production period [103]. Note that the existence of a demand α in period $T + 1$ for which $l(\alpha) = T$ is guaranteed; e.g., $l(\alpha) = T$ for $\alpha \geq \frac{k}{h}$. Let $\hat{\alpha} = \min\{\alpha : \alpha \in \Lambda_q, l(\alpha) = T\}$. Thus, if all problem instances P(α), $0 \leq \alpha < \hat{\alpha}$, have a common first period production in set χ^q, then period T is an FH^0. Let $\Lambda_{q,\hat{\alpha}} = \{\alpha : \alpha \in \Lambda_q, 0 \leq \alpha < \hat{\alpha}\}$.

Theorem 3.3.6 *Given an optimal last regeneration period $l(\alpha)$ for Problem $P(\alpha)$, $\alpha \in \Lambda_{q,\hat{\alpha}}$, there exists a number $\tilde{\delta}(\alpha) > 0$ with $l(\alpha + \delta) = l(\alpha) \forall\ 0 < \delta \leq \tilde{\delta}(\alpha)$ if, and only if, $l(\alpha) = L(\alpha)$.*

Proof: The cost of Problem P($\alpha + \delta$) with the last regeneration period fixed to $l(\alpha)$ can be written as

$$C_{l(\alpha)}(\alpha + \delta) = C_{l(\alpha)}(\alpha) + \delta(T - l(\alpha))h.$$

In order for period $l(\alpha)$ to remain an optimal last regeneration period, the following inequality has to be satisfied.

$$\begin{aligned} C_j(\alpha + \delta) &\geq C_{l(\alpha)}(\alpha + \delta) \quad \forall j \in \{l(\alpha)+1, l(\alpha)+2, ..., T\} \\ C_j(\alpha) + \delta(T - j)h &\geq C_{l(\alpha)}(\alpha) + \delta(T - l(\alpha))h \quad \forall j \in \{l(\alpha)+1, l(\alpha)+2, ..., T\} \\ \frac{C_j(\alpha) - C_{l(\alpha)}(\alpha)}{(j - l(\alpha))h} &\geq \delta \quad \forall j \in \{l(\alpha)+1, l(\alpha)+2, ..., T\} \end{aligned}$$

Let

$$\tilde{\delta}(\alpha) = \min_{j \in \{l(\alpha)+1, l(\alpha)+2, ..., T\}} \left\{ \frac{C_j(\alpha) - C_{l(\alpha)}(\alpha)}{(j - l(\alpha))h} \right\}$$

By construction, $l(x) = l(\alpha)$ for all $x \in [\alpha, \alpha + \tilde{\delta}(\alpha)]$. If $l(\alpha) = L(\alpha)$, then $C_j(\alpha) > C_{l(\alpha)}(\alpha)$, and hence $\tilde{\delta}(\alpha) > 0$. Conversely, if $\tilde{\delta}(\alpha) > 0$, then $C_j(\alpha) > C_{l(\alpha)}(\alpha)\ \forall\ j \in \{l(\alpha)+1, l(\alpha)+2, ..., T\}$. This implies $l(\alpha) = L(\alpha)$. □

Corollary 3.3.7 *Let T be an FH^q with* $\cap_{\alpha\in\Lambda_{q,\hat{\alpha}}}\hat{\mathcal{Q}}_1(L(\alpha))\neq\emptyset$ *and* $q\leq\tilde{\delta}$ $=\min_{\alpha\in\Lambda_{q,\hat{\alpha}}}\{\tilde{\delta}(\alpha)\}$. *Then, T is an FH^0.*

Proof: If $\cap_{\alpha\in\Lambda_{q,\hat{\alpha}}}\hat{\mathcal{Q}}_1(L(\alpha))\neq\emptyset$, then using $L(\alpha)$ instead of $l(\alpha)$ in the proof of Theorem 3.3.6, we have $\tilde{\delta}(\alpha)>0\ \forall\alpha\in\Lambda_{q,\hat{\alpha}}$. It follows that $\tilde{\delta}>0$. Since $q\leq\tilde{\delta}$, Theorem 3.3.6 implies that for all $\alpha\in\Lambda_{q,\hat{\alpha}}$, any Problem P$(\alpha+\delta), 0<\delta<q$, has an optimal solution with the first-period production in χ^q. This observation, together with the remarks immediately before Theorem 3.3.6, implies the existence of an optimal first period production in χ^q for all Problems P(α), $\alpha\geq 0$. Therefore, period T is an FH^0. □

Example 3: Theorem 3.3.6 and Corollary 3.3.7 can be illustrated using the following example. Consider a four-period problem (i.e., T = 4) with demands 2, 1, 4, 2, respectively in the four periods. Let the setup cost $k=5$, and the holding cost $h=1$. For this data, $T=4$ is a FH^1 (i.e., $q=1$) with $\chi^q=\{3\}$. It is easy to verify that for all integer fifth-period demands greater than two, period five is a production period. Thus, $\hat{\alpha}=3$, and $\Lambda_{q,\hat{\alpha}}=\{0,1,2\}$. For each $\alpha\in\Lambda_{q,\hat{\alpha}}$, Table 3.3.1 shows the computation of $\tilde{\delta}$.

Table 3.2: Computations to illustrate Theorem 3.3.6 and Corollary 3.3.7.

α	$l(\alpha)$	$\frac{C_j(\alpha)-C_{l(\alpha)}(\alpha)}{(j-l(\alpha))h}$	$\tilde{\delta}(\alpha)$
0	2	6 ($j=3$)	6
1	2	5 ($j=3$), 1.5 ($j=4$)	1.5
2	2	4 ($j=3$), 1 ($j=4$)	1

Thus, $q=1=\tilde{\delta}=\min_{\alpha\in\Lambda_{q,\hat{\alpha}}}\{\tilde{\delta}(\alpha)\}$. Also, $\cap_{\alpha\in\Lambda_{q,\hat{\alpha}}}\hat{\mathcal{Q}}_1(L(\alpha))=\{3\}$. Therefore, from Corollary 3, we have that $T=4$ is an FH^0.

Theorem 3.3.8 *For period T to be an FH^0, it is necessary that* $\cap_{\alpha\in\Lambda_q}\hat{\mathcal{Q}}_1(L(\alpha))\neq\emptyset$ *for any* $q>0$.

Proof: Let period T be an FH^0, and let x^* be a common optimal first-period production quantity for all N-period problems, $N \geq T$. For $q > 0$, we know that T is also an FH^q.

Consider $\alpha \in \Lambda_q$. Suppose Problem P$(\alpha, L(\alpha))$ has an optimal solution with first period production $Q_1 \neq x^*$. This implies, since T is an FH^0, that there exists an optimal solution to Problem P(α) with last regeneration period $l(\alpha) \neq L(\alpha)$. That is, Problem P(α) has at least two optimal solutions with different last regeneration periods. Then, there exists $\epsilon > 0$ such that every optimal solution of Problem P$(\alpha+\epsilon)$ has $L(\alpha)$ as the unique last regeneration period [22]. Since T is an FH^0, Problem P$(\alpha + \epsilon, L(\alpha))$ has x^* as an optimal first period production quantity. Therefore, P$(\alpha, L(\alpha))$ also has x^* as an optimal first period production quantity. Since the choice of α is arbitrary, $x^* \in \cap_{\alpha\in\Lambda_q} \hat{\mathcal{Q}}_1(L(\alpha))$. $\square$

Note: For $q > 0$, the condition $\cap_{\alpha\in\Lambda_q} \hat{\mathcal{Q}}_1(L(\alpha)) \neq \emptyset$ implies that T is an FH^q. The converse, of course, is not necessarily true.

The expressions in the hypotheses of Corollaries 2 and 3 are easy to compute. We explain the computation of two fundamental quantities: (i) $C_l(\alpha)$, the optimal cost of Problem P(α) with last regeneration period l, can be computed by setting $X_{l+1} = 1$ and $X_j = 0$ for $l + 2 \leq j \leq T + 1$, in the formulation DLS-T of Section 2, (ii) For Problem P(α), the largest last regeneration period $L(\alpha)$ can be found by first finding the optimal cost $M_{T+1}(\alpha)$ and a corresponding last regeneration period $l(\alpha)$. Then, $L(\alpha)$ is the largest period k, $l(\alpha) \leq k \leq T$ such that fixing the last regeneration period to k produces an optimal solution of cost $M_{T+1}(\alpha)$. Again, formulation DLS-T can be used for all these computations.

3.3.2 The impact on cost of using FH$^q_{min}$ in place of FH$^0_{min}$

Unlike a continuous forecast horizon, a discrete forecast horizon does not guarantee the existence of optimal decisions over the decision horizon for all non-negative future demands. Therefore, the impact on the cost resulting from using discrete forecast horizons (instead of a continuous forecast horizons) becomes an important measure to assess their utility in practice. Specifically, we pose the following question:

Let T be an $FH^q, q > 0$, with an optimal common first-period production $x^* \in \chi^q$. Then by definition, it is optimal to choose x^* as a first-period production quantity if future demands (i.e., demands during period $T+1$ and later) are in integer multiples of q. If, however, future demands do not satisfy this restriction, then how does the cost of a solution that uses x^* as a first-period production quantity compare with the optimal cost?

The relevance of this question is clear: if we can guarantee that, for any N-period problem, $N \geq T+1$, the cost of using x^* as a first-period production quantity is reasonably close to the optimal cost, then the shorter discrete forecast horizon (i.e., T) becomes a reasonable substitute for the longer continuous forecast horizon. Our analysis below shows that this is indeed true. We start by explaining the concept of an ϵ-forecast horizon, first used in [87].

Let $\mathrm{P}(\bar{d}_1^T, \bar{d}_{T+1}^N)$ denote an N-period problem, $N > T$ with (known) demand $\bar{d}_1^T = (d_1, d_2, ..., d_T)$ in the first T periods, and demand $\bar{d}_{T+1}^N = (d_{T+1}, d_{T+2}, ..., d_N)$ in periods $T+1, T+2, ..., N$. Let $M_N(\bar{d}_1^T, \bar{d}_{T+1}^N)$ denote the optimum cost of $\mathrm{P}(\bar{d}_1^T, \bar{d}_{T+1}^N)$.

Definition: Period T is an ϵ-*forecast horizon*, denoted as ϵ-FH^0, if for every N-period problem, $N \geq T+1$ and all vectors of demands $\bar{d}_{T+1}^N \in \Re_+^{N-T}$, there exists at least one solution with x^* as the first-period production quantity, and with cost at most $(1+\epsilon)M_N(\bar{d}_1^T, \bar{d}_{T+1}^N)$.

For $\epsilon = 0$, note that the above definition reduces to that of the classical forecast horizon. Clearly, if period T is a forecast horizon (or, equivalently, a 0-FH^0), then T is also an FH^q. We are interested in examining the reverse relationship.

Question: If T is an FH^q, then for what ϵ is T an ϵ-FH^0?

Suppose T is an FH^q. Note that

$$M_{T+1}(\alpha) \leq M_{T+1}(\alpha + \delta) \leq M_{T+1}(\alpha + q), \ 0 \leq \delta \leq q, \ \alpha \in \Lambda_q.$$

This follows because any feasible solution to $\mathrm{P}(\alpha + q)$ is feasible for $\mathrm{P}(\alpha + \delta)$, and any feasible solution to P$(\alpha+\delta)$ is feasible for P(α). Our analysis uses an upper bound on the ratio $\frac{M_{T+1}(\alpha+q)}{M_{T+1}(\alpha)}$.

An Upper Bound on the Ratio $\frac{M_{T+1}(\alpha+q)}{M_{T+1}(\alpha)}$ and Further Analysis

Let T be an FH^q. For $\alpha \in \Lambda_q$, consider the following two cases:

1. There is an optimum solution to P(α) with only one setup (i.e., the mandatory setup in the first period). Since T is an FH^q, it is easy to see that a single setup can occur only if $\alpha = 0$. Then, the unique optimal first-period production for all Problems P$(\alpha), \alpha \in \Lambda_q$ is $\sum_{i=1}^{T} d_i$. Therefore, $M_{T+1}(0) = k + h\sum_{i=2}^{T}(i-1)d_i$, and $M_{T+1}(\alpha) = M_{T+1}(0) + k$ for $\alpha \in \Lambda_q \backslash \{0\}$. Thus,
$$\begin{aligned}\frac{M_{T+1}(q)}{M_{T+1}(0)} &= \frac{M_{T+1}(0)+k}{M_{T+1}(0)} \\ &= \left(1 + \frac{k}{k + h\sum_{i=2}^{T}(i-1)d_i}\right),\end{aligned}$$
and $\frac{M_{T+1}(\alpha+q)}{M_{T+1}(\alpha)} = 1$ for $\alpha \in \Lambda_q \backslash \{0\}$. In general, we have $\frac{M_{T+1}(\alpha+q)}{M_{T+1}(\alpha)} \le 1 + \epsilon_1$, where $\epsilon_1 = \frac{k}{k+h\sum_{i=2}^{T}(i-1)d_i}$. Note that $\epsilon_1 \le 1$.

2. There exists an optimum solution to P(α) with $\rho_\alpha \ge 2$ setups. In this case,
$$M_{T+1}(\alpha) \ge \rho_\alpha k$$
Given a feasible solution to Problem P(α), a feasible solution to P$(\alpha+q)$ can be obtained by producing an extra q units in the period $T+1$. Therefore, $M_{T+1}(\alpha+q) \le M_{T+1}(\alpha) + k$. Then, we have
$$\begin{aligned}M_{T+1}(\alpha+q) &= \left[\frac{M_{T+1}(\alpha+q)}{M_{T+1}(\alpha)}\right] M_{T+1}(\alpha) \\ &\le \left[\frac{M_{T+1}(\alpha)+k}{M_{T+1}(\alpha)}\right] M_{T+1}(\alpha) \\ &\le \left[1 + \left\{\frac{k}{\rho_\alpha k}\right\}\right] M_{T+1}(\alpha) \\ &= (1+\epsilon_2^\alpha) M_{T+1}(\alpha),\end{aligned}$$
where $\epsilon_2^\alpha = \frac{1}{\rho_\alpha}; \rho_\alpha \ge 2$.

Let $\epsilon^\alpha = \max\{\epsilon_1, \epsilon_2^\alpha\}$. We have thus proved the following result.

Theorem 3.3.9 *Let T be an FH^q. For any $\alpha \in \Lambda_q$, the optimum cost of $P(\alpha+q)$ is at most $(1+\epsilon^\alpha)$ times the optimum cost of $P(\alpha)$, where $\epsilon^\alpha = \max\{\epsilon_1, \epsilon_2^\alpha\}; \epsilon_1 \leq 1, \epsilon_2^\alpha \leq \frac{1}{2}$.*

Finally, we note the following for all $\alpha \in \Lambda_q$:

1. Any solution to P$(\alpha + q)$ is *feasible* for P$(\alpha + \delta), 0 \leq \delta \leq q$.
2. Since T is an FH^q, there exists a solution of value $M_{T+1}(\alpha+q)$ with first-period production quantity x^*.

Let $\epsilon_2 = \max_{\alpha\in\Lambda_q} \epsilon_2^\alpha$; note that $\epsilon_2 \leq \frac{1}{2}$, and let $\epsilon = \max\{\epsilon_1, \frac{1}{2}\}$.

Also, for $\alpha > \frac{k}{h}$, $\mathcal{Q}_1(\alpha) = \mathcal{Q}_1(0)$ (see Section 3). Thus, for any $\alpha \in \Re_+$, we have demonstrated a feasible solution to P(α) with first-period production quality x^* such that the cost of the solution is at most $(1+\epsilon)M_{T+1}(\alpha)$. We formally state this result below.

Corollary 3.3.10 *If T is an FH^q, then T is a $(1+\epsilon)$-FH^0 with*
$\epsilon = \max\left\{\frac{1}{1+\frac{h}{k}\sum_{i=2}^{T}(i-1)d_i}, \frac{1}{2}\right\}$.

Notes: Let T be an FH^q. Then, since $\epsilon_1 = \frac{1}{1+\frac{h}{k}\sum_{i=2}^{T}(i-1)d_i} \leq 1$, T is a 2-FH^0. Furthermore, T is a 1.5-FH^0 if $\sum_{i=2}^{T}(i-1)d_i \geq \frac{k}{h}$. For example, for the DLS model with unit initial demand considered in [24], T is a 1.5-FH^0 if $T \approx \sqrt{\frac{2k}{h}}$, or higher.

3.3.3 Computational Experience

Throughout this section, we use minimal forecast horizons assuming integer future demand (i.e., FH$^1_{min}$) to illustrate our results for discrete forecast horizons. Our computations address the following issues:

1. *The practicality of using integer programming (IP) for computing discrete and continuous forecast horizons.* We use the IP formulation in Section 3.2 for computing discrete forecast horizons; classical forecast horizons are computed using the IP formulations presented in [35]. The computation of discrete horizons for variants of the dynamic lot-size problem is discussed later in Section 3.4.

2. *The impact of discreteness (of future demands) on the length of the minimal forecast horizon.* As mentioned earlier, the reduction in this length, with respect to the length of the continuous minimal forecast horizon, can imply significant savings in expenses related to forecasting. We measure the magnitude of this reduction via a detailed computational study.

3. *The impact on cost of using a discrete minimal forecast horizon.* Our analysis in Section 3.3.2 establishes an upper bound on the increase in total cost from using the shorter discrete minimal forecast horizon instead of the longer continuous forecast horizon. For realistic data, we compute the exact value of the upper bound and show that this increase is much smaller than the worst-case guarantee; therefore, we believe that discrete horizons can be acceptable substitutes for continuous horizons in practice.

The Test Bed

We start by describing the test bed. We consider stationary as well as non-stationary demands.

Stationary demand:

The problem instances are generated the same way as discussed in Section 2.5.1 in chapter 2.

Non-stationary demand:

The problem instances are generated the same way as discussed in Section 2.5.2 in chapter 2.

The integer programming formulations were solved using the MINTO (version 8.1) programming library [79]. All the computations were carried out on a Pentium IV computer (2.4 GHz, 512 MB RAM) running Red Hat Linux 7.2.

Computation Times

For stationary demand, the minimum forecast horizons, with and without the assumption of integer future demand, and the corresponding computation times are shown in Table 3.3.3. For non-stationary data, the minimum, maximum, and median minimal forecast horizons along with the average computation times are reported in Tables 3.3.3. Each row in Table 3.3.3 corresponds to the average statistics over the 11 instances generated with $G = 1.00$ and $S = 0.50$. Observe that, on average, the problems are solved within an hour of computation time. It is, therefore, reasonable to conclude that using integer programming is an attractive option for practitioners.

Table 3.3: The minimal forecast horizon and the minimal forecast horizon assuming integer future demand for unit initial demand.

k	$\mathbf{FH}^0_{min}$	**Time (sec.)**	$\mathbf{FH}^1_{min}$	**Time (sec.)**	**% Reduction** $(\frac{\mathrm{FH}^0_{min}-\mathrm{FH}^1_{min}}{\mathrm{FH}^0_{min}}) \times 100$
10	20	1.01	16	2.58	20.00%
11	17	0.92	13	1.79	23.52%
12	13	0.86	13	1.77	0.00%
13	18	1.03	13	2.08	27.7%
14	24	1.73	19	5.62	20.83%
15	30	34.58	24	14.99	20.00%
16	26	7.39	21	9.95	19.23%
17	21	1.16	16	5.24	23.80%
18	21	1.42	16	3.92	23.80%
19	28	8.53	16	4.82	42.85%
20	35	216.63	28	139.55	20.00%
21	42	402.11	32	805.23	23.80%
22	37	245.83	29	617.34	21.62%
23	31	179.32	25	855.13	19.35%
24	25	10.45	23	703.30	5.71%
25	32	203.71	24	312.29	25.00%

Table 3.4: The minimal forecast horizon and the minimal forecast horizon assuming integer future demand: non-stationary demand with $G = 1.000$ and $S = 0.50$.

k	FH^0_{min}				FH^1_{min}			
	Min.	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**
10	2	4	3	0.05	2	4	3	1.21
15	3	13	6	0.64	3	9	5	2.68
20	3	17	7	1.34	3	17	5	4.13
30	4	18	9	2.29	4	9	6	3.87
50	8	20	12	3.18	8	20	11	5.68
75	12	23	15	3.94	10	18	14	126.51
100	15	22	17	35.32	13	21	16	1129.34
150	16	32	24	192.44	16	29	20	2496.12

Reduction in Forecast Horizon

For stationary demand, Figure 3.2 shows the relationship between the minimal forecast horizons, with and without the integrality assumption, and the setup cost; the corresponding data is in Table 3.3.3. For non-stationary demand, this relationship is shown in Figure 3.3; the corresponding data is in Table 3.3.3. The average reduction in the length of the minimal forecast horizon resulting from the integrality assumption is around 20% for stationary demand, and around 15% for non-stationary demand.

Behavior of discrete horizons with demand growth and variability

For non-stationary data, Figure 3.4 is a plot of the median forecast horizon, assuming integer future demand, as a function of the setup cost for the three values of S (0.15, 0.50, 1.15) and G set equal to 1.005. For each of the three values of S, the median minimal forecast horizon typically increases with the setup cost. For a given setup cost, the median forecast horizon is typically lower for higher values of S. While a precise mathematical explanation seems difficult to prove, we offer the following intuitive explanation: a higher value of the demand variability parameter S indicates a larger fluctuation in the demand, thereby increasing the probability of a large demand, and consequently reducing the forecast horizon. As expected FH^1_{min} is smaller than FH^0_{min} for each parameter setting. Also note that FH^1_{min} is less sensitive

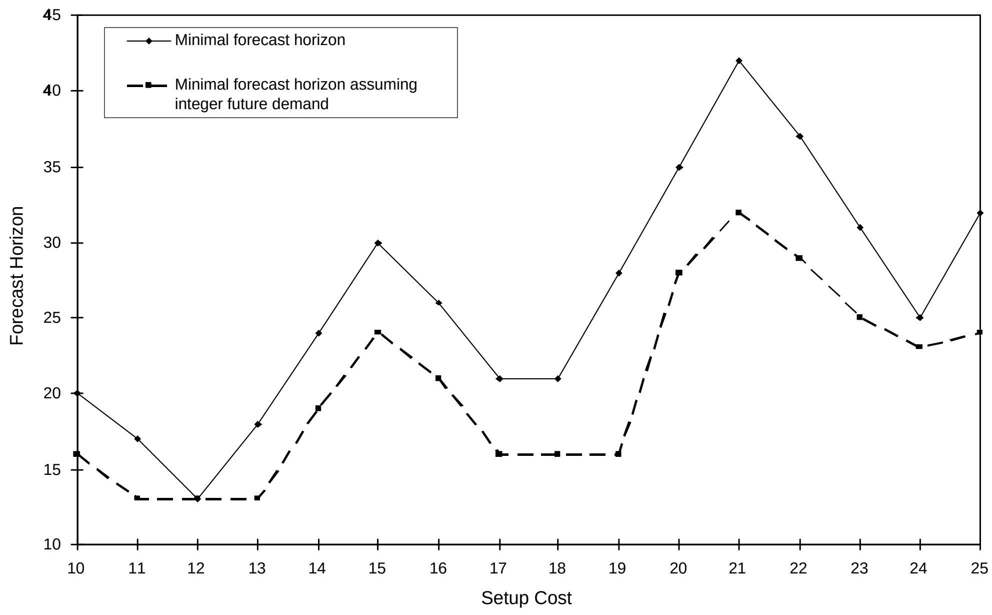

Figure 3.2: Relationship between the setup cost and the minimal forecast horizon for stationary demand, with and without the integrality assumption on future demand – unit demand case.

to changes in S. With S fixed at 0.50, Figure 3.5 presents the relationship between the median minimal forecast horizon and the setup cost for the three values of the growth parameter G: 0.995, 1.000, and 1.005. As with Figure 3.4, the minimal forecast horizons typically increase with the setup cost. The other observation is that the median forecast horizon is larger for smaller values of G. When G is large, the demands get increasingly larger in the later periods and as a result, there is a better chance of a smaller forecast horizon. As with Figure 3.4, FH^1_{min} tends to be smaller than FH^0_{min} for each parameter setting, and also less sensitive to changes of G.

A summary of the results for all combinations of the setup cost k, the growth parameter G, and the variability parameter S, is reported in Tables 3.3.3-3.3.3. Tables 3.3.3-3.3.3 show the results for continuous forecast horizons, while Table 3.3.3-3.3.3 show the results assuming integer future demand.

Table 3.5: Minimal forecast horizons for non-stationary demand: flat demand pattern ($G = 1.000$).

k	S=0.15				S=0.50				S=1.15			
	FH$^0_{min}$				**FH**$^0_{min}$				**FH**$^0_{min}$			
	Min.	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**
10	2	6	3	0.56	2	6	3	0.21	2	7	3	0.16
15	9	14	8	1.02	2	9	5	0.92	2	8	4	0.73
20	4	23	11	1.11	2	9	4	1.03	2	7	4	0.89
30	3	20	13	1.78	6	15	8	1.52	2	13	6	1.32
50	11	27	19	2.42	5	21	17	1.96	8	14	6	1.61
75	12	24	15	3.52	7	22	15	2.51	6	11	8	1.94
100	15	25	21	35.62	9	17	14	14.24	5	16	9	22.12
150	20	35	28	142.51	16	32	24	125.71	7	28	16	122.16

Table 3.6: Minimal forecast horizons for non-stationary demand: increasing demand pattern ($G = 1.005$).

k	S=0.15				S=0.50				S=1.15			
	FH$^0_{min}$				**FH**$^0_{min}$				**FH**$^0_{min}$			
	Min.	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**
10	2	8	5	0.83	2	5	3	0.61	2	6	4	0.43
15	3	15	7	0.99	2	5	3	0.78	2	6	4	0.59
20	3	11	8	1.31	2	8	5	1.04	2	7	5	0.81
30	5	14	6	2.17	6	19	14	1.41	4	12	7	0.95
50	6	19	12	3.12	5	23	16	1.94	5	11	8	1.48
75	13	29	17	6.23	8	25	15	5.12	4	15	6	3.82
100	16	26	21	121.52	9	22	16	103.42	3	17	13	89.41
150	18	31	25	452.92	12	27	19	326.12	2	23	16	173.4

Table 3.7: Minimal forecast horizons for non-stationary demand: decreasing demand pattern ($G = 0.995$).

k	S=0.15				S=0.50				S=1.15			
	FH$^0_{min}$				**FH**$^0_{min}$				**FH**$^0_{min}$			
	Min.	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**
10	4	8	5	1.13	2	6	4	1.04	4	7	6	0.82
15	4	14	8	1.52	2	7	5	1.31	2	6	5	0.93
20	6	16	12	1.89	4	12	6	1.42	7	9	6	1.18
30	5	15	10	2.41	7	13	8	1.72	8	13	10	1.39
50	8	25	17	6.27	7	21	14	2.18	6	24	17	1.92
75	14	28	18	22.17	6	22	17	3.62	13	25	16	2.04
100	18	34	24	153.82	12	28	25	24.73	14	23	18	11.04
150	23	36	27	843.23	10	35	25	632.68	10	28	24	428.22

Table 3.8: Minimal forecast horizons assuming integer future demand: flat demand pattern ($G = 1.000$).

k	S=0.15				S=0.50				S=1.15			
	FH$^1_{min}$			**Time (sec.)**	**FH**$^1_{min}$			**Time (sec.)**	**FH**$^1_{min}$			**Time (sec.)**
	Min.	**Max.**	**Med.**		**Min.**	**Max.**	**Med.**		**Min.**	**Max.**	**Med.**	
10	2	4	3	1.21	2	3	2	1.32	2	5	3	1.01
15	7	9	5	2.68	2	5	2	1.41	2	4	2	1.89
20	3	17	5	4.13	2	7	3	2.02	2	6	2	1.43
30	4	9	6	3.87	3	10	6	2.14	2	8	4	1.80
50	8	20	11	5.68	5	16	11	4.07	4	8	5	1.93
75	10	18	14	126.51	6	14	11	4.86	4	8	5	2.26
100	13	21	16	1129.34	9	13	8	704.14	5	12	7	323.14
150	16	29	20	2496.12	12	25	16	941.13	5	23	7	530.19

Table 3.9: Minimal forecast horizons assuming integer future demand: increasing demand pattern ($G = 1.005$).

k	S=0.15				S=0.50				S=1.15			
	FH$^1_{min}$			**Time (sec.)**	**FH**$^1_{min}$			**Time (sec.)**	**FH**$^1_{min}$			**Time (sec.)**
	Min.	**Max.**	**Med.**		**Min.**	**Max.**	**Med.**		**Min.**	**Max.**	**Med.**	
10	2	5	3	1.46	2	3	2	0.90	2	4	3	0.83
15	3	10	6	4.18	2	4	3	1.11	2	6	3	0.94
20	3	7	5	4.32	2	4	3	2.07	2	6	4	2.10
30	3	9	7	4.01	4	15	6	2.02	2	8	4	2.22
50	6	13	8	7.03	5	16	7	5.22	2	8	3	2.31
75	7	21	13	421.56	5	15	10	11.13	2	11	5	2.39
100	11	19	14	2312.40	5	10	8	280.33	2	11	7	163.11
150	13	24	15	3013.29	5	22	17	2046.20	2	14	8	491.43

Table 3.10: Minimal forecast horizons assuming integer future demand: decreasing demand pattern ($G = 0.995$).

k	S=0.15				S=0.50				S=1.15			
	FH$^1_{min}$			**Time (sec.)**	**FH**$^1_{min}$			**Time (sec.)**	**FH**$^1_{min}$			**Time (sec.)**
	Min.	**Max.**	**Med.**		**Min.**	**Max.**	**Med.**		**Min.**	**Max.**	**Med.**	
10	2	5	3	1.63	2	4	4	0.83	2	5	2	0.81
15	3	7	3	3.08	2	5	3	1.26	2	5	2	1.01
20	3	6	3	2.07	2	8	4	1.80	4	7	4	1.33
30	4	7	4	2.97	4	10	7	2.41	3	8	5	1.80
50	5	16	8	11.13	4	12	8	4.79	4	17	11	4.36
75	10	23	13	291.10	6	14	8	129.13	8	14	10	2.76
100	10	25	22	2418.50	6	25	11	703.49	5	21	16	484.12
150	16	32	24	4205.17	7	28	22	3489.19	8	19	14	852.18

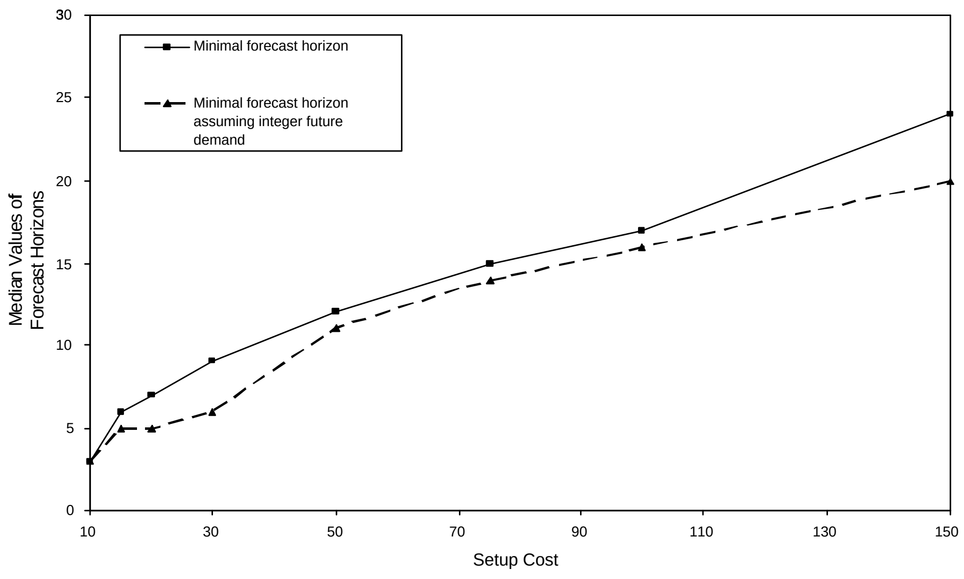

Figure 3.3: Relationship between the setup cost and the median values of forecast horizons FH^0_{min} and FH^1_{min} - non-stationary demand case

Impact on cost due to the integrality assumption

The issue of the impact on cost from using FH^1_{min} in place of FH^0_{min} was addressed in Section 3.3.2. Let T be an FH^q. Then, for $\alpha \in \Lambda_q$, we showed that $\frac{M_{T+1}(\alpha+q)}{M_{T+1}(\alpha)} \leq 1 + \max\{\epsilon_1, \epsilon_2^\alpha\}$. Here, $\epsilon_1 = \frac{k}{k+h\sum_{i=2}^{T}(i-1)d_i}$, and $\epsilon_2^\alpha = \frac{1}{\rho_\alpha}$, where ρ_α is the maximum number of setups in an optimal solution to Problem P(α) with at least two setups. As defined in Section 3.3.2, $\epsilon^\alpha = \max\{\epsilon_1, \epsilon_2^\alpha\}$. The argument presented in Section 3.3.2 in fact shows that if T is an FH^q, then T is a $(1+\epsilon_{th})$-FH^0, where $\epsilon_{th} = \max\{\epsilon^\alpha; \alpha \in \Lambda_q\}$ represents the theoretical upper bound on the increase in cost. Given the granularity q and the demands for the first T periods, the value of ϵ_{th} can be easily computed. However, for realistic data, the actual values of ϵ_1 and ϵ_2^α are typically much smaller than their theoretical upper bounds. Let $\epsilon_{act} = \max \frac{M_{T+1}(\alpha+q)}{M_{T+1}(\alpha)}; \alpha \in \Lambda_q$. Under the assumption of integer future demand (i.e., $q = 1$), Table 3.3.3 lists the theoretical and actual values of ϵ (ϵ_{th} and ϵ_{act}, respectively) for a sample of 10 instances. The results indicate that ϵ_{act} is in fact much smaller than ϵ_{th}. Also, ϵ_{act} tends to decrease with an increase in FH^1_{min}.

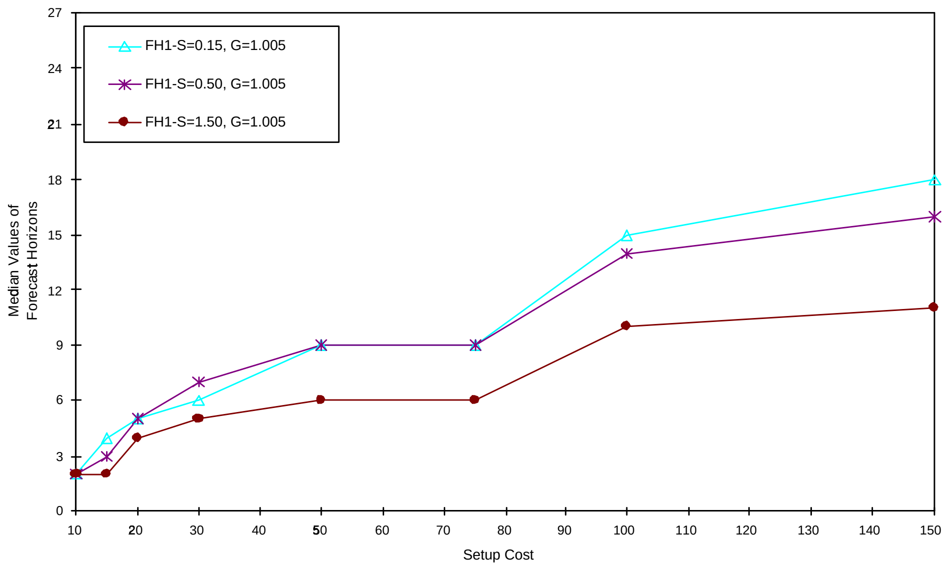

Figure 3.4: The median values of forecast horizons, assuming integer future demand, as a function of setup cost and demand variability.

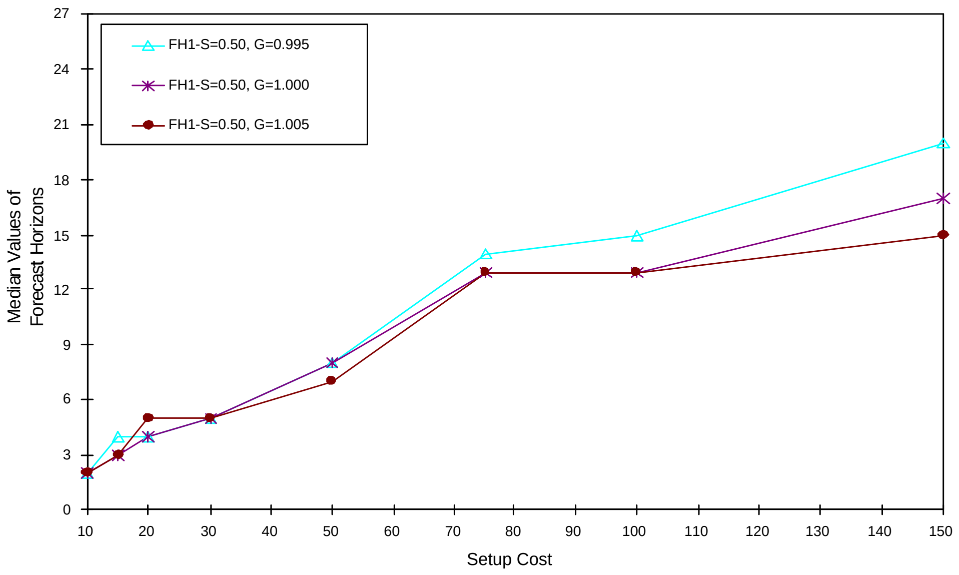

Figure 3.5: The median values of forecast horizons, assuming integer future demand, as a function of setup cost and demand growth.

Table 3.11: The Theoretical and actual upper bounds on the cost of using FH^1_{min} in place of FH^0_{min}: non-stationary demand with $G = 0.995$ and $S = 0.15$.

Problem No.	k	FH^1_{min}	ϵ_{th}	ϵ_{act}
1	10	2	0.500	0.217
2	20	5	0.500	0.235
3	30	8	0.333	0.182
4	40	11	0.333	0.161
5	50	15	0.333	0.136
6	60	18	0.333	0.011
7	70	21	0.250	0.026
8	80	26	0.250	0.054
9	90	33	0.250	0.084
10	100	41	0.250	0.073

3.4 Extensions to the Basic Dynamic Lot-Size model

The structured, constraint-based nature of the integer programming approach makes it easy to incorporate additional constraints and to address multi-product variants of the DLS problem. We illustrate this simplicity on two significant extensions. The forecast horizons considered assume integer future demand.

3.4.1 Dynamic Lot-Sizing in the Presence of a Warehouse Capacity Constraint

Consider a situation where the inventory at the end of a period needs to be stored in a warehouse with capacity, say, W. The dynamic lot-size problem, therefore, has an additional constraint that specifies the ending inventory at each period to be at most W. Note that, for this warehouse-capacity constrained problem, if the demand in a period exceeds W or exceeds $\lfloor \frac{k}{h} \rfloor$, then that period is a production period (i.e., production must occur in that period). The proof of the following result is similar to that of Theorem 3.2.1; we omit the proof. The notation is the same as that defined in Section 3.2.

Theorem 3.4.1 *T is an FH (assuming integer future demand) for the warehouse-capacity constrained DLS problem if, and only if, $\cap_{\alpha \in \{0,1,2,\ldots,\min(W,\lfloor \frac{k}{h} \rfloor)\}} \mathcal{Q}_1(\alpha) \neq \emptyset$.*

A corresponding formulation can be obtained by adding the following constraints to Problem DLS-T-NSC

$$I_j^\alpha \leq W,\ 1 \leq j \leq T; \alpha \in \Lambda, \tag{3.12}$$

where $\Lambda = \{0, 1, 2, ..., \min(W, \lfloor \frac{k}{h} \rfloor)\}$.

3.4.2 A Two-Product Dynamic Lot-Size Problem with Individual and Joint Setup Costs

In this section we extend the necessary and sufficient conditions discussed in Section 3.2 to a two-product variant. Let Product i, $i = 1, 2$, refer to the two products. The per-unit holding cost is h_1 (resp. h_2) for Product 1 (resp. Product 2). If production is limited to a single product in a period, say Product i, then the setup cost incurred in that period is $k_i + K$ (i=1, 2). If both products are produced in a period, the setup cost is $k_1 + k_2 + K$.

Let $\mathcal{Q}_1(\alpha_1, \alpha_2)$ be the set of optimal first-period production quantities for a $(T+1)$-period problem with a demand of α_i for Product i, $i = 1, 2$, in period $T+1$. If $\alpha_i > \lfloor \frac{K+k_i}{h_i} \rfloor, i = 1, 2$, then $\mathcal{Q}_1(\alpha_1, \alpha_2) = \mathcal{Q}_1(0, 0)$.

Theorem 3.4.2 *T is a forecast horizon (assuming integer future demand) if, and only if*

$\cap_{(\alpha_1,\alpha_2)\in\bar{\Lambda}} \mathcal{Q}_1(\alpha_1, \alpha_2) \neq \emptyset$, *where* $\bar{\Lambda} = \{(\alpha_1, \alpha_2) | \{(\alpha_1 \leq \lfloor \frac{K+k_1}{h_1} \rfloor)\ OR\ (\alpha_2 \leq \lfloor \frac{K+k_2}{h_2} \rfloor)\}\}$.

Proof: The necessity is trivial. To prove sufficiency, assume that $\cap_{(\alpha_1,\alpha_2)\in\bar{\Lambda}} \mathcal{Q}_1(\alpha_1, \alpha_2) \neq \emptyset$. We know that every instance of a N-period problem with $N \geq T+1$ has the same first-period production as a $(T+1)$-period problem with an appropriately chosen $T+1$-period demands, α_1 and α_2, for the two products [21]. If $(\alpha_1, \alpha_2) \in \bar{\Lambda}$, the result follows from the hypothesis. Otherwise, $\mathcal{Q}_1(\alpha_1, \alpha_2) = \mathcal{Q}_1(0, 0)$, and the result follows. $\square$

For a given planning horizon T, the following formulation tests whether or not T is a forecast horizon:

$\bar{\Lambda}$	:	$\{(\alpha_1, \alpha_2) \| \{\alpha_1 \leq \lfloor \frac{K+k_1}{h_1} \rfloor$ OR $\alpha_2 \leq \lfloor \frac{K+k_2}{h_2} \rfloor\}\}$. We index $\bar{\Lambda}$ by $\bar{\alpha}$.
$M_{T+1}(\bar{\alpha})$	:	the optimal objective function value for the $(T+1)$-period. problem with period T+1 demand $\bar{\alpha}$. We use $\bar{\alpha}=0$ to denote the T-period problem.
$X_{ij}^{\bar{\alpha}}$	:	$\begin{cases} 1 & \text{if a setup is required for Product } i \text{ in period } j \text{ in} \\ & \text{the problem with period } T+1 \text{ demand } \bar{\alpha}; \\ 0 & \text{otherwise.} \end{cases}$.
$Q_{ij}^{\bar{\alpha}}$	:	the quantity of Product i produced in period j in the problem with period T+1 demand α.
$I_{ij}^{\bar{\alpha}}$	:	the inventory of Product i carried over from period j to $j+1$ in the problem with period $T+1$ demand $\bar{\alpha}$.
$Z_{j}^{\bar{\alpha}}$	:	$\begin{cases} 1 & \text{if at least one product is produced in period } j \text{ in} \\ & \text{the problem with period } T+1 \text{ demand } \bar{\alpha}; \\ 0 & \text{otherwise.} \end{cases}$.
d_{ij}	:	the demand for Product i in period j, $j=1,2,...,T$.

Problem TP-T-NSC: *Find a feasible solution to the following set of constraints:*

$$\sum_{j=1}^{T+1}\sum_{i=1}^{2}(k_i X_{ij}^{\bar{\alpha}} + KZ_j^{\bar{\alpha}}) + \sum_{j=1}^{T+1}\sum_{i=1}^{2} h_i I_{ij}^{\bar{\alpha}} = M_{T+1}(\bar{\alpha}) \qquad \bar{\alpha} \in \bar{\Lambda} \tag{3.13}$$

$$(\sum_{r=j}^{T+1} d_{ir})X_{ij}^{\bar{\alpha}} \geq Q_{ij}^{\bar{\alpha}} \qquad 1 \leq j \leq T+1, \bar{\alpha} \in \bar{\Lambda} \quad i = 1, 2. \tag{3.14}$$

$$I_{i(j-1)}^{\bar{\alpha}} + Q_{ij}^{\bar{\alpha}} - d_{ij} - I_{ij}^{\bar{\alpha}} = 0 \qquad 1 \leq j \leq T,\ \bar{\alpha} \in \bar{\Lambda}, \quad i = 1, 2. \tag{3.15}$$

$$I_{ij}^{\bar{\alpha}} + Q_{i(j+1)}^{\bar{\alpha}} - \bar{\alpha}_i^{T+1} - I_{i(j+1)}^{\bar{\alpha}} = 0 \qquad j = T, \bar{\alpha} \in \bar{\Lambda}, i = 1, 2. \tag{3.16}$$

$$Q_{i1}^{\bar{\alpha}} - Q_{i1}^{\bar{\alpha}'} = 0 \qquad \bar{\alpha}, \bar{\alpha}' \in \bar{\Lambda}, \bar{\alpha} \neq \bar{\alpha}', i = 1, 2. \tag{3.17}$$

$$Z_j^{\bar{\alpha}} \leq \sum_{i=1}^{2} X_{ij}^{\bar{\alpha}} \qquad 1 \leq j \leq T+1, \bar{\alpha} \in \bar{\Lambda} \quad i = 1, 2. \tag{3.18}$$

$$Z_j^{\bar{\alpha}} \geq 0.5 \sum_{i=1}^{2} X_{ij}^{\bar{\alpha}} \qquad 1 \leq j \leq T+1, \bar{\alpha} \in \bar{\Lambda}, \quad i = 1, 2. \tag{3.19}$$

$$I_{i0}^{\bar{\alpha}}, I_{i(T+1)}^{\bar{\alpha}} = 0 \qquad \bar{\alpha} \in \bar{\Lambda}, i = 1, 2. \tag{3.20}$$

$$I_{iT}^{\bar{\alpha}} = 0 \qquad \bar{\alpha} = 0,\ i = 1, 2. \tag{3.21}$$

$$I_{ij}^{\bar{\alpha}}, Q_{ij}^{\bar{\alpha}} \geq 0 \qquad 1 \leq j \leq T+1, \bar{\alpha} \in \bar{\Lambda}, i = 1, 2.$$

$$X_{ij}^{\bar{\alpha}} \in \{0, 1\} \qquad 1 \leq j \leq T+1, \bar{\alpha} \in \bar{\Lambda}, i = 1, 2.$$

Constraints (3.13) ensure that only those solutions with cost equal to the optimal cost are considered. Constraints (3.14)-(3.17) and (3.20)-(3.21) are similar in flavor to the constraints in Problem DLS-T-NSC. Constraints (3.18)-(3.19) ensure that $Z_j^{\bar{\alpha}} = 1$ if either $X_{1j}^{\bar{\alpha}}$ or $X_{2j}^{\bar{\alpha}}$ is 1.

3.4.3 Computational Results for the Two Variants

Assuming integer future demand, we summarize our experience with computing minimum forecast horizons for the two variants discussed above. Table 3.4.3 (resp. Table 3.4.3) shows the results for the warehouse-capacity constrained (resp. two-product) DLS variant. For all problem instances, demand was generated by setting the growth and variance parameters to their median values: $G = 1.00$ and $S = 0.50$ (see Section 3.3.3).

Table 3.4.3 summarizes the results of computing FH^1_{min} for the warehouse-capacity constrained variant. As in Section 3.3.3, the setup cost k was allowed to have eight values, ranging from 10 to 150. The warehouse capacity W was allowed to have three values: $0.4k$, $0.6k$ and $0.8k$. For each parameter setting, 11 problem instances were generated; each row of Table 3.4.3 corresponds to the average statistics over these 11 instances. Table 3.4.3 summarizes the results of computing FH^1_{min} for the two-product variant. The joint setup cost K was allowed to have five values: 2, 4, 6, 8 and 10. Each of the individual setup costs k_1 and k_2 was allowed to have five values: 0, $0.25K, 0.50K, 0.75K$ and K. The holding cost for the first product h_1 was set to 1 and the holding cost for the second product was allowed to have two values: 1 and 3. As before, each row of Table 3.4.3 corresponds to the average statistics over the 11 instances generated for each combination of the parameters. Note that FH^1_{min} typically decreases with a decrease in the warehouse capacity: in general, a smaller warehouse capacity results in more production periods, and consequently a smaller forecast horizon. For example, production occurs in each period in the extreme case of zero warehouse capacity and, hence, each period is a forecast horizon. Similarly, FH^1_{min} decreases with a decrease in either the individual or joint setup cost. Figure 3.6 plots the decrease in the length of FH^1_{min} with a decrease in the joint setup cost K.

Table 3.12: Forecast horizons, assuming integer future demand, for the warehouse-capacity constrained dynamic lot-size problem: non-stationary demand with $G = 1.000$ and $S = 0.50$.

k	Warehouse Capacity W											
	$0.4 * k$				$0.6 * k$				$0.8 * k$			
	Min.	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**	**Min.**	**Max.**	**Med.**	**Time (sec.)**
10	2	4	3	0.89	2	4	3	0.73	2	5	3	0.91
15	2	5	3	3.01	2	4	3	2.12	3	7	4	2.42
20	2	5	4	3.43	3	7	4	3.16	3	9	6	3.94
30	3	7	5	4.11	5	15	7	4.09	5	20	7	5.21
50	7	32	10	193.14	11	26	15	63.45	8	18	14	5.50
75	8	13	10	201.17	11	26	19	158.33	12	26	19	131.69
100	10	26	15	1211.38	12	28	21	1706.75	12	30	22	1863.43
150	10	36	15	1854.23	16	28	22	1987.41	16	31	24	2016.56

3.5 Concluding Remarks

This paper presents an analysis for the class of discrete forecast horizons obtained by assuming that future demands are integer multiples of a fixed positive real number. We use the dynamic lot-size problem to illustrate our analysis. Important structural results presented in this paper include (i) a characterization for a period to be a minimal discrete forecast horizon, (ii) sufficient conditions under which a discrete forecast horizon implies a continuous forecast horizon, and (iii) the impact on cost from assuming discrete future demands. We show, through an extensive computational study, that integer programming provides an efficient approach to compute and analyze discrete forecast horizons. Finally, we address two variants of the dynamic lot-size problem – a capacity-constrained warehouse scenario, and a two product DLS problem with joint and individual fixed costs – to demonstrate the simplicity of extending our approach.

This Chapter concludes our discussion on Forecast Horizons. In the next Chapter we look at the problem of traffic grooming in optical networks. We formulate the traffic grooming problem as a mixed-integer program. We prove that problem is NP-Hard. The solution method we use is an heuristic method using column generation.

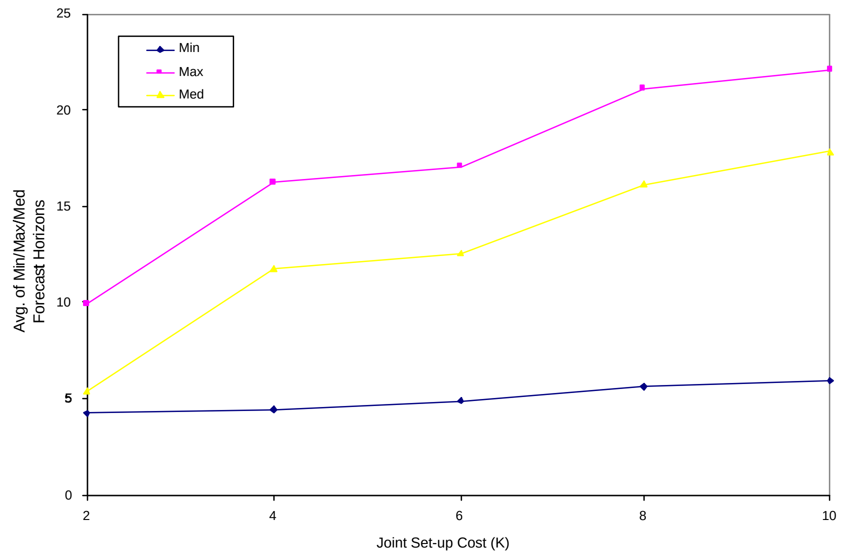

Figure 3.6: The average of the minimum, median and maximum lengths of FH^1_{min} as a function of the joint setup cost K for the two-product DLS variant.

Table 3.13: Forecast horizons, assuming integer future demand, for the two-product dynamic lot-size problem with individual and joint setup costs: non-stationary demand with $G = 1.000$ and $S = 0.50$.

h_1	h_2	k_1 (%K)	k_2 (%K)	K	**Min.**	**Max.**	**Med.**	**Time (sec.)**
1	1	0	0		2	3	2	0.4
1	1	25	25		2	7	4	31.2
1	1	50	50		2	11	4	49.1
1	1	75	75		3	10	6	121.7
1	1	100	100	2	4	10	7	241.3
1	3	0	0		2	4	2	3.1
1	3	25	25		4	12	4	4.5
1	3	50	50		6	13	5	14.2
1	3	75	75		7	13	8	109.2
1	3	100	100		10	16	12	184.6
1	1	0	0		2	3	2	0.3
1	1	25	25		2	11	9	124.2
1	1	50	50		2	13	9	145.1
1	1	75	75		4	15	10	1938.1
1	1	100	100	4	6	16	11	2017.7
1	3	0	0		2	6	3	5.0
1	3	25	25		4	16	11	8.4
1	3	50	50		6	21	17	45.2
1	3	75	75		8	29	21	904.4
1	3	100	100		8	33	25	934.8
1	1	0	0		2	3	2	0.4
1	1	25	25		2	11	8	934.1
1	1	50	50		4	14	9	1031.0
1	1	75	75		5	16	11	2412.2
1	1	100	100	6	5	16	12	2398.3
1	3	0	0		2	5	3	102.1
1	3	25	25		7	21	10	234.1
1	3	50	50		6	24	20	957.4
1	3	75	75		8	26	22	1005.3
1	3	100	100		8	35	29	1928.3
1	1	0	0		2	5	3	0.5
1	1	25	25		3	11	8	1742.1
1	1	50	50		6	14	8	3029.9
1	1	75	75		6	16	12	4352.2
1	1	100	100	8	6	16	14	5072.1
1	3	0	0		2	7	4	164.0
1	3	25	25		6	29	21	299.2
1	3	50	50		5	35	23	1084.2
1	3	75	75		10	37	32	1424.1
1	3	100	100		10	41	37	2331.5
1	1	0	0		2	7	3	1912.3
1	1	25	25		3	14	10	4124.2
1	1	50	50		3	16	10	4940.1
1	1	75	75		5	16	12	4103.0
1	1	100	100	10	5	17	15	5693.2
1	3	0	0		2	9	5	1002.9
1	3	25	25		6	29	25	1348.7
1	3	50	50		9	31	27	1902.3
1	3	75	75		9	39	34	2490.2
1	3	100	100		15	43	38	2993.1

CHAPTER 4

A TRAFFIC GROOMING ALGORITHM FOR WAVELENGTH ROUTED OPTICAL NETWORKS

4.1 Synopsis

We start by illustrating the Traffic Grooming problem with an example network shown in Figure 4.1. The traffic routing problem is on a mesh-network with 7 nodes and a link capacity of two lightpaths in each direction. Each node consists of two transmitters and receivers. The requested traffic connections are shown in Table 4.1.

Table 4.1: Example problem: Traffic connections required between pair of nodes.

Origin Node	*Dest.Node*	*No. of connections requested*
1	6	2
2	5	2
3	4	1
5	7	2
6	2	1

Figures 4.1(a) and 4.1(b) indicate (by dotted lines) two feasible routings of lightpaths. The number corresponding to each lightpath is the number of requests carried by the lightpath. The routing in Figure 4(a) allocates a total of 7 connection requests. Since the capacity of link 3-4 is two, this routing is able to satisfy only one connection request from node 1 to node 6. The routing shown in Figure 4(b) accommodates all (i.e., a total of 8) connection requests.

In this chapter, we consider the problem of traffic grooming and routing in optical mesh networks. The objective is throughput maximization subject to constraints on the maximum link-load and the number of transceivers available at each node. We

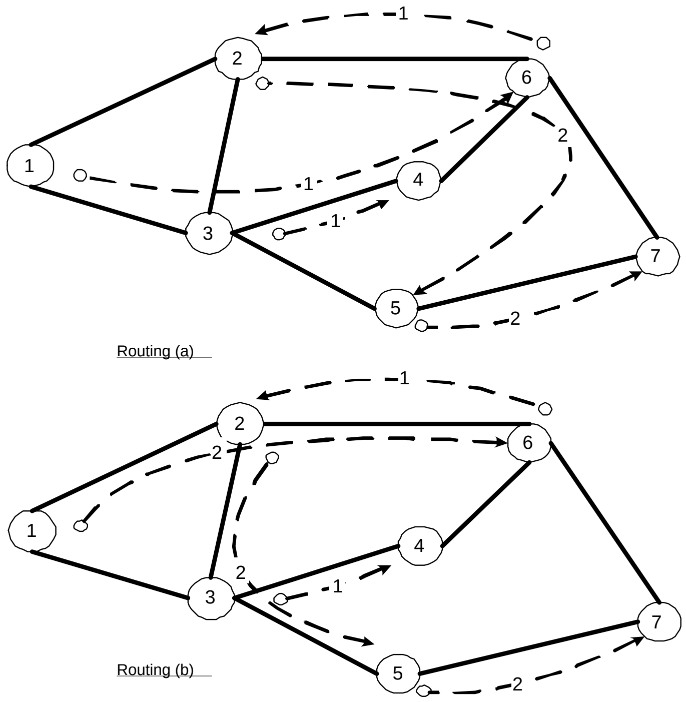

Figure 4.1: A six node network with two transmitters and two receivers at each node.

develop a heuristic based on a column generation technique and verify the quality of the heuristic solution through a upper bound computed using a Lagrangian relaxation. We start by presenting a review of relevant literature in Section 4.2. In Section 4.3, we present the notation and a mathematical formulation for grooming in all-optical networks with the objective of traffic maximization. We then describe the column generation based algorithm. In Section 4.4, we provide a Lagrangian based technique for attaining a tight upper bound on the objective. Section 4.5 provides a computational study and discusses the performance of our solution procedure. Finally, Section 4.6 concludes the study.

4.2 Literature Review

The traffic grooming and wavelength assignment literature has focused on a large number of different objectives, such as network throughput maximization, minimization of blocking probability, minimization of the total number of wavelengths, minimization of total route distance and network cost, among others. We focus on the objective of traffic maximization and therefore restrict our review of the literature to this (and related) areas of research.

Our work is closest to that of Zhu and Mukherjee [110], who also provide mathematical models and solution procedures for grooming in all-optical networks with an objective of traffic maximization. While Zhu and Mukherjee [110] consider both single and multi-hop all-optical networks, they provide a greedy heuristic without any means of evaluating the quality of the solution achieved. Further, the authors restrict their attention to networks of a relatively small size of five nodes. While we focus only on single-hop networks, the heuristic procedure we suggest in this work is different due to the fact that it provides verifiably high quality solutions for large sized networks.

Zhu et al. [109] propose a generic network model for traffic grooming in heterogeneous mesh-networks. The model can be adapted for various objectives by using different grooming policies such as the number of transceivers available, the number of wavelengths available and wavelength conversion capabilities. The authors also address the problem of partial wavelength conversion capabilities and different grooming capabilities at each node. They propose an integrated grooming procedure and traffic selection scheme where connection requests are routed one request at a time. For static grooming (i.e., when all the traffic demands are known in advance) such a technique will be influenced by the order in which requests are routed. The paper proposes two traffic request selection schemes, Least Cost First and Maximum Utilization First. The Least Cost First scheme chooses the most cost-effective traffic request under the current network state and routes it. The paper defines the cost of a traffic request as the weight of the shortest path for routing the traffic on the corresponding auxiliary network divided by the amount of the traffic (which is computed as the granularity multiplied by the units of the traffic). The Maximum Utilization First scheme selects the connection with the highest utilization (the total amount of the request divided by the number of hops from the source to the destination on the physical topology).

Hu and Leida [56] consider traffic grooming in combination with traffic routing and wavelength assignment (GRWA). Their objective is to minimize the total number of transponders required in the network. The paper presents an integer programming (IP) formulation for GRWA. Further, they introduce a decomposition method to divide the GRWA problem into two smaller problems - Traffic Grooming (GR) and Wavelength Assignment (WA). A relaxed IP formulation of problem GR is solved assuming that the lightpaths and their routes in the physical layer are given. For a given solution to the GR problem, the WA problem is solved using a sequential node coloring scheme.

Lee et al. [67] consider the routing and wavelength assignment problem with the objective of minimizing the number of wavelengths used, and provide an algorithm based on a column generation technique. However in existing optical networks, the wavelength minimization is not a critical requirement because of two reasons. First, the use of additional wavelengths has been found to only marginally increase the overall network costs (Zang et al. [107]). Second, with the enhancements in technology, the wavelength capacity of optical networks has dramatically increased in recent years.

Mukherjee [75], presents a multi-commodity flow formulation to minimize the link-load in GRWA. A randomized rounding technique is used to route the lightpaths. This technique relies on the assumption that there can be at most one lightpath for any source-destination pair. Also, limits on the number of transceivers at each node are not considered.

Ozdalgar et al. [82] provide an algorithmic approach for routing and wavelength assignment (RWA) in optical networks with full wavelength conversion. They address the routing and wavelength assignment problem jointly. Their integer programming formulation is based on multi-commodity network flows. The objective is to minimize the link load in order to minimize blocking probability. By adding penalty function to the objective, they also address the problem of infeasibility of the formulation when some nodes have only sparse wavelength conversion capability.

4.3 Problem Notation and ILP Formulation

Consider a network *G=(N,A)*, where N is the set of nodes (wavelength routed switches) and A is the set of arcs which are directed fiber links between nodes. We make the following assumptions:

1. Each arc consists of a single fiber link between nodes in a mesh topology; a link corresponds to an edge of the network G.
2. If arc $(i,j)\epsilon A$, then arc $(j,i)\epsilon A$.
3. Each fiber link can carry up to $\mathcal{C}$ wavelengths (lightpaths). This is referred to as the *maximum link load* allowed in the network.
4. There is no wavelength conversion capability available in the wavelength-routed switches at the nodes of the network.
5. Each node contains a pre-specified number of transmitters and receivers which are tunable to any wavelength on the fiber.
6. We are given a set of connection requests where each such request is represented by a tuple, $[(i,j),q]$. Here $i,j \in N$ are respectively the source and destination of the connection and $q \in Q = \{1,3,12,48\}$ is the type of the connection request. If the connection type is q, the bandwidth requirement for the connection is OC-q. Let $o(q)$ be the bandwidth requirement of connection type $q \in Q$ in multiples of OC-1 (ie. 51.84 Mbps). We assume that a connection request between two nodes is indivisible. That is, it must be routed in one single ligthpath and cannot be divided into several lower speed connection requests and routed separately.
7. Each wavelength-routed switch has unlimited capacity to multiplex/demultiplex channels. This implies that any number of lower speed connection requests of

type $q \in Q$ can be multiplexed (demultiplexed) to (from) a lightpath originating (terminating) at that node. Each lightpath however has a finite given capacity $\mathcal{M}$ (e.g.: we assume lightpath capacity as OC-48 which is 48×51.84 Mbps or 2.5 Gbps).

8. Each communicating pair of nodes (i, j) can be considered to be a specific commodity $r \in R$. Hence for each such commodity there can be upto $|Q|$ different kinds of connection requests. Let $n_q^r \in \mathcal{Z}^+$ be the number of connection requests of bandwidth OC-q ($q \in Q$) for commodity r. A lightpath is therefore a path for commodity $r \in R$, which uses a particular wavelength of light for transmission.

We can now formally define the traffic grooming problem as follows: we seek to assign a set of lightpaths to each commodity r such that the bandwidth of communication requests assigned to these lightpaths is maximized.

4.3.1 Multi-commodity flow formulation

In this section we provide a multi-commodity formulation for the traffic grooming problem. For this, we need to aggregate demand requests d_r for each commodity $r \in R$ which are computed as follows:

$$d_r = \sum_{q \in Q} o(q).n_q^r \tag{4.1}$$

The aggregate demand d_r represents the total bandwidth requirement for commodity r in multiples of OC-1. For example, commodity r has the following demand requests OC-1: 15, OC-3: 2,OC-12: 1 and OC-48: 1. Therefore, aggregate demand for commodity r is;

$$d_r = 1 \times 15 + 3 \times 2 + 12 \times 1 + 48 \times 1 = 81$$

Note that the number of lightpaths required to fully satisfy the demand $d_r = 81$ is, $\lceil \frac{d_r}{\mathcal{M}} \rceil = \lceil \frac{81}{48} \rceil = 2$, where $\mathcal{M} = 48$ is the capacity of a lightpath.

We use the following notation for formulation of the traffic grooming problem.

Parameters

$\mathcal{C}$	:	Link capacity in number of wavelengths (lightpaths)
R	:	Set of all commodities, $R = \{1, 2, ...\|N\|.(\|N\| - 1)\}$
P_r	:	Index set of all lightpaths in G for $r \in R$. Note that, two lightpaths could have identical physical paths, in which case they would require distinct wavelengths.
δ_{mn}^{rp}	:	$\begin{cases} 1 & \text{if link } (m,n) \text{ is on path } p \text{ of } r \in R,\ (m,n) \in A \\ 0 & \text{otherwise} \end{cases}$
$\mathcal{M}$	:	Capacity of one lightpath in number of wavelengths (in OC-48)
d_r	:	Aggregated demand for commodity $r \in R$
TT_u	:	Number of transmitters at node $u \in N$
RR_v	:	Number of receivers at node $v \in N$
R_u^o	:	Set of commodities originating at node $u \in N$
R_v^d	:	Set of commodities terminating at node $v \in N$

Decision Variables

f_{rp}	:	flow on p^{th} path of commodity $r \in R$
X_{rp}	:	$\begin{cases} 1 & \text{if path } p \in P_r \text{ for commodity } r \in R \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$

Problem Groom_MC:

$$Max \sum_{r \in R} \sum_{p=1}^{|P_r|} f_{rp} \tag{4.2}$$

s.t.

$$\begin{aligned}
\sum_{r \in R} \sum_{p=1}^{|P_r|} \delta_{mn}^{rp} X_{rp} &\leq \mathcal{C}, \quad \forall\, (m,n) \in A && (4.3)\\
f_{rp} &\leq \mathcal{M} X_{rp}, \quad \forall\, r \in R,\ \forall p \in P_r && (4.4)\\
\sum_{p=1}^{|P_r|} f_{rp} &\leq d_r, \quad \forall\, r \in R && (4.5)\\
\sum_{r \in R_u^o} \sum_{p=1}^{|P_r|} X_{rp} &\leq TT_u, \quad \forall\, u \in N && (4.6)\\
\sum_{r \in R_v^d} \sum_{p=1}^{|P_r|} X_{rp} &\leq RR_v, \quad \forall\, v \in N && (4.7)\\
X_{rp} &\in \langle 0, 1 \rangle, \quad \forall\, i \in R,\ \forall p \in P_r && (4.8)\\
f_{rp} &\in \mathcal{Z}_+, \quad \forall\, r \in R,\ \forall p \in P_r && (4.9)
\end{aligned}$$

The objective function maximizes the total traffic flow of all commodities. Constraint 4.3 ensures that the number of lightpaths supported by each link is limited to $\mathcal{C}$. Constraint 4.4 limits the flow supported by each lightpath to the capacity of the lightpath. Constraint 4.5 enforces that for a commodity, the total flow over all lightpaths must be less than or equal to the total aggregated demand for the commodity. Constraint 4.6 (4.7) limits the total lightpaths generated (received) by a node to total transmitters (receivers) at the node.

The following properties of *Groom_MC* enable us to solve the problem more efficiently.

Property 1 Given a solution to *Groom_MC*, where total demand routed for commodity r is $\bar{d}_r = \sum_p f_{rp}$, there always exists a feasible allocation of all original connection requests $[(i,j),q]$ (where each (i,j), pair is a commodity $r \in R$), such that allocation is equivalent to $\bar{d_r}$. Given two numbers $n_1, n_2 \in \{\mathcal{Z}^+\}$, $\bar{d_r}$ can be represented as,

$$\bar{d_r} = 48n_1 + n_2 \quad , n_2 < 48, \quad n_1, n_2 \in \{\mathcal{Z}^+\}$$

There can be three possibilities:

1. $\bar{d_r} = 0$
2. $\bar{d_r} < d_r$
3. $\bar{d_r} = d_r$

In case 1, no lightpaths are allocated to commodity r. In cases 2 and 3, if the solution allocates $\chi, \chi \in \{\mathcal{Z}^+\}$ lightpaths to commodity r, then the total bandwidth of the lightpaths allocated to this commodity (in multiples of OC-1's) is, 48χ. Clearly if $48\chi \geq d_r$, the solution to problem *Groom_MC* satisfies all original requests for commodity r, i.e. case 3. If $48\chi < d_r$, we are only able to satisfy some subset of requests (i.e. case 2). Given the number χ, we can see that the problem of allocating the original bandwidth requests of OC-q ($q \in Q$) using these lightpaths is similar to a simple bin-packing problem with bin capacity 48 and item sizes {1,3,12,48}. In our case bin-capacity is an integer multiple of the item sizes and the number of item sizes is fixed, therefore it is possible to solve this bin-packing problem in polynomial time. Note that bin-packing problem is solvable in polynomial time even when the number of item size is arbitrary and the item sizes are integer multiples (Coffman et al. [29]).

Property 2 Since link capacity and demand at each node are integer valued, there exist an optimal solution with integer flow values. Therefore, we can relax the integrality constraint $f_{rp} \in Z^+$ to $f_{rp} \geq 0$.

Proof: Consider Problem *Groom_MC* with constraints (4.9) replaced by $f_{rp} \geq 0$, $\forall r \in R$, $\forall p \in P_r$. Let S denote an optimal solution vector for this modified problem corresponding to a corner point of the convex hull of feasible solutions. We show that $f_{rp} \in \mathcal{Z}_+$, $\forall r \in R$, $\forall p \in P_r$.

Since S is a feasible solution, it satisfies the following inequalities:

$$\begin{aligned} f_{rp} &\leq \mathcal{M}X_{rp}, \quad \forall\, r \in R,\ \forall p \in P_r \\ \sum_{p=1}^{|P_r|} f_{rp} &\leq d_r, \quad \forall\, r \in R \end{aligned}$$

Consider the following two possibilities for a commodity r:

1. S contains exactly one non-integer flow, say f_{r1}. In this case, since $\mathcal{M}$ and d_r are integers, we have

   $$\begin{aligned} f_{r1} &< \mathcal{M}X_{r1} \\ \sum_{p=1}^{|P_r|} f_{rp} &< d_r. \end{aligned}$$

   Hence, there exists $\epsilon > 0$ such that increasing f_{r1} to $f'_{r1} = f_{r1} + \epsilon$ maintains the feasibility of the solution. This contradicts the optimality of S.

2. S contains two or more non-integer flows. Consider two non-integer flows, say f_{r1} and f_{r2}. Note that, $\mathcal{M} > f_{r1}$ and $\mathcal{M} > f_{r2}$. Let $\epsilon = \min\{f_{r1}, f_{r2}, \mathcal{M} - f_{r1}, \mathcal{M} - f_{r2}\}$. Keeping all other variables fixed, two feasible solution vectors S_1 and S_2 can be obtained by changing f_{r1} and f_{r2} as follows:

   Solution S_1: Increase f_{r1} by ϵ and reduce f_{r2} by ϵ,

   $$(f'_{r1} = f_{r1} + \epsilon, f'_{r2} = f_{r2} - \epsilon).$$

   Solution S_2: Increase f_{r2} by ϵ and decrease f_{r1} by ϵ,

   $$(f''_{r1} = f_{r1} - \epsilon, f''_{r2} = f_{r2} + \epsilon).$$

   It is easy to see that both S_1 and S_2 are feasible solutions. However, $S = \frac{S_1+S_2}{2}$. This contradicts the fact that S is an extreme point solution. The result follows.

   □

Next we will address the complexity of the problem *Groom_MC*.

Theorem 4.3.1 *The recognition version of Problem Groom_MC is NP-complete in the strong sense.*

Proof: Even et al. [39] proves that the directed two commodity integral flow problem is *NP-complete* in the strong sense.

Directed two commodity integral flow problem:

Instance: Directed network $G = (V, A)$, specific vertices s_1,s_2,t_1 and t_2, capacity $c(a) \in Z^+$ for each $a \in A$, requirement R_1,$R_2 \in Z^+$.

Question: Are there two flow functions f_1,f_2:$A \rightarrow Z_0^+$ such that,

1. For each $a \in A$, $f_1(a) + f_2(a) \leq c(a)$,
2. For each $\nu \in V - \{s, t\}$ and $i \in \{1, 2\}$, flow f_i is conserved at ν, and
3. For $i \in \{1, 2\}$, the net flow into t_i under flow f_i is at least R_i?

We can reduce problem *Groom_MC* to the two commodity integral flow problem as shown below. Consider the following instance of the problem *Groom_MC*:

A connected directed network $G = (V, A)$, link capacity $c(a) = \mathcal{C}$ for each $a \in A$, nodes $s_1 \in V$ and $t_1 \in V$ having R_1 transmitters and R_1 receivers respectively, and nodes $s_2 \in V$ and $t_2 \in V$ having R_2 transmitters and R_2 receivers, respectively. The demand vector is,

$$d_r : \begin{cases} R_1 & \text{if } i = s_1, j = t_1 \\ R_2 & \text{if } i = s_2, j = t_2 \\ 0 & \text{otherwise} \end{cases}$$

where, $(i, j) \rightarrow r \in R$, $(i, j) \in V$, $i \neq j$. The demand is given in number of lightpaths.

The decision question is: Does there exists a solution with objective value $\geq R_1 + R_2$?

The above restricted version of problem *Groom_MC* is equivalent to directed two commodity integral flow problem with link capacity $\mathcal{C}$, $s_1(s_2)$ and $t_1(t_2)$ being the source and terminal nodes of commodity 1(2) respectively, and commodity 1(2) having a demand requirement of $R_1(R_2)$. Clearly since the directed two commodity integral flow problem is *NP-complete* in the strong sense, problem *Groom_MC* is also *NP-complete* in the strong sense. □.

4.3.2 Solving *Problem Groom_MC*

The previous section proved problem *Groom_MC* is NP-Hard in the strong sense. Hence, there is probably no polynomial algorithm to solve the problem optimally (Garey and Johnson [45]). For example, Table 4.3.2 shows results of running the problem *Groom_MC* for 3600 seconds on nine problem instances, by using Cplex 8.1 solver on a Pentium IV, 2.4GHz, 512 MB RAM computer. To find all the paths for each commodity a simple depth-first search was used.

Typically, WDM-based all-optical networks are sparse when they operate over a wide geographical area. Given this, a number of methods have been proposed by researchers to emulate the topology of such networks. The general principle behind these techniques is that of distributing vertices at random locations in a plane and then adding edges between pairs of vertices based on a specific probability distribution. For Table 4.3.2, the networks were generated by the procedure outlined in Zegura et al., [108]. Specifically, we used the probability function, $\alpha e^{\frac{-d}{(L-d)}}$, which relates edge probability to distance between edges. The probability of an edge in this model decreases exponentially with the distance between the two vertices. Following Zegura et al., [108], the parameters α and L was set at 0.06 and 141 respectively. The variable d is the distance between the two nodes under consideration.

The columns 5 and 6 in Table 4.3.2 give the optimal solutions to *Groom_MC* and its LP relaxation respectively. From the computational results, it is clear that on average IP formulation *Groom_MC* can only be solved for networks with a maximum of 10 nodes within the specified time limit of 3600 seconds. In bigger networks (more than 14 nodes), even the LP relaxation of the problem could not be solved within

3600 seconds (Table 4.3.2). For example, LP relaxation for a twenty node network problem given in Table 4.4.1 took more than five hours to solve. When the network size increases the number of paths generated become prohibitively large. For example, the 16 node network in Table 4.3.2 have a total of 37,821 directed paths. These paths will introduce $5 \times 37,821$, (0,1) variables to the formulation. The last two columns in Table 4.3.2, *Groom_Agg* and *FSA*, represent respectively the solution from an alternative formulation and a proposed Feasible Solution Algorithm (FSA) that we introduce later in the paper.

Table 4.2: Results for 3600 seconds runs by using CPLEX 8.1.

Problem	*Network Size*		*Problem Size*			*Objective Value*			
Instance	*No. Nodes*	*No. Links*	*Constraints*	*Variables*	*Results*	*Groom_MC*	*LP*	*Groom_Agg*	*FSA*
1	6	7	1890	6210	Optimal	806	913	806	792
2	7	13	1921	7140	Optimal	1031	1206	1031	968
3	8	11	2043	8903	Optimal	1586	1781	1586	1507
4	10	16	14520	60001	feasible IP sol.	2392	2914	–	2266
5	12	22	18323	65731	No IP sol.	–	5312	–	4126
6	14	29	18920	66391	No IP sol.	–	6881	–	5062
7	16	33	19034	78181	No IP or LP sol.	–	–	–	9143
8	18	41	20193	93858	No IP or LP sol.	–	–	–	15214
9	20	56	21284	249231	No IP or LP sol.	–	–	–	18006

Given the results in Table 4.3.2, our approach is to find an efficient heuristic procedure which can be used to get near optimal solutions in a reasonable time. Note that, Problem *Groom_MC* involves the establishment of lightpaths to facilitate routing of demand requests of the commodities. However, finding best possible routes to generate these lightpaths is very difficult because the number of paths increases exponentially with the network size. Therefore, we use a well known technique in the literature - *column generation* - to generate these paths efficiently. The following three sub-sections explain our solution approach in detail. Sub-section 3.2.1 describes how to price out and choose profitable paths as possible candidates to support lightpaths. Following this, we describe the column generation procedure and a heuristic algorithm to obtain solutions to Problem *Groom_MC* in subsections 3.2.2 and 3.2.3 respectively.

Pricing procedure

Consider the linear programming relaxation of Problem *Groom_MC*. We call this Problem *Groom_MCR*. We associate non-negative dual multipliers $\lambda_{mn}, \mu_p^r, \nu_r, \xi_u^o$ and ϕ_v^d for link capacity(4.3), lightpath capacity(4.4),commodity demand(4.5) and two transceiver constraints(4.6 and 4.7), respectively. Let problem *Groom_MCRD* be the dual of problem *Groom_MCR*. Hence, *Groom_MCRD* has the following constraints:

$$\mu_p^r + \nu_r \geq 1, \quad \forall\, p \in P_r,\ \forall\, r \in R \tag{4.10}$$

$$\sum_{\forall (m,n)\in A} \lambda_{mn}\delta_{mn}^{rp} - \mathcal{M}\mu_p^r + \xi_s^o + \phi_t^d \geq 0, \quad \forall\, p \in P_r,\ \forall\, r \in R \tag{4.11}$$

$$\lambda_{mn}, \mu_p^r, \nu_r, \xi_u^o, \phi_v^d \geq 0 \tag{4.12}$$

where s and t are the origin and destination respectively of path p.

The following procedure is used to solve Problem *Groom_MCR*. First, a restricted version of Problem *Groom_MCR (Groom_MCRR)* is solved. This is done by taking a subset of paths Ψ for each commodity. The initial set of paths are generated by getting the shortest path for each pair of nodes (i.e. each commodity) and set these shortest paths as the initial set in the column generation procedure.

For each solution to Problem *Groom_MCRR*(Ψ), we price-out non-basic columns in the following manner.

A path p for commodity r is potentially profitable if it satisfies the following two inequalities:

$$1 - \nu_r > \mu_p^r, \ p \in P_r, r \in R \tag{4.13}$$

$$\mathcal{M}\mu_p^r - \xi_s^o - \phi_t^d > \sum_{\forall (m,n)\in A} \lambda_{mn}\delta_{mn}^{rp}, \ p \in P_r, r \in R \tag{4.14}$$

Where, s and t are the origin and destination nodes respectively of commodity $r \in R$. The set of paths satisfying (4.15) below will be a superset of those satisfying (4.13) and (4.14).

$$\mathcal{M}(1-\nu_r) - \xi_s^o - \phi_t^d \quad > \sum_{\forall (m,n)\in A} \lambda_{mn}\delta_{mn}^{rp}, \; p \in P_r, r \in R \tag{4.15}$$

By simplifying (4.15) we can state that, a potentially profitable path for commodity r satisfies (4.16).

$$\zeta(r) \quad > \quad \mathcal{L}(r,j), \quad p \in P_r, r \in R \tag{4.16}$$

where $\mathcal{M}(1-\nu_r) - \xi_s^o - \phi_t^d = \zeta(r)$ and $\sum_{\forall (m,n)\in A} \lambda_{mn}\delta_{mn}^{rp} = \mathcal{L}(r,j)$ is the length of path p with arc weights λ_{mn}.

We find the shortest path distance to each node pair (hence, commodity) with link weights λ_{mn}. If none of the shortest paths satisfy (4.16), there are no profitable paths left to be added to Ψ. Therefore we have an optimal solution to Problem *Groom_MCR*.

Column Generation Procedure

Step 1 Find the aggregate demand d_r for each commodity r.

Step 2 Solve a restricted version of Problem *Groom_MCR*
(i.e., Problem *Groom_MCRR*(Ψ)), and let the basis obtained be β.

Step 3 Calculate shortest paths and check optimality of β for
Problem *Groom_MCR* using equation (4.16). If basis is optimal goto Step 6.

step 4 Identify profitable columns and add to the set of paths (Ψ) in
Groom_MCRR(Ψ).

Step 5 Solve the modified *Groom_MCRR*(Ψ) and get an improved basis.
Goto step 3.

Step 6 Output solution to *Groom_MCRR*(Ψ) and set of columns (Ψ). End.

Note that the above procedure will result in a solution that optimally solves problem *Groom_MCR*. If this solution is integral then it is also an optimal solution for Problem *Groom_MC*. In general however, it is possible to obtain a good (but not necessarily optimal) integral solutions by using the algorithm provided in the next section.

Feasible Solution Algorithm

The optimal solution to the above column generation procedure finds a set of paths(Ψ), which can be used to support lightpaths. However, the solution may not be feasible to Problem *Groom_MC* because variable X_{rp} may be non-integer. Therefore, to find a feasible solution to Problem *Groom_MC* we limit the number of available paths to the set Ψ. Since, Ψ is a small fraction of all possible number of paths, the IP-Problem *Groom_MC*- can be solved quickly. We complete the algorithm by executing the following two steps after the column generation procedure.

Step 7 Take the set of paths Ψ obtained in Step 6 to formulate a restricted version of Problem *Groom_MC* called Problem *Groom_MC*(Ψ).

Step 8 Solve Problem *Groom_MC*(Ψ) to optimality.

This heuristic allocates a set of lightpaths to each commodity $r \in R$. Since we aggregate the original demand to d_r it is clear from our discussion in section 3.1, that a feasible disaggregated solution from this is easy to obtain. In general one way to gauge quality of the solution to Problem *Groom_MC*(Ψ) would be to measure its gap with the solution to Problem *Groom_MCR*. However, this gap may not necessarily be very tight due to the presence of a duality gap for problem Problem *Groom_MC*. In the next section, we therefore develop a technique to get tighter upper bounds for the Problem *Groom_MC*.

4.4 Upper Bounding Procedure

The following formulation for the single-hop case is adapted from Zhu and Mukheerjee [110]. Our modified formulation does not take into account wavelength assignment for each light path. This modification ensures that Problem *Groom_Agg* is equivalent to Problem *Groom_MC*.

Notation

Variables

S_{ij}^{qt}	:	$\begin{cases} 1 & \text{if } t^{th} \text{ request of bandwidth } q \in Q \text{ bet. } (i,j) \text{ is successfully routed} \\ 0 & \text{otherwise} \end{cases}$
		where $t = 1, 2.., n_q^r$ and $i = o.(r),\ j = d.(r),\ r \in R$
V_{ij}	:	Number of lightpaths from node i to j, $i = o.(r),\ j = d.(r),\ r \in R$
P_{mk}^{ij}	:	No. of lightpaths bet. nodes (i,j) routed through fiber link, $(m,k) \in A$

Problem Groom_Agg:

$$Max \sum_{q \in Q, i, j, t} o(q) \times S_{ij}^{qt} \tag{4.17}$$

subject to:

$$\sum_{j \in N} V_{ij} \leq TT_i, \quad \forall\, i \tag{4.18}$$

$$\sum_{i \in N} V_{ij} \leq RR_j, \quad \forall\, j \tag{4.19}$$

$$\sum_{m \in N} P_{mk}^{ij} = \sum_{n \in N} P_{kn}^{ij}, \quad if\ k \neq i, j \quad \forall\, i, j, k \quad (m, k), (k, n) \in A \tag{4.20}$$

$$\sum_{m \in N} P_{mi}^{ij} = 0, \quad \forall\, i, j \quad (m, i) \in A \tag{4.21}$$

$$\sum_{n \in N} P_{jn}^{ij} = 0, \quad \forall\, i, j \quad (j, n) \in A \tag{4.22}$$

$$\sum_{n \in N} P_{in}^{ij} = V_{ij}, \quad \forall\, i, j \quad (i, n) \in A \tag{4.23}$$

$$\sum_{m \in N} P_{mj}^{ij} = V_{ij}, \quad \forall\, i, j \quad (m, j) \in A \tag{4.24}$$

$$\sum_{i, j \in N} P_{mn}^{ij} \leq \mathcal{C}, \quad \forall\, m, n \quad (m, n) \in A \tag{4.25}$$

$$\sum_{q \in Q, t} o(q) \times S_{ij}^{qt} \leq V_{ij} \times \mathcal{M}, \quad \forall\, i, j \tag{4.26}$$

$$V_{ij} \in \mathcal{Z}^+ \tag{4.27}$$

$$P_{mn}^{ij} \in \mathcal{Z}^+ \tag{4.28}$$

$$S_{ij}^{qt} \in \{0, 1\} \tag{4.29}$$

The objective (4.17) is to maximize the total bandwidth (traffic) routed. Constraint (4.18) limits the number of lightpaths beginning at node $i (\in N)$ to the number of transmitters at the node (TT_i). Similarly, constraint (4.19) limits the number of lightpaths ending at node $j (\in N)$ to the number of receivers at the node (RR_j). Constraint (4.20) is the lightpath continuity constraint. The constraint states that, for every lightpath between node i and j entering an intermediate node k, an equal number leave the node k. Constraint (4.21) says that zero lightpaths end at origin node i. Similarly, (4.22) says that zero lightpaths begin at destination node j. Constraint (4.23)(constraint (4.24)) limits the number of lightpaths originating (ending) at node $i(j)$ to V_{ij}. Constraint (4.25) limits the number of lightpaths passing through link $(m, n) \in A$ to the maximum link load $\mathcal{C}$. Constraint (4.26) limits the bandwidth allocated to nodes (i, j) to the capacity of the total lightpaths generated between nodes (i, j).

Theorem 4.4.1 *The two IP formulations,* Groom_MC *and* Groom_Agg *are equivalent when $o(q)$ are integer multiples, where $q \in Q$.*

Proof: We prove the theorem by showing that, any optimal solution to problem *Groom_Agg* can be converted into a feasible solution to problem *Groom_MC* and vice versa. Let an optimal solution to *Groom_Agg* be $\bar{S}_{ij}^{qt}$ and the total number of lightpaths allocated $\sum_{\forall\{(i,j)\in N,q,t\}} \bar{V}_{ij}$, where $\bar{V}_{ij}$ is the number of lightpaths allocated to node pair (i, j). We can find a feasible solution to *Groom_MC* by allocating the same lightpath routing. This solution is feasible to *Groom_MC* because $\bar{V}_{ij}$ is feasible under the transceiver and maximum link-load constraints. The solution value of *Groom_MC* is;

$$\sum_{p} \bar{f}_{rp} = \sum_{q,t} o(q)\bar{S}_{ij}^{qt},$$

where in this instance, commodity r is the node pair (i, j). Since $\sum_p \bar{f}_{rp}$ is an integer number we can easily convert the $\sum_p \bar{f}_{rp}$ solution into the original demand requests for node pair (i, j) as discussed in Property 1.

Similarly, an optimal solution to *Groom_MC* $(\tilde{f}_{ip})$ can be converted to a feasible solution of *Groom_Agg* $(\tilde{S}_{ij}^{qt})$ with,

$$\sum_{q,t} o(q)\tilde{S}_{ij}^{qt} = \sum_{p} \tilde{f}_{rp}, \; \forall (i, j) \in \{N, q, t\}.$$

Therefore, the formulations *Groom_MC* and *Groom_Agg* are equivalent. $\square$

4.4.1 Lagrangian Relaxation

We relax constraints (4.19),(4.25) and (4.26) and multiply them by non-negative Lagrangian multipliers γ_j,α_{mn}, and β_{ij}, and include them in the objective function to get the relaxed problem *Groom_AggR*.

Problem Groom_AggR:

$$Max \sum_{q,i,j,t} o(q) \times S_{ij}^{qt} + \sum_{m,n} \alpha_{mn}\{\mathcal{C} - \sum_{i,j} P_{mn}^{ij}\} + \sum_{i,j} \beta_{ij}\{V_{ij} \times \mathcal{M} - \sum_{q,t} o(q) \times S_{ij}^{qt}\}$$
$$+ \sum_{j} \gamma_j\{RR_j - \sum_{i} V_{ij}\}$$
$$subject\ \ to: (4.18), (4.20), (4.21), (4.22), (4.23), (4.24), (4.27), (4.28), (4.29).$$

Problem *Groom_AggR* can be decomposed into two sub-problems as shown below.

Sub-Problem 1

$$Max \sum_{q,i,j,t} (1 - \beta_{ij}) o(q) \times S_{ij}^{qt} \tag{4.30}$$
$$subject\ \ to: (4.29).$$

Solving sub-problem 1 is trivial. We assign $S_{ij}^{qt} = 1, \forall (q,t)$ to any (i,j) pair such that $(1 - \beta_{ij}) > 0$. Otherwise we assign $S_{ij}^{qt} = 0\ \forall (q,t)$.

Sub-Problem 2

$$Max \sum_{m,n} \alpha_{mn}\{- \sum_{i,j} P_{mn}^{ij}\} + \sum_{i,j} \beta_{ij}\{V_{ij} \times \mathcal{M}\} + \sum_{i} \gamma_j\{- \sum_{j} V_{ij}\} \tag{4.31}$$
$$subject\ \ to: (4.18), (4.20), (4.21), (4.22), (4.23), (4.24), (4.27), (4.28)$$

Sub-problem 2 can be decomposed into separate problems for all commodities $r \in R$ starting at node $i \in N$.

Problem (i,*);

$$Max - \sum_{m,n,j} \alpha_{mn} P_{mn}^{ij} + \sum_{j} (\mathcal{M}\beta_{ij} - \gamma_j) V_{ij} \tag{4.32}$$

subject to:

$$\sum_j V_{ij} \leq TT_i \tag{4.33}$$

$$\sum_m P_{mk}^{ij} = \sum_n P_{kn}^{ij}, \quad if\ k \neq i, j \quad \forall\, j, k \tag{4.34}$$

$$\sum_m P_{mi}^{ij} = 0, \quad \forall\, j \tag{4.35}$$

$$\sum_n P_{jn}^{ij} = 0, \quad \forall\, j \tag{4.36}$$

$$\sum_n P_{in}^{ij} = V_{ij}, \quad \forall\, j \tag{4.37}$$

$$\sum_m P_{mj}^{ij} = V_{ij}, \quad \forall\, j \tag{4.38}$$

$$V_{ij} \in \mathcal{Z}^+ \tag{4.39}$$

$$P_{mn}^{ij} \in \mathcal{Z}^+ \tag{4.40}$$

Remark 4.4.1 *The above decomposition scheme enables us to solve the Lagrangian sub-problems very easily. Unfortunately, the bound given by the Lagrangian solution and the LP relaxation solution are the same. This is due to the integrality property of the sub-problems. Therefore, the subproblems are required to be strengthen by adding some valid constraints.*

Note that, the maximum number of lightpaths generated from node i is TT_i and the maximum number of lightpaths received at node j is RR_j. Furthermore, the number of lightpaths needed to satisfy the demand from i to j is $\lceil \frac{d_{ij}}{\mathcal{M}} \rceil$, where d_{ij} is the total aggregate demand from node i to j ($d_{ij} = \sum_{q,t} o(q) \times S_{ij}^{qt}$). Also, the maximum number of lightpaths supported at each link is C. Considering all of these observations we can strengthen the upper bound of the Lagrangian relaxation by adding the following constraints to the Problem (i,*).

$$V_{ij} \leq min(RR_j, \lceil \frac{d_{ij}}{\mathcal{M}} \rceil), \quad \forall\, j \neq i \tag{4.41}$$

$$\sum_{j \neq i} P^{ij}_{mn} \leq C, \quad \forall\, mn \in A \tag{4.42}$$

Solving the $(i,*)^{th}$ problem:

The addition of extra constraints makes the Problem (i,*) difficult to solve. However, with this extra effort in solving we obtain an improved upper bound. To solve the Problem (i,*) optimally, we use CPLEX 8.1 MIP solver.

The sum of all solutions for (i,*) problems give the solution to *sub-problem 2*. Furthermore, the sum of $(\sum_{m,n} \alpha_{mn}\mathcal{C} + \sum_j \gamma_j RR_j)$ and the solutions from *sub-problem 1* and *sub-problem 2* will give the solution to the Lagrangian relaxation problem, *Groom_AggR*. We note that, this solution to problem *Groom_AggR* serves as an upper bound to the solution of problem *Groom_Agg* and therefore problem *Groom_MC*.

We employ a subgradient optimization algorithm to update the Lagrangian multipliers. Fisher [43], provides a detailed explanation of the subgradient procedure.

Table 4.3: Comparison of Heuristic, LP and Lagrangian solutions.

Problem	***Nodes***	***Tranceivers***	***Link Capacity***	***Heuristic***	***LP***	***Lagrangian***
1	20	12	2	438595	496020	496020
2	20	12	4	442646	523954	523954
3	20	12	6	457300	552003	552003
4	20	16	2	638568	991361	698300
5	20	16	4	642444	1035160	795167
6	20	16	6	655120	1037370	801202
7	20	20	2	697046	1256821	841252
8	20	20	4	702453	1269884	853495
9	20	20	6	764218	1143285	908733

Table 4.3 compares the heuristic solution, LP solution and Lagrangian solution for a network with twenty nodes. It can be seen that when the number of transceivers are

low the Lagrangian bound is very close to the LP relaxation solution. However, when the number of transceivers increases, the Lagrangian gives a better bound than the LP relaxation solution.

4.4.2 Wavelength Assignment

One cost component of an optical network is the number of unique wavelengths needed to accommodate demand. Our procedure provides a set of routes which are feasible with regard to the maximum link load ($\mathcal{C}$). To verify that the maximum number of distinct wavelengths needed for this set of routes is not inordinately large, we use a sequential network coloring approach given in Banerjee and Mukherjee [11]. Note that, this task of assigning wavelengths to a set of routes while minimizing the number of distinct wavelengths needed is itself NP-hard (it is referred to as the node coloring problem, see Garey and Johnson [45]). Hence, we use the aforementioned sequential network coloring approach as a post-processing step once a good set of routes are obtained. This post-processing step provides us with a measure $\mathcal{W}$, which is an upper bound on the maximum number of unique wavelengths needed for each solution provided by our heuristic.

4.5 Computational Study

For all our computational experiments, we generate networks such that a path exists between each pair of nodes. Following Zhu and Mukherjee [110], the traffic demand OC-q was allowed to be any one of $q \in Q = \{1, 3, 12\}$. Each demand was generated as a uniformly distributed random number between 0 and U_q ($U_q \in \{16, 8, 2\}$), as suggested by Zhu and Mukherjee [110]. A typical demand matrix for OC-12 requests is shown in Table 4.5.

It has been observed that WDM-based all-optical networks operating over a wide geographical area are sparse, with in/out-degree values ranging from 2-5 (Ajmon et al.[3], Banerjee and Mukherjee [11], Jia [57], Waxman [105]). A number of methods

Table 4.4: OC-12 demands

Node	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
0	0	0	0	2	0	1	2	1	0	0
1	1	0	2	0	2	0	1	2	1	2
2	1	1	0	1	0	1	2	1	1	1
3	1	2	2	0	1	1	1	1	0	0
4	2	2	0	1	0	1	0	2	2	0
5	1	1	2	1	0	0	0	1	1	2
6	2	1	0	2	0	0	0	0	0	0
7	1	0	1	0	2	0	2	0	0	1
8	1	0	0	2	0	2	0	0	0	1
9	1	1	1	0	0	0	2	2	0	0

have been proposed by researchers to emulate the topology of such wide-area, sparse networks (Baroni and Bayvel [13], Jia [57], Jia et al. [58], Waxman [105]). We utilize the procedure first developed in Waxman [105] to generate a realistic network $G(N, A)$:

1. **Generation of Nodes:** Three different values for $|N|$ were considered: 20,40 and 80.

2. **Generation of Arcs:** We assume that if two nodes u and v are adjacent, there exists a unidirectional fiber link from u to v and a separate unidirectional fiber link from v to u [59]. Therefore, throughout the arc generation process whenever we add an arc (u, v) to the network, we also add the reverse arc (v, u).

   (i) To ensure that the network is connected, we first find a minimum spanning tree (using Euclidean distances as arc weights) and add the arcs corresponding to the tree.

   (ii) Arcs between the nodes are added until the average in-degree (or equivalently, average out-degree since we always add bi-directional arcs) in the network reaches a specified maximum value. Three different values of the average in-degree are considered: 2, 4 and 6. We define networks of degree 2, 4 and 6 respectively as sparse, medium and dense.

3. **Transceivers:** We vary the number of transceivers at each node from $0.6 \times |N|$, to $1.0 \times |N|$ in steps of $0.2|N|$.

4. **Link capacity:** We consider link capacities ($\mathcal{C}$) of 2,4,and 6 lightpaths for each fiber.

Thus, there are a total of 81 problem settings (3 values each for the number of nodes (N), the average degree, the number of transceivers (T) and the link capacity ($\mathcal{C}$). For each setting, we generated 10 problem instances. Hence, a total of 810 problem instances were created on which our heuristic solution procedure was run. The heuristic was coded in C++ (using the CPLEX version 8.1 programming library) and all computations were carried out on a Pentium IV computer (2.4 GHz, 512 MB RAM) running Windows XP.

4.5.1 Discussion

Tables 4.5-4.7 depict the computational results for our heuristic solution procedure for the 810 problem instances generated as described in the previous section. Each row in these tables corresponds to a summary of 10 randomly generated problem instances. The following notation are required to read the results.

Notation:

$\lvert N\rvert$	:	Number of nodes in the network.
T	:	Number of transceivers at a node.
$\mathcal{C}$	:	Maximum link load.
$Network\ Type$	:	Average degree of the network ($degree \in \{Sparse, Medium, Dense\}$).
D	:	Percentage of total demand routed in terms of throughput.
U	:	Average transceiver utilization.
L	:	Number of lightpaths established.
Gap	:	$=\frac{Lagrangian\ Upper\ bound - Lower\ bound}{Lower\ bound} \times 100.$
W	:	Average number of distinct wavelengths required for a feasible solution.
$CPU\ time$	:	Running time in seconds for the algorithm, including the Lagrangian relaxation and wavelength assignment procedure.

It can be seen that our column generation heuristic is successful in obtaining high quality routes for large networks . We note that, the maximum gap between our feasible solution and the Lagrangian upper bound is small and typically ranges from 0% - 13%. The average gap is significantly lower and is on the order of 2% to 9% (see Table 4.8). With the increases in number of transceivers , the volume of traffic routed also increases and the gap between the heuristic solution and Lagrangian solution widens. One reason for this is when the volume of traffic increases, the LP relaxation and Lagrangian relaxation values tend to become very close. Further, our heuristic is successful in satisfying a significantly large fraction of the total demand (i.e., connection requests). Also dense networks route more traffic than sparse networks. This is to be expected, as dense networks imply more paths are available to

Table 4.5: Summary results for $|N|$=20. The average gap varies between 1.3% to 9.3%.

T	*C*	***Network Type***	***D***	***Gap***			***W***	***CPU time***			***U***	***L***
				max	min	avg		max	min	avg		
12	2	Sparse	73.8	5.0	0.0	1.3	3.0	12.0	9.3	11.4	100	93
12	2	Medium	76.6	7.3	3.0	5.5	3.0	13.0	9.0	10.5	100	95
12	2	Dense	76.8	4.7	1.5	3.3	3.7	12.9	9.3	11.5	100	97
12	4	Sparse	74.1	10.7	3.3	7.0	5.0	14.4	9.4	10.3	100	97
12	4	Medium	76.8	5.7	0.5	4.2	5.1	13.2	7.4	9.5	100	98
12	4	Dense	77.2	7.4	2.5	4.7	5.6	14.3	8.2	11.8	100	98
12	6	Sparse	74.3	7.4	2.4	5.2	6.9	13.0	9.3	10.4	100	97
12	6	Medium	76.9	8.2	3.6	6.3	7.2	12.3	8.4	12.0	100	97
12	6	Dense	77.3	8.4	1.8	4.8	8.2	16.3	9.5	11.3	100	98
16	2	Sparse	79.3	10.1	2.3	4.3	3.1	15.0	10.3	13.2	94.1	150
16	2	Medium	83.4	9.8	3.1	5.6	3.5	17.3	10.5	14.0	99.0	158
16	2	Dense	84.3	11.4	5.0	8.5	4.1	15.9	10.1	12.4	100	160
16	4	Sparse	79.8	9.5	1.3	3.0	5.0	16.9	11.0	11.7	94.7	150
16	4	Medium	83.8	9.5	1.7	5.1	5.0	19.8	10.9	16.3	99.5	158
16	4	Dense	84.4	11.8	3.3	8.0	6.0	18.3	10.4	15.3	100	160
16	6	Sparse	79.8	11.0	4.8	7.1	7.7	17.3	10.8	13.9	94.7	158
16	6	Medium	83.9	12.8	5.2	8.5	7.9	17.3	10.5	14.3	99.6	158
16	6	Dense	84.8	13.5	5.0	8.2	9.1	18.0	10.3	14.8	100	150
20	2	Sparse	87.1	9.7	2.7	4.5	3.2	19.2	11.2	16.9	82.7	164
20	2	Medium	93.9	11.1	3.8	5.7	3.3	20.4	11.0	17.3	89.2	178
20	2	Dense	94.5	10.2	3.6	5.8	4.3	19.4	10.9	16.0	89.7	178
20	4	Sparse	87.5	8.9	1.9	4.0	6.0	20.3	11.5	16.2	83.1	166
20	4	Medium	94.1	11.6	4.2	7.0	6.0	21.4	11.9	17.0	89.3	178
20	4	Dense	95.0	11.8	4.7	6.6	6.7	19.8	12.0	16.3	90.2	180
20	6	Sparse	87.7	13.3	5.4	9.3	7.6	22.5	12.2	17.3	83.3	166
20	6	Medium	94.1	9.7	3.8	5.8	8.0	20.5	11.3	16.8	89.3	178
20	6	Dense	95.1	10.0	2.6	6.6	9.0	21.3	12.0	15.9	90.3	180

Table 4.6: Summary results for $|N|$=40. The average gap varies between 2.2% to 9.2%.

T	*C*	*Network Type*	*D*	*Gap*			*W*	*CPU time*			*U*	*L*
				max	min	avg		max	min	avg		
24	2	Sparse	69.3	4.3	2.5	3.1	4.0	36.4	29.1	32.8	100	480
24	2	Medium	70.5	4.8	2.8	3.4	4.2	37.1	27.7	35.0	100	480
24	2	Dense	71.6	4.5	1.2	2.9	5.2	35.0	29.2	31.1	100	480
24	4	Sparse	69.4	6.9	4.4	5.1	6.0	36.3	28.6	30.6	100	480
24	4	Medium	72.3	4.9	3.6	3.9	6.1	37.6	29.0	31.3	100	480
24	4	Dense	72.8	6.2	1.8	4.3	6.0	36.3	30.2	32.5	100	480
24	6	Sparse	69.5	9.6	4.0	7.5	8.2	34.6	27.9	33.7	100	480
24	6	Medium	74.0	4.5	1.8	2.2	8.0	37.6	28.9	30.9	100	480
24	6	Dense	75.0	6.4	3.2	4.5	9.9	38.0	30.0	32.2	100	480
32	2	Sparse	87.9	7.2	5.6	6.4	4.2	47.0	34.1	40.2	98.8	627
32	2	Medium	89.3	7.5	6.0	6.8	5.0	46.6	36.0	38.3	100	640
32	2	Dense	91.9	8.0	6.4	7.3	5.2	45.0	34.0	41.5	100	640
32	4	Sparse	88.0	6.5	4.9	5.6	6.7	47.1	32.8	40.0	99	633
32	4	Medium	90.1	8.0	6.5	7.3	6.0	46.5	33.1	41.5	100	640
32	4	Dense	92.7	6.8	5.2	6.1	7.2	47.1	31.6	39.7	100	640
32	6	Sparse	88.1	5.6	3.1	4.5	8.2	46.4	33.6	40.0	99.1	633
32	6	Medium	91.0	6.2	4.7	5.5	8.8	48.3	34.2	41.0	100	640
32	6	Dense	93.3	4.8	3.2	4.1	9.5	44.1	35.0	40.5	100	640
40	2	Sparse	97.0	10.7	6.4	8.2	4.0	51.8	36.1	47.0	87.3	696
40	2	Medium	98.2	8.1	6.9	7.0	4.2	50.7	37.5	44.6	88.3	704
40	2	Dense	98.7	7.4	5.0	6.1	5.0	48.8	35.5	45.6	88.8	704
40	4	Sparse	97.1	11.4	7.4	9.2	5.3	49.6	38.1	47.1	87.3	696
40	4	Medium	98.2	6.9	4.7	5.8	6.5	50.4	36.9	44.6	88.3	704
40	4	Dense	98.7	7.1	4.7	5.8	8.0	49.9	37.2	45.4	88.8	704
40	6	Sparse	97.9	7.4	5.0	6.5	8.4	51.1	37.0	47.9	88.1	704
40	6	Medium	98.2	10.2	8.0	9.1	9.0	50.1	38.1	49.7	88.3	704
40	6	Dense	98.9	9.0	3.7	7.4	9.3	51.1	37.2	46.9	89.0	712

Table 4.7: Summary results for $|N|$=80. The average gap varies between 3.2% to 8.8%.

T	***C***	***Network Type***	***D***	***Gap***			***W***	***CPU time***			***U***	***L***
				max	min	avg		max	min	avg		
48	2	Sparse	72.9	5.2	2.5	3.4	3.6	88.1	69.0	77.8	100	1920
48	2	Medium	73.3	5.4	2.0	3.7	3.8	92.2	68.0	78.2	100	1920
48	2	Dense	73.9	3.9	1.3	3.2	4.2	85.1	65.2	77.7	100	1920
48	4	Sparse	73.1	8.3	3.7	7.2	5.5	88.3	66.3	75.4	100	1920
48	4	Medium	73.7	8.5	3.3	7.2	5.8	89.4	67.5	76.3	100	1920
48	4	Dense	74.2	6.0	2.8	5.2	6.2	85.2	67.5	79.2	100	1920
48	6	Sparse	75.0	5.2	2.8	5.8	7.3	90.3	65.2	78.9	100	1920
48	6	Medium	75.2	4.6	1.7	3.9	7.6	93.2	64.2	77.5	100	1920
48	6	Dense	76.5	10.2	4.7	7.3	8.6	94.9	65.2	77.5	100	1920
64	2	Sparse	88.8	9.1	1.9	5.6	3.5	103.5	78.3	87.6	100	2560
64	2	Medium	89.8	9.8	2.8	5.7	4.2	94.9	76.4	89.3	100	2560
64	2	Dense	89.9	10.8	4.8	7.8	4.6	95.4	79.3	87.6	100	2560
64	4	Sparse	85.5	12.0	5.6	7.8	6.2	100.3	76.2	86.4	100	2560
64	4	Medium	93.0	11.0	4.8	7.3	5.9	97.5	75.4	88.4	100	2560
64	4	Dense	93.4	10.1	4.0	6.3	6.8	95.7	74.3	90.4	100	2560
64	6	Sparse	91.1	10.6	5.0	6.7	7.7	99.4	76.3	88.5	100	2560
64	6	Medium	94.0	9.8	2.9	4.6	8.5	101.3	73.4	90.4	100	2560
64	6	Dense	95.2	10.7	3.9	8.8	9.8	99.3	78.3	86.3	100	2560
80	2	Sparse	96.7	9.3	1.4	4.7	3.9	117.4	88.4	107.3	94.7	3008
80	2	Medium	98.1	9.9	2.2	5.2	3.8	115.5	84.4	108.2	96.1	3072
80	2	Dense	99.3	10.6	5.1	8.8	5.6	116.3	91.9	108.6	97.3	3104
80	4	Sparse	97.9	10.5	3.8	8.1	6.4	113.5	96.4	105.8	95.9	3040
80	4	Medium	98.2	8.5	2.3	5.7	6.6	120.3	95.3	108.2	96.2	3072
80	4	Dense	99.1	10.0	3.1	6.5	7.3	116.1	89.3	109.2	97.1	3104
80	6	Sparse	97.5	10.8	3.7	8.0	7.7	114.8	95.3	108.2	95.5	3040
80	6	Medium	98.4	9.7	3.0	5.8	8.0	115.4	96.4	106.3	96.4	3072
80	6	Dense	99.2	13.2	4.9	8.6	9.6	108.5	99.2	108.2	97.2	3104

Table 4.8: Summary of Tables 4.5-4.7, range of average gaps.

Number of nodes $|N|$	***Average Gap varies between (%)***
20	1.2-9.3
40	2.2-9.2
80	3.2-8.8
Overall average	2.2-9.1

Table 4.9: Summary results for two practical networks. The average gap varies between 0.0% to 5.5%.

$\|N\|$	T	***C***	***D***	***Gap***			***W***	***CPU time***			***U***	***L***
				max	min	avg		max	min	avg		
25	15	2	72.1	5.8	0.6	1.8	5.8	13.9	11.2	13.3	100	187
	15	4	74.8	10.5	3.1	6.8	7.8	16.3	11.3	12.2	100	187
	15	6	75.2	5.3	0.7	3.8	9.7	14.9	11.2	12.3	100	187
	20	2	89.4	10.4	4.0	7.5	6.3	19.2	12.4	15.9	100	250
	20	4	90.8	9.7	1.2	5.9	7.8	21.7	12.8	18.2	100	250
	20	6	92.3	10.0	1.5	4.7	9.5	19.9	12.2	16.7	100	250
	25	2	95.7	7.9	1.3	3.6	6.0	21.1	13.1	18.8	89.0	278
	25	4	96.2	10.1	3.0	4.9	8.8	23.3	13.8	18.9	89.4	278
	25	6	96.9	9.0	3.1	5.1	9.4	22.3	13.2	18.7	90.1	281
79	47	2	68.8	6.5	2.8	4.8	6.8	87.2	68.1	76.9	100	1856
	47	4	70.5	6.6	2.2	5.4	9.0	87.4	65.4	74.5	100	1856
	47	6	73.9	9.4	3.1	6.3	9.8	92.3	63.2	76.5	100	1856
	63	2	80.9	12.0	2.7	7.4	7.8	102.5	77.4	86.6	94.3	2339
	63	4	84.8	10.4	4.1	6.3	9.0	99.4	75.3	85.5	98.8	2438
	63	6	87.2	11.4	3.4	6.7	10.6	98.5	75.4	87.6	100	2488
	79	2	90.1	10.4	4.4	7.3	8.8	116.5	87.5	106.4	83.7	2590
	79	4	93.2	11.2	3.8	7.4	9.5	112.5	95.5	104.9	86.6	2683
	79	6	95.5	11.6	4.5	7.7	10.2	114.5	95.5	105.4	88.8	2746

support the lightpaths.

The use of the sequential wavelength assignment technique indicates that the number of unique wavelengths ($\mathcal{W}$) needed for the routes are not inordinately large, typically being $\leq 1.5C$. As expected, the number of distinct wavelengths required increases with the maximum link-load. We also note that, dense networks require more wavelengths than sparse networks. An explanation for this is that dense networks generate more lightpaths, thus increasing the possibility of more lightpaths sharing the same link. In this case more wavelengths are required.

Computationally, the column generation heuristic is not very expensive. Computational times are approximately 80 to 100 seconds on the average for networks with 80 nodes, and much smaller for smaller networks, on a Pentium IV computer (2.4 GHz, 512 MB RAM) running Windows XP. This heuristic algorithm can therefore be utilized to efficiently benchmark the performance of greedy procedures such as those due to Zhu et al., [110] among others.

Finally, we applied our heuristic procedure to two typical network topologies provided by a large telecommunication equipment manufacturer, with node sizes of 25 and 79. For each of these networks, we generated traffic data as outlined in our computational study while varying the same parameters (number of nodes , maximum degree, the number of transceivers and the link capacity). For each parameter setting we generated 10 traffic matrices and solved them using our upper and lower bounding techniques. A summary of the results for these two networks are given in Table 4.9. We note that the average gap between our heuristic and Lagrangian upper bound ranges between 1.8% and 7.7%. Further, these results indicate that the values we obtained for CPU times and average $\mathcal{W}$ for this data set, are similar to results obtained for the randomly generated problem instances.

4.6 Concluding Remarks

We consider the problem of traffic grooming, routing, and wavelength assignment in optical networks with a mesh topology. Two different integer programming based formulations are presented. One formulation is used to approximate a traffic maximizing set of routes based on a column generation technique, and the other uses a Lagrangian relaxation technique to compute an upper bound. Our models limit the number of available transceivers at each node which accounts for a substantial fraction of network costs. These formulations do not restrict the number of wavelengths used, but do limit the link load. While we cannot guarantee that the number of wavelengths required will be minimal, our computational results show that the number of unique wavelengths needed for accommodating the set of routes (found by our heuristic) is not inordinately large. Our contribution in this chapter therefore lies, (i) in the development of the two models discussed above, and (ii) in the construction of efficient and high quality procedures for achieving lower and upper bounds for this computationally difficult problem.

The next chapter looks at the problem of flowshop scheduling subject to blocking. The solution approach we use in this problem is meta-heuristic search methods. Specifically, we look at genetic algorithms.

CHAPTER 5

BLOCKING FLOWSHOP SCHEDULING PROBLEM WITH A MATERIAL HANDLING CONSTRAINT

5.1 Synopsis

In the introduction, a formal statement of the blocking flowshop scheduling problem was provided. This chapter is organized as follows. The section 5.2 provides an overview of the literature on the closely related scheduling problems. The computational complexity of the blocking flowshop is investigated in section 5.3. In section 5.4, we briefly describe the existing scheduling methodology for a simple blocking flowshop problem. The structural insights are developed for the problem $F2|blocking, MH, m_1 = 1, m_2 = 2|C_{max}$ in section 5.5 and two heuristic algorithms are proposed. In section 5.6 we describe the elements of the genetic algorithm (GA) devised to tackle the blocking flowshop scheduling problem. An optimized set of parameters for the GA are also determined in section 5.6. Results of computational tests for the blocking problem are described with and without hybridizing in section 5.7. Finally, section 5.8 provides some concluding remarks.

5.2 Literature Review

Conway, Maxwell and Miller [31], Baker [8] [9], Rinnooy Kan [91], and Pinedo [84] provide comprehensive bibliographies for scheduling problems. First we review the flowshop with unlimited buffer capacity, also known as the permutation flowshop. Garey, Johnson and Sethi [46] prove that for three or more machines, the complexity of the recognition version of the permutation flowshop problem with makespan minimization objective as unary *NP*-complete. See Garey and Johnson [45] for related definitions. Baker [9] remarks that, the most successful use of branch and bound for makespan minimization in a permutation flowshop with unlimited buffers is by Potts [86], where instances with five machines and up to 100 jobs are often solved quickly. However, in as many as 25% of such problems, the algorithm was terminated after generating 100,000 branching nodes. Many heuristic procedures are available to solve this problem (Turner and Booth [100]). The best known fast heuristic method here is due to Nawaz, Enscore and Ham [78]. Osman and Potts [81] provide a more accurate but slower simulated annealing algorithm.

More closely related to the work in this chapter are studies of cyclic scheduling in flowshops with a variety of buffer designs. In cyclic scheduling, a set of jobs is produced repetitively in proportion to their demand until the demand for the jobs is met during the planning horizon (Matsuo [69]). Here the objective is the minimization of cycle time, which is the time to produce a set of jobs. Gilmore and Gomory [48] provide an efficient algorithm for the two machine flowshop without buffers. Kamoun and Sriskandarajah [61] show that a similar problem with three machines is intractable. Details of these and many related results appear in Hall and Sriskandarajah [54]. Karabati and Kouvelis [62] consider a simultaneous buffer design and cyclic scheduling problem, for which they describe and test an iterative heuristic.

The main reasons for designing a flowshop without buffers are cost and space efficiency. Intermediate buffers in the line tend to fill up due to poor shop floor practices while the absence of buffers prevents the accumulation of inventory which can be expensive [36]. Moreover, the factories of the future are being designed in a very space efficient way which discourages the provision of buffers for work-in-process [44]. A further reason is that absence of buffers limit the processing time of a job, which may improve its quality (Hall and Sriskandarajah, [54]).

Scheduling models with blocking gained considerable attention from the study of McCormick et al [70], who consider an m machine flowshop with fixed capacity buffers between machines. However, each unit of buffer capacity can be considered as a machine with zero processing times for all jobs. Therefore, their scheduling environment is exactly the same as ours, and they attempt to minimize cycle time. They describe a heuristic which constructs a partial sequence of the jobs, one job at a time, based on job's *penalty*, which is assessed by its flow contribution in a related network. They also describe a second heuristic with additional features. These heuristics provide either optimal or close to optimal schedules for five test problems with nine machines and eight jobs. Eight instances with ten machines and 80 jobs are solved heuristically, but comparisons with optimal cycle times or lower bounds are not provided.

Karabati and Kouvelis [63] develop an integer programming model for the blocking flowshop problem. Lower bounds on the cycle time are obtained by solving a blocking flowshop problem on two consecutive machines. A constructive heuristic sequences the jobs one at a time, using the smallest possible cycle time as a criterion for where to insert the jobs. Computational results suggest that this heuristic outperforms those of McCormick et al., [70]. A branch and bound algorithm that incorporates these approaches solves problem instances with 10 machines and 12 jobs to optimality within a few minutes of IBM 3081 CPU time. For problems with nine machines and 25 jobs, the heuristics deliver solutions with relative errors that average less than 3%, but can exceed 10% occasionally.

Abadi, Hall, Sriskandarajah [1] use the idea of deliberately slowing down the processing of operations (i.e., increasing their processing times) to establish a precise mathematical connection between the blocking flowshop and the no-wait flowshop. This enables them to adapt a very effective heuristic for the no-wait flowshop as a heuristic for the blocking flowshop. In the context of minimizing cycle time they use Lagrangean relaxation and decomposition methods to generate lower bounds on the cycle time. Their computational results show relative errors that average less than 2% for instances with 20 machines and 250 jobs.

Closely related to blocking flowshop problems are no-wait flowshop problems, where a job cannot pause between operations. In a no−wait flowshop, each job must be processed continuously from its start in the first processing stage, to its completion in the last processing stage, without any interruption on machines and without any waiting in between the processing stages. Therefore, the completion time of the last operation of a job must equal the starting time of its first operation plus the total completion time of all the operations. Applications of no-wait and blocking scheduling occur in the aluminum, steel, chemical and food processing industries, among others. A detailed discussion of applications of no-wait and blocking scheduling models, and the available algorithms, computational complexity results, and heuristics, is given by Hall and Sriskandarajah [54].

Scheduling problems in a no−wait flowshop with parallel machines arise in the chemical processes and petro-chemical production environment, where there are several simple flow lines. The nature of the process at each level in the flow lines is such that they are effectively identical and hence interchangeable(Salvador [93]). Another example of no−wait scheduling situations arise in hot metal rolling industries, where metals are processed at continuously high temperature.

Note that a no-wait flowshop with parallel machines represents a generalization of the traditional no-wait flowshop and the identical parallel machine shop. The former corresponds to the case $m_j = 1, \quad 2 \leq j \leq k$, while the latter corresponds to the case $k = 1$ and $m_1 \geq 2$, i.e., only one machine center incorporating more than one parallel machine. These special cases have been studied extensively in the literature. Efficient algorithms for minimizing makespan in a no-wait flowshop are not likely to exist, as even the special cases mentioned above (with the notable exception of $k = 2, m_1 = m_2 = 1$; Gilmore and Gomory, [48]) belong to the class of NP-complete problems (Garey and Johnson [45], Röck [92], Sriskandarajah [97], and Sriskandarajah and Ladet [98]). Salvador [93] inspired by an actual application in the synthetic fibers industry, developed a branch and bound algorithm to find minimum finish time schedules for a no-wait flowshop with parallel machines. The worst case analysis of some heuristic algorithms in the context of the no-wait flowshop with parallel machines has been carried out in Sriskandarajah [97] and Sriskandarajah and Sethi [99]. A number of studies consider buffers existing between the machine centers instead of no-wait restriction. Wittrock[104] has proposed a heuristic algorithm primarily to minimize the finish time and secondarily to minimize the in-process inventory. Some heuristic algorithms for two machine center flowshop (with parallel machines and unlimited buffer between processing stages) in the worst case performance context have been studied in Buten and Shen [19], Langston [65] and Sriskandarajah and Sethi [99]. Arthanri [5] studied a flowshop consisting of two machine centers Z_1 and Z_2 with $m_1 \geq 2$ and $m_2 = 1$. He developed a branch and bound based heuristic procedure for minimizing makespan.

5.3 Complexity of the Blocking Flowshop Problem

Hall and Sriskandarajah (1996) show that the problem $F2|\text{blocking}, m_1 \geq 1, m_2 \geq 1|C_{max}$ is equivalent to $F2|\text{no-wait}, m_1 \geq 1, m_2 \geq 1|C_{max}$. Sriskandarajah and Ladet (1986) show that $F2|\text{no-wait}, m_1 = 1, m_2 = 2|C_{max}$ is unary $NP-$complete. Thus $F2|\text{blocking}, m_1 = 1, m_2 = 2|C_{max}$ is also unary $NP-$complete. This result does not directly carry over to our blocking problem $F2|blocking, MH, m_1 = 1, m_2 = 2|C_{max}$, which is not equivalent to $F2|blocking, m_1 = 1, m_2 = 2|C_{max}$. Consequently, we investigate below the complexity of the two-stage blocking flowshop with material handling constraint, $F2|blocking, MH, m_1 = 1, m_2 = 2|C_{max}$.

Theorem 5.3.1 *The recognition version of the $F2|blocking, MH, m_1 = 1, m_2 = 2|C_{max}$ is unary $NP-$complete.*

Proof: The reduction is from the following unary NP-complete problem.
Numerical Matching with Target Sums [NMTS] (Garey and Johnson, [45]): Given three sets of positive integers $\bar{X} = \{x_1, \ldots, x_s\}, \bar{Y} = \{y_1, \ldots, y_s\}, \bar{Z} = \{z_1, \ldots, z_s\}$, can $\bar{X} \cup \bar{Y}$ be partitioned into s disjoint subsets $\Gamma_1, \ldots, \Gamma_s$ with $\Gamma_k = \{x_{i_k}, y_{j_k}\}$ such that $z_k = x_{i_k} + y_{j_k}$, for $k = 1, \ldots, s$?

Consider the following instance of the problem :
The job set $J = \{J_i|1 \leq i \leq n\}$ consists of three types $J = \{Jx_i, Jy_i, Jz_i|1 \leq i \leq s\}$

of parts where each type consists of s parts, or $n = 3s$. The processing times of three types of jobs are as follows:

The processing times for type Jx_i on Z_1 and Z_2 are $Z + x_i$ and L, respectively, for $i = 1, \ldots, s$.

The processing times for type for type Jy_i are $L + y_i$ and 1, respectively, for $i = 1, \ldots, s$.

The processing times for type Jz_i they are 1 and $L + Z + z_i$, respectively, for $i = 1, \ldots, s$.

It is also assumed that $Z = X + Y$, where $X = \sum x_i, Y = \sum y_i$, and $Z = \sum z_i$. Here we let $L = 2(s + 1)(Z + 2)$. The decision problem can be stated as follows: "Does there exist a sequence with cycle time $C_{max} \leq D = sL + (s + 1)(Z + 1)$?". Clearly the decision problem is class NP.

($\Rightarrow$) Suppose that a solution to NMTS exists. Thus we assume that $z_i = x_i + y_i$, $i = 1, \ldots, s$. Consider the following job schedule σ=[Jz_1, Jx_1, Jy_1, Jz_2, Jx_2, Jy_2, $\ldots$, Jz_s, Jx_s, Jy_s]. The Gantt chart for this sequence can be viewed as concurrently processing parts on three machines, as shown in Figure 5.1, where the processing times of parts on three machines which we denote by M_1, M_{21} and M_{22} are given. Hence the makespan $C_{max} = \sum_{i=1}^{s}(Z+x_i)+\sum_{i=1}^{s}(L+y_i)+s+1 = sL+(s+1)(Z+1)$.

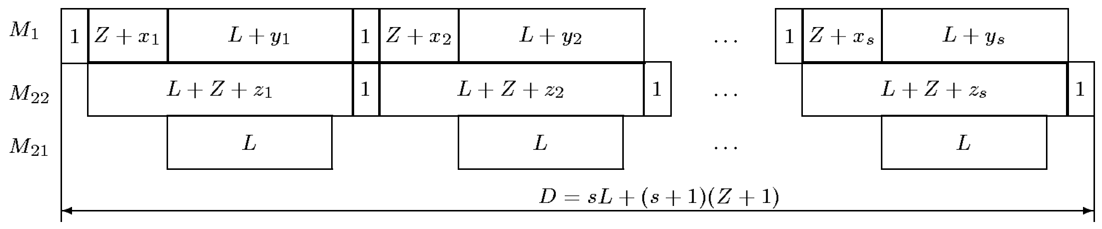

Figure 5.1: Sequence σ=[Jz_1, Jx_1, Jy_1, Jz_2, Jx_2, Jy_2, $\ldots$, Jz_s, Jx_s, Jy_s].

($\Leftarrow$) Suppose there exists a job schedule σ_0 such that $C_{max} \leq D$. We now show that, if there exists a schedule σ_0, then it will take the form (Jz_1, Jx_1, Jy_1, Jz_2, Jx_2, Jy_2, $\ldots$, Jz_s, Jx_s, Jy_s), and furthermore if $C_{max} \leq D$ then there must exist a solution to NMTS. To do so, we present and prove a series of facts about schedule σ_0.

Fact 1. Machine M_1 is busy processing parts throughout schedule σ_0 except one unit of idle time at the end of the schedule.

The total processing time of all the jobs on M_1 is $sL + sZ + X + Y + s$. Since $D = sL + (s+1)(Z+1)$ is the lower bound on the optimal schedule for this problem instance, any idle time on M_1 (except one unit at the end) will imply that $C_{max} > D$. Thus $C_{max} = D$.

Fact 2. In σ_0, total idle time on both machines M_{21} and M_{22} is equal to $(s+1)(Z+1)+1$.

The total processing time of all the jobs on the second stage machines is $2sL + sZ + Z + s$. As $C_{max} = D$ total idle time on both machines M_{21} and M_{22} is $2D - 2sL - sZ - Z - s = (s+1)(Z+1)+1$.

Fact 3. In σ_0, the number of jobs types Jx_i or Jz_k scheduled on M_{21} (or on M_{22}) is exactly s.

Suppose r jobs of types Jx_i or Jz_k are scheduled on M_{21}, where $r > s$. Then $C_{max} \geq rL \geq (s+1)L > D$. This contradicts with the fact that $C_{max} = D$ (Fact 1).

Fact 4. In σ_0, jobs are scheduled in the following form:

$(Jz_1, Jx_1, Jy_1, Jz_2, Jx_2, Jy_2, \ldots, Jz_s, Jx_s, Jy_s)$.

There are nine possible ways to schedule the first two jobs in σ_0: (Jz_i, Jx_j), (Jz_i, Jz_j), (Jz_i, Jy_j), (Jx_i, Jz_j), (Jx_i, Jx_j), (Jx_i, Jy_j), (Jy_i, Jx_j), (Jy_i, Jy_j), (Jy_i, Jz_j). We show that the first two jobs scheduled in σ_0 cannot be any of the five following forms: (Jz_i, Jy_j), (Jx_i, Jy_j), (Jy_i, Jx_j), (Jy_i, Jy_j), (Jy_i, Jz_j). Suppose the first two jobs scheduled in σ_0 takes any of the above five subsequences. Then the total idle time on both M_{21} and M_{22} is at least L contradicting with Fact 2.

We show that the first two jobs scheduled in σ_0 cannot be any of the two following forms: (Jz_i, Jz_j), (Jx_i, Jz_j). Suppose the first two jobs scheduled in σ_0 takes any of the above two subsequences. Then there will be an idle time on M_1 when scheduling any third job. This contradicts with Fact 1.

Now we are left with only two possibilities: (Jz_i, Jx_j), (Jx_i, Jx_j). In both cases, third job scheduled should be of type Jy_k. Otherwise an idle time on M_1 will be created contradicting with Fact 1.

Thus there are two possible ways to schedule the first three jobs in σ_0: (Jz_i, Jx_j, Jy_k) or (Jx_i, Jx_j, Jy_k).
By continuing arguments above for next three subsequent jobs scheduled in σ_0 (job $3i+1$, job $3i+2$ and job $3i+3$, $i = 1, 2, \ldots, s-1$), we can show that those three jobs take one of the following subsequences: (Jz_i, Jx_j, Jy_k) or (Jx_i, Jx_j, Jy_k).
Now we show that subsequence (Jx_i, Jx_j, Jy_k) cannot occur. Suppose subsequence, (Jx_i, Jx_j, Jy_k) occurs in σ_0, then there will be a subsequence (Jz_i, Jz_j, Jy_k). This subsequence (Jz_i, Jz_j, Jy_k) will cause an idle time on M_1 contradicting with Fact 1. Thus the subsequence (Jz_i, Jx_j, Jy_k) is only feasible. Hence σ_0 takes form of the following sequence: $(Jz_1, Jx_1, Jy_1, Jz_2, Jx_2, Jy_2, \ldots, Jz_s, Jx_s, Jy_s)$.

In σ_0, if $z_i < y_i + x_i$ then an idle time of $x_i + y_i - z_i$ occurs on M_1, contradicting Fact 1. If $z_i > y_i + x_i$, then there exists an index r such that $z_r < y_r + x_r$, hence an idle time of $x_r + y_r - z_r > 0$ occurs on M_1, contradicting Fact 1. Therefore, $x_r = y_r + z_r$ for $r = 1, \ldots, s$. Thus, there exists a solution to NMTS. $\square$

5.4 Two-Machine Blocking Flowshop Problem

In order to tackle the problem $F2|blocking, MH, m_1 = 1, m_2 = 2|C_{max}$ and gain structural insights, we first study the existing scheduling methodology for a simple two machine problem $F2|blocking|C_t$, where each stage has only one machine. The problem $F2|blocking|C_t$ is equivalent to the corresponding no-wait problem $F2|no-wait|C_t$. In the two machine flow-shop, minimizing the cycle time C_t is equivalent to minimizing total idle time on the first machine. Therefore, the no-wait problem and the blocking problem have the same objective function value for the same sequence of jobs. By expressing the elapsed time between the start of two successive jobs J_j and J_k on the second machine (M_2) as distance from city j to k, the no-wait problem, hence blocking problem, can be converted to a TSP as shown below:

Traveling Salesman Equivalence of Problems $F2|no-wait|C_{max}$ and $F2|blocking|C_{max}$:

Let $e_j = p_{j1}$, and $f_j = p_{j2}$, $j = 0, 1, \ldots, n, n+1$, where $e_0 = f_0 = e_{n+1} = f_{n+1} = 0$. Note that we need to introduce one artificial job J_0 with $e_0 = f_0 = 0$. The next cycle begins with job J_{n+1} where J_0 and J_{n+1} are identical.

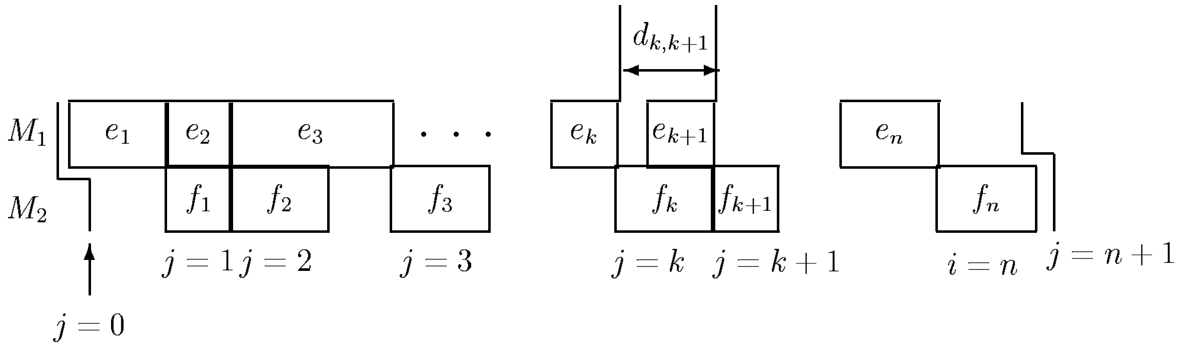

Figure 5.2: TSP Equivalence of $F2|no-wait|C_{max}$ and $F2|blocking|C_{max}$.

The distance, $d_{k,k+1}$ between two cities k, $k+1$ is defined as the time between start time of job k on M_2 and the start of job $k+1$ on M_2. Thus,

$$d_{k,k+1} = max\{e_{k+1}, f_k\}.$$

The makespan, C_{max} can be expressed as follows:

$$C_{max} = \textstyle\sum_{i=0}^{n} d_{k,k+1}.$$

Note that, the algorithm of Gilmore and Gomory [48] that solves the TSP with the special distances described above (and therefore $F2|no-wait|C_t$ problem) also solves the blocking problem $F2|blocking|C_t$. In both problems, a given set of jobs is produced repetitively, once each cycle. The algorithm described below solves $F2|no-wait|C_t$. The intuition behind the algorithm is that, ideally, the shortest processing time on first machine would be concurrent with that on the second machine, similarly for the second shortest processing times on the two machines, and so on. If this is not possible, a dual improvement step moves the current schedule towards feasibility at minimum cost. As the algorithm described below tries to match closely the task processing times on both machines, one can speculate that this may form a good heuristic to our original two-stage problem $F2|blocking, MH, m_1 = 1, m_2 = 2|C_{max}$ (Figure 1.1).

The optimal makespan of job set $J = \{J_i | 1 \leq j \leq n\}$ for $F2|no-wait|C_{max}$ is equal to the optimal cycle time of job set $J \cup \{J_0\}$ in $F2|no-wait|C_t$, where J_0 is an artificial job with zero processing time on both machines. Therefore, the Gilmore-Gomory algorithm also solves $F2|no-wait|C_{max}$.

Gilmore and Gomory Algorithm

Step 1. Sort $\{f_j\}$ in the nondecreasing order and renumber the jobs so that with the new numbering $f_j \leq f_{j+1}$, $j = 1, 2, ..., n-1$.

Step 2. Sort $\{e_j\}$ in the nondecreasing order $e_{\phi(j)} \leq e_{\phi(j+1)}$, $j = 1, 2, ..., n-1$. Find $\phi(p)$ for all p. The permutation ϕ is defined by $\phi(p) = q$, q being such that e_q is the p^{th} smallest member in $\{e_j\}$.

Step 3. Compute the distance of arc $(j, j+1)$ as

$c_{j,j+1} = max\{0, \{min(f_{j+1}, e_{\phi(j+1)}) - max(f_j, e_{\phi(j)})\}\}$ for $j = 1, 2, ..., n-1$.

Step 4. Construct an undirected graph with n nodes and undirected arcs $(j, \phi(j))$, $j = 1, 2, ..., n$.

Step 5. If the current graph has only one component, go to step 7. Otherwise select the smallest value $c_{j,j+1}$ such that j is in one component and $j+1$ in another. In the case of a tie for the smallest, choose any.

Step 6. Adjoin the undirected arc $(j, j+1)$ to the graph using the j value selected in step 5. Return to step 5.

Step 7. Divide the arcs added in step 6 into two groups. Arcs $(j, j+1)$ for which $f_j \leq e_{\phi(j)}$ go in group 1 while arcs with $f_j > e_{\phi(j)}$ go in group 2.

Step 8. Find the largest index j_1 such that arc (j_1, j_1+1) is in group 1. Find the second largest j_2, etc., up to j_r, assuming there are r elements in group 1.

Step 9. Find the smallest index t_1 such that arc (t_1, t_1+1) is in group 2. Find the second smallest t_2, etc., up to t_k, assuming there are k elements in group 2.

Step 10. The minimal cycle time is obtained by following the j^{th} job by the job $\psi^*(j)$, where $\boxed{\psi^*(j) = \phi(v)}$, and $v = \alpha_{j_1,j_1+1}\alpha_{j_2,j_2+1}...\alpha_{j_r,j_r+1}\alpha_{t_1,t_1+1}\alpha_{t_2,t_2+1}...\alpha_{t_k,t_k+1(j)}$. In the above expression the permutation $\alpha_{p,q}$ is defined to be $\alpha_{p,q}(p) = q, \alpha_{p,q}(q) = p$ and $\alpha_{p,q}(j) = j$, if $j \neq p, q$.

Example 1: $F2|no - wait|C_t$

- Take the eight-job problem given in Table 5.4. j is the job number and e_j and f_j are processing times on machines M_1 and M_2, respectively.

Table 5.1: Jobs and their processing times

j	e_j	f_j
1	1	10
2	10	9
3	16	25
4	1	5
5	8	5
6	17	24
7	4	10
8	22	10

- After sorting f_j's in non-decreasing order and re-numbering jobs we obtain Table 5.4 (Step 1).

Table 5.2: Jobs after sorting by f_j

j	rename jobs	f_j	e_j
1	J_1	5	1
2	J_2	5	8
3	J_3	9	10
4	J_4	10	1
5	J_5	10	4
6	J_6	10	22
7	J_7	24	17
8	J_8	25	16

- We get the column $e_{\phi(j)}$ in Table 5.4 by sorting in non-decreasing order, the entries for each f_j in column e_j of Table 5.4. Column $\phi(j)$ gives the corresponding job number of e_j's in Table 5.4. Next we calculate the max and min of columns f_j and $e_{\phi(j)}$. Using these and the equation given in Step 3 of the Gilmore and Gomory algorithm we calculate distance or cost column $c_{j,j+1}$ (Table 3).

Table 5.3: Distance of arc $c_{j,j+1}$.

j	f_j	$e_{\phi(j)}$	$\phi(j)$	$max\{f_j, e_{\phi(j)}\}$	$min\{f_j, e_{\phi(j)}\}$	$c_{j,j+1}$
1	5	1	1	5	1	0
2	5	1	4	5	1	0
3	9	4	5	9	4	0
4	10	8	2	10	8	0
5	10	10	3	10	10	0
6	10	16	8	16	10	1
7	24	17	7	24	17	0
8	25	22	6	25	22	-

- Next we construct a undirected graph (Figure 5.3) with nodes corresponding to job numbers $j = 1$ to 8. By looking at the columns j and $\phi(j)$ of Table 5.4 we draw the undirected edges j-$\phi(j)$ on the graph (Step 4). Thus we obtain edges 2-4, 3-5 and 6-8 in Figure 5.3.
  The graph in Figure 5.3 has five components. Node 1 is in component one, nodes 2 and 4 are in component two, nodes 3 and 5 are in component three, nodes 6 and 8 are in component four and node 7 is in component five.

- Except edge 6-7, all other edges have cost zero. So we adjoin undirected arcs 1-2, 2-3, 5-6, and 7-8 to the graph to obtain a single component graph (Figure 5.4), completing Steps 5 and 6.
  Figure 5.4 shows the final result.

- An inspection of Table 5.4 shows that, condition $f_j \leq e_{\phi(j)}$ is satisfied only by the arc (5 6) (Step 7). Hence, from Step 7 we obtain two groups with group one containing the arc (5 6) and group two containing arcs (1 2), (2 3) and (7 8).

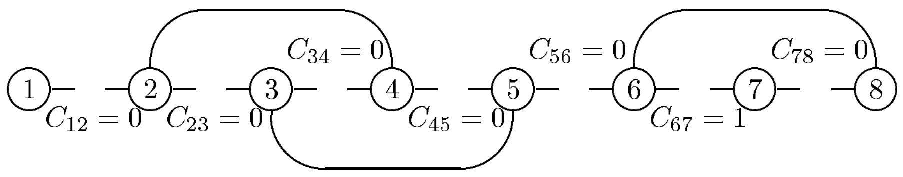

Figure 5.3: Undirected Graph with five components

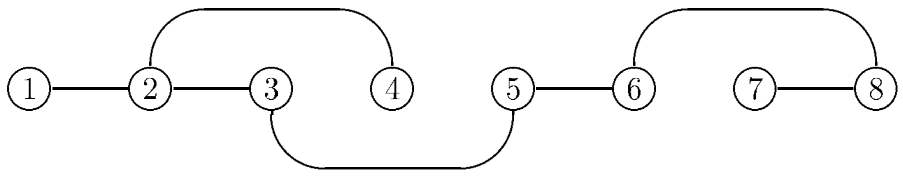

Figure 5.4: Single component Graph

$f_j \leq e_{\phi(j)}$	$f_j > e_{\phi(j)}$
Group 1	**Group 2**
(5,6)	(1,2) (2,3) (7,8)

- Next we execute Steps 8 and 9.

  The largest index j_1 such that arc $(j_1, j_1 + 1)$ is still in Group 1 is 5. So $j_1 = 5$. The smallest index s.t. arc $(t_1, t_1 + 1)$ is in Group 2 is 1. Hence $t_1 = 1$. Similarly we find that $t_2 = 2$ and $t_3 = 7$.

- For any job j, cycle time is minimized when the job following it $\psi^*(j)$ is such that $\psi^*(j) = \phi(v)$, where $v = \alpha_{5,6}\alpha_{1,2}\alpha_{2,3}\alpha_{7,8}(j)$ (see Step 10). Hence, to obtain the job following job j, we apply the permutation $\psi^*(j) = \phi\alpha_{5,6}\alpha_{1,2}\alpha_{2,3}\alpha_{7,8}(j)$.

For example, for $j = 1$,

$$\begin{aligned}\psi^*(1) &= \phi\alpha_{5,6}\alpha_{1,2}\alpha_{2,3}\alpha_{7,8}(1) \\ &= \phi\alpha_{5,6}\alpha_{1,2}\alpha_{2,3}(1) \\ &= \phi\alpha_{5,6}\alpha_{1,2}(1) \\ &= \phi\alpha_{5,6}(2) \\ &= \phi(2) = 4\end{aligned}$$

From Table 5.4 we see that $\phi(2) = 4$. Hence job 4 will follow job 1. Similarly for all other jobs the immediately following job may be found. These are shown in Table 5.4.

Table 5.4: Job Sequence.

j	$\psi^*(j) = \phi\alpha_{5,6}\alpha_{1,2}\alpha_{2,3}\alpha_{7,8}(j)$
1	4
2	5
3	1
4	2
5	8
6	3
7	6
8	7

- From Table 5.4 the optimal job sequence can be obtained by reading the $(j, \psi^*(j))$ pairs. The optimal job sequence is $\boxed{14258763}$, that is, sequence J_1, J_4, J_2, J_5, J_8, J_7, J_6, J_3.

- The Gantt chart in Figure 5.5 gives the optimal makespan for this sequence. Note that cycle time $C_t = 104$. If we were to solve $F2|no-wait|C_{max}$ problem, we must introduce an artificial job, J_0 with $e_0 = 0$ and $f_0 = 0$, and re-run the above algorithm. In this case we will obtain the same job sequence with $C_{max} = 105$.

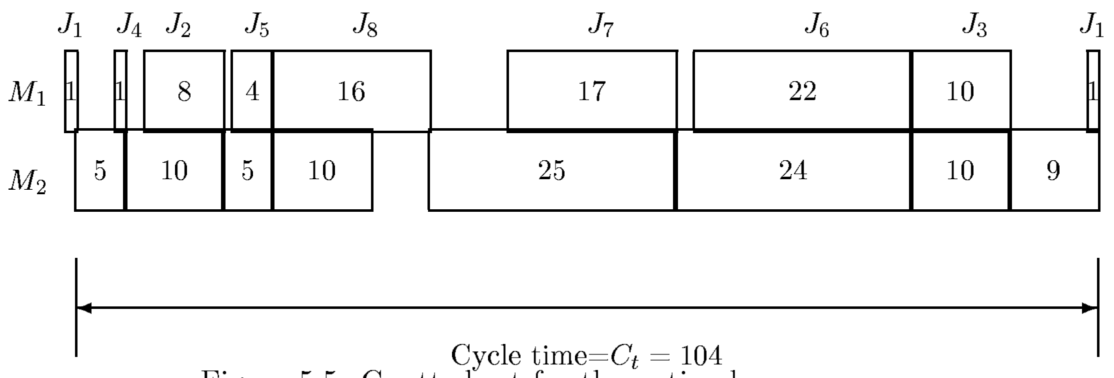

Figure 5.5: Gantt chart for the optimal sequence

5.5 Two-Stage Blocking Flowshop Problem

In this section, we study the structure of the two-stage blocking flowshop problem, $F2 \,|\, blocking,\ MH,\ m_1 = 1, m_2 = 2 \,|\, C_{\max}$. In constructing a schedule for a given job sequence, it is intuitive that a job should be assigned to M_{22} whenever both machines in the second stage are free. Processing that job on M_{21} will block other jobs from entering machine M_{22}. Now we state our schedule rule below.

Scheduling Rule 1: Given a job sequence, the job schedule is constructed by assigning the jobs to machine M_{22} whenever both machines in the second stage are free.

It is difficult to establish a precise mathematical formula for the makespan under this scheduling rule 1. Therefore, we use the following scheduling rule 2 that is useful in approximately capturing the makespan formula for the scheduling rule 1.

Scheduling Rule 2: We assume without loss of generality that the job sequence is $\{J_1, J_2, \ldots, J_n\}$. Odd numbered jobs are scheduled on machine M_{22} and even numbered jobs are scheduled on M_{21}.

We use the scheduling rule 2 to establish a mathematical formula for the makespan. Under rule 2 for a given job sequence $\{J_1, J_2, \ldots, J_n\}$, the idle time on M_1 may be written as follows (see Figure 5.6 for $n = 6$):

Let $e_j = p_{j1}$, and $f_j = p_{j2}$, for all j.

$d_{12} = \max\{0, f_2 - e_3, f_1 - e_2 - e_3\}$,

$d_{23} = \max\{0, f_4 - e_5, f_3 - e_4 - e_5\}$,

$d_{34} = \max\{0, f_6 - e_7, f_5 - e_6 - e_7\}$, where $e_7 = 0$.

Thus $d_{i,i+1} = \max\{0, f_{2i} - e_{2i+1}, f_{2i-1} - e_{2i} - e_{2i+1}\}$.

For n even, the makespan C_{max} for the job sequence $\{J_1, J_2, ...J_n\}$ can be written as follows:

$$C_{max} = \textstyle\sum_{i=1}^{n} e_i + \sum_{i=1}^{r} d_{i,i+1}.$$

where we let $(n+1)$th job be an artificial job with $e_{n+1} = f_{n+1} = 0$ and $r = n/2$.

The makespan can be expressed as follows (assume n is even):

$$C_{max} = e_1 + e_2 + \max\{e_3, f_2, f_1 - e_2\} + e_4 + \max\{e_5, f_4, f_3 - e_4\} + .. + e_{2r} + \max\{e_{2r+1}, f_{2r}, f_{2r-1} - e_{2r}\}.$$

Similarly, the makespan for n odd can easily be derived as follows:

$$C_{max} = \textstyle\sum_{i=1}^{n} e_i + \sum_{i=1}^{r} d_{i,i+1}.$$

where we let $(n+1)$th and $(n+2)$th jobs be artificial jobs with $e_{n+1} = f_{n+1} = e_{n+2} = f_{n+2} = 0$ and $r = (n+1)/2$.

Note that it is easy to see that the makespan given by rule 2 is an upper bound for the makespan given by rule 1. More precisely, for a given job sequence the following relation holds.

$$C_{max}(rule\ 1) \leq C_{max}(rule\ 2) = \sum_{i=1}^{n} e_i + \sum_{i=1}^{r} d_{i,i+1}.$$

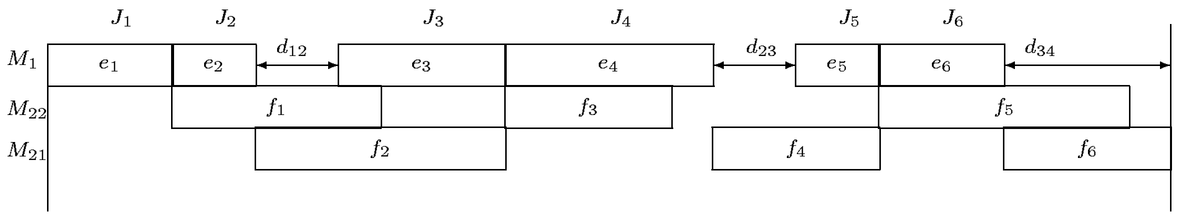

Figure 5.6: A Schedule for the job sequence $\{J_1, J_2, ...J_n\}$ using rule 2, $n = 6$.

5.5.1 Structural Insights and Heuristics

Note that in all of the heuristics proposed below, we use scheduling rule 1 in constructing a job schedule. That is, in a given job sequence, the job entering stage 2 should be assigned to M_{22} whenever both machines in the second stage are free.

We consider two heuristics for the $F2|blocking, MH, m_1 = 1, m_2 = 2|C_{max}$ problem. These are respectively, the Gilmore and Gomory algorithm [48], and DECf procedure explained below. Furthermore, when implementing the sequence obtained from either of these two heuristics we follow the scheduling rule 1 described above.

Gilmore and Gomory Algorithm

We apply the Gilmore and Gomory algorithm and get a job sequence. This is optimal for the two machine case. However it is not optimal for our 2-stage three-machine situation. But when the processing times on both stages are nearly equal, the *max* term in the makespan equation (5.1) becomes nearly equivalent to $max\{e_{2r+1}, f_{2r}\}$. This form is solvable by Gilmore and Gomory algorithm which is a good heuristic for problem instances where processing times on both machines are nearly equal (or generated from the same Uniform distribution).

Algorithm DECf

Our second heuristic is called DECf. This is a simple heuristic obtained as follows: Process the jobs in the *non-increasing* order of f_i (their processing times in the second stage). Note that, when the processing times f_i's are much larger than e_i's, the *max* term in the makespan equation (5.1) becomes closer to $max\{f_{2r}, f_{2r-1}\}$. This form is solvable by algorithm DECf, a good heuristic for problems where processing times f_i's are much larger than e_i's.

On the basis of computational experiments conducted, we find that this heuristic performs better when second stage is the bottleneck (i.e. has the longer processing times) . In this case it is important to minimize the total processing time in the second stage. By following DECf sequence we try to process equal-sized second processing time jobs by successively parallelizing operations in M_{21} and M_{22}. Matching jobs this way reduces the total completion time on stage two.

5.6 Genetic Algorithms (GA)

A large class of combinatorial problems remains analytically intractable. Scheduling jobs in a blocking Flowshop to minimize the makespan ($Fm|blocking|C_{max}$) is one such problem (Hall and Sriskandarajah [54]). Scheduling in a no-wait Flowshop ($Fm|no-wait|C_{max}$) is another. Many other optimization problems contain very little easily-apparent structural information that may be exploited systematically. Such problems are now being tackled by heuristic or meta-heuristic methods. In the present work, we have also evaluated the use of GAs in scheduling a blocking flowshop with material handling constraint. GAs are directed global search meta-heuristics that can find solutions by simultaneously exploring multiple regions of the solution space and by exploiting the promising areas. However, unlike many standard optimization techniques, GAs do not make strong assumptions about the forms of the objective function. This section outlines the GA and special GA operators used in the present study. There is yet another reason for using meta-heuristic global search methods. This is when the presence of local optima is suspected. First, note that in many real-world problems we are seeking the global optimum rather than the local optima. Only a subclass of these problems, called convex programming problems, are guaranteed to be free of distinct local optima. A nonlinear optimization problem in general will not be a convex programming problem and hence, it may have local optima distinct from the global optimum. Further and importantly, there is no simple way to test a nonlinear response function for convexity.

GAs belong to a class of heuristic methods that uses randomization as well as directed smart search methods to seek the global optima (Holland [55]). GA procedures are inspired by evolutionary processes through which life is believed to have evolved in its present forms (Goldberg [50]). The other such popular heuristic search methods are simulated annealing (Connoly [30]) and tabu search (Glover and Laguna [49]), all used to assist in the solution of NP-hard problems. However, GAs are regarded to be more general and abstract (context independent) than both simulated annealing and tabu search (Pinedo [84], p. 154). Applications of GAs in scheduling include Uckun et al [101], Murata and Ishibuchi [76], Chen, Vempati and Aljaber [25] and Portmann [85] among numerous others. GAs are being steadily tested in solving many large and difficult Operations Research problems, such as the TSP to speed up its solution (Michaelewicz [72]). In scheduling, GAs often view sequences (of jobs) as *chromosomes* (the candidate schedules or solutions), which in turn are members of a population. Each individual (a schedule) is characterized (merited) by its fitness (e.g., by its makespan value). Thus, the fitness of an individual is measured by the associated value of the objective function . The GA procedure works iteratively (thus emulating an algorithm) with the members of the population; each new iteration is a *generation* in the GA terminology. In the GA environment, a generation of chromosomes consists of surviving chromosomes of the previous generation and some new solutions or progeny created from the previous generation. While the GA is iterating, the population size is usually kept constant from one generation to the next.

The progeny are produced through reproduction, mating by crossover, and mutation of the chromosomes of the previous generation (the parents). Individual solutions consist of the juxtapositioning of *genes* as in biology. It is the presence of certain genes that imparts each solution its individual and sometimes unique characteristic. In the context of scheduling, a chromosome may represent information regarding the job sequence on a machine, an example being [A C B D F]. The different *holding* positions in this sequence are known as *alleles*. Thus, the second allele in this sequence has the value C. A mutation in a parent chromosome may be equivalent to an adjacent pairwise interchange in the corresponding sequence. A crossover combines some features of two parent chromosomes to create progeny inheriting some characteristics from each parent. As the GA iterates, in each generation the *fittest* chromosomes reproduce and the least fit die. The birth, death, and evolutionary reproduction processes that determine the characteristics of the next generation of solutions can lead to a complicated stochastic process. This process is a function of the fitness levels of the chromosomes of the current generation and the rates of mutation, crossover and reproduction, the controlling elements or operators of the genetic evolutionary process.

5.6.1 Selection, crossover, and mutation of GA solutions

Selection is a key operation in GA search, which in scheduling may be directed to optimize an objective function, such as the minimization of makespan. The purpose of selection is to propagate high fitness characteristics in the population and to eliminate from the population the weaker chromosomes, thus keeping the search *directed* toward the optimization of the objective function. One effective way of doing selection is to select the elite (higher fitness) solutions in the population and then reproduce them in proportion to their relative fitness (Goldberg, [50]). A parameter (called *elite fraction* or ϵ) often controls the (upper) fraction of the population that is treated as elite. Many other variations of this simple selection scheme may be devised. The *crossover* operator lets suitably chosen pairs of chromosomes to *mate* with each other to produce progeny. In scheduling, crossover is implemented slightly differently from the manner in which it is done in simple numerical optimization. One has to worry about preventing infeasible solutions (schedules) from being created, using special repair methods, if necessary. Murata and Ishibuchi [76] list some examples of *job sequence* type chromosomes and their processing. Erben [38] used GAs similarly to guide university time tabling decisions. Construction of the appropriately coded solution representations and the corresponding GA operators is one vital aspect of successfully applying GAs in heuristic search-based optimization. Perhaps an equally important challenge is the optimization of the computational effort, balancing exploration (of the solution space) and exploitation (of the features of good solutions or sequences produced along the way). This balance is affected greatly by the choices of the different GA parameters, including elite fraction ϵ of current parents selected to participate in mating, population size (P_s), the probability of crossing two selected parents (P_c), and the probability of mutation (P_μ). The problem itself is one optimization and guidelines here are still rather few (Davis [33], Chen, Vempati and Aljaber [25]). Parameterization has also been noted to be problem-specific. We used the design-of-experiments (DOE) approach to identify the optimal settings for ϵ, P_s, P_c and P_μ before we used GA in the *production* mode to discover the optimum job sequence (Bagchi and Deb [6]). In this study we kept ϵ=1 and maintained the same optimum population size (P_s) throughout. In an extensive simulation study of flowshop scheduling, Murata and Ishibuchi [76] reported the superior performance of the two-point crossover and the shift change mutation procedure in genetic search of optimum sequence in flowshops. However, as with the parameterization of GAs, such observations are also problem-domain specific rather than valid in all situations. For instance, in the present study best results in sequencing the products were obtained when one-point crossover and arbitrary two-job change mutation were used.

5.6.2 Special GA operators devised for the present problem

Here we explore a way to choose the different GA parameters, namely P_s, P_c, P_μ, etc., and the different crossover and mutation schemes that introduce a controlled amount of randomness while the GA is executing. However, it is well known now that the GA's efficiency depends to a high degree upon the selection of these *control parameters* (Davis [33]). A further complication exists because it is also widely reported that *parameter settings* may be problem-specific, and that the effects of these parameters may interact.

Initialization of the GA

The GA starts with a population of randomly generated initial solutions. As it is conventional, we started this study by randomizing this initial condition. Subsequently we hybridized the GA with suitably selected initial solutions.

Crossover and mutation for sequencing the products

A key consideration in designing the crossover operator for the sequencing chromosome is avoidance of infeasible sequences. Recall that optimal sequencing in a flowshop is a permutation search. Earlier published work report good results with partially mapped crossover with necessary repairs to restore feasibility to be performing well in permutation search (Uckun et al [101]). In the present study, the one-point crossover when combined with appropriate parameters P_s, P_c and P_μ performed the best.

In the crossover method we used, two offsprings are produced by mating of two parents. Upto the crossover point one offspring will have the same allele sequence on its chromosome as one parent. (The other offspring will have the same as the other parent.) To fill the remaining alleles we search the other parent's chromosome from the beginning (the left-most allele) to end (the rightmost allele) sequentially. When an allele which is not already in the offspring's chromosome is found, we copy this to the offspring's chromosome. We proceed this way until all remaining allele positions are filled (Figure 5.7). In this method no repair is needed on offspring's chromosome to ensure feasibility of the sequence. Mutation was done by occasionally switching two randomly choosen alleles in an offspring's chromosome (Figure 5.8).

Description of the GA

The GA algorithm presented below generates P_s random job sequences for generation 0 and evolves the population of solutions for a given number of generations (Gen_{max}). The generation index is specified by variable Gen. Each generation has the same population size P_s. The algorithm keeps track of the best sequence σ^* found in each generation and the corresponding fitness value f^* which is equal to the reciprocal of the makespan C_{max}. Each generation's best sequence is fed to the initial population of the following generation. This way the best solution is preserved from one generation to the next. The best sequence after P_s generations (σ^{**}) is the overall best sequence.

GA algorithm

Step 1. Initialize P_s, P_c, P_μ and Gen_{max}. Set $Gen = 0$ and $f^{**} = 0$.

Step 2. Generate P_s random job sequences σ_i, $i = 1, ..., P_s$.

Step 3. Evaluate each sequence σ_i and find fitness value f_i $(= 1/C_{max})$.

Step 4. Sort all the sequences in non-increasing order of fitness values. Find the current best sequence of generation Gen, σ^* and the corresponding fitness value f^*.

If $f^* > f^{**}$, then set $\sigma^{**} = \sigma^*$ and $f^{**} = f^*$.

Step 5. If $Gen = Gen_{max}$ go to step 11.

Step 6. Select top two parents from the ordered population of generation Gen and crossover with probability P_c to produce two children. Next select the next two parents and do the same. Continue this way till all population members are selected. The children obtained from crossover and the parents from generation Gen will be members of generation $Gen + 1$.

Step 7. Mutate the whole population (Children+Parents) based on their mutation probabilities.

Step 8. Sort all the sequences in non-increasing order of fitness values. Preserve the top P_s members and delete the rest.

Step 9. Replace the sequence having the worst fitness value in P_s members obtained in Step 8 with σ^*.

Step 10. Set $Gen = Gen + 1$ and go to Step 3.

Step 11. σ^{**} is the best sequence obtained with makespan $C_{max} = 1/f^{**}$.

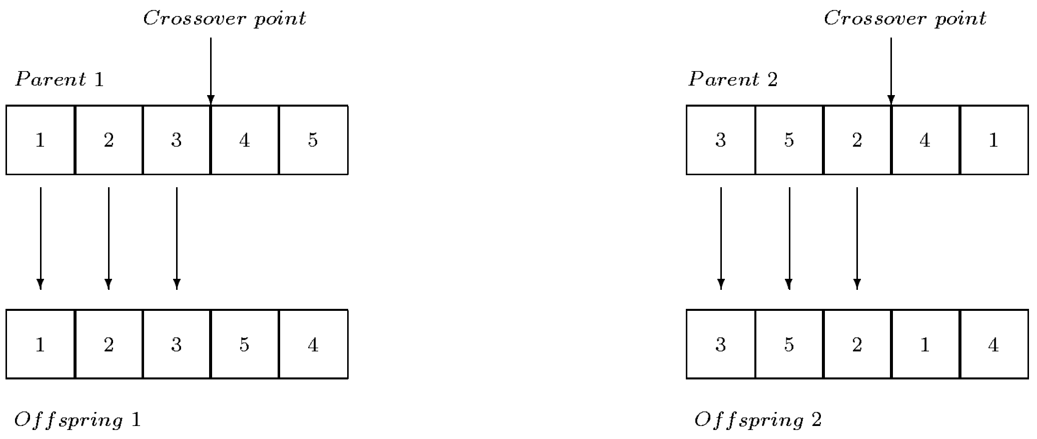

Figure 5.7: One-Point Crossover Implemented

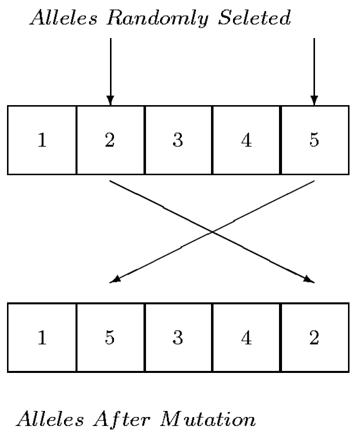

Figure 5.8: Single Random Allele Mutation

GA parameterization by DOE

Since the GA's convergence is noted to be greatly affected by the choice of the GA parameters, this aspect was especially addressed in this study. A good combination of parameter values has two properties. First, it produces a lower makespan average of the best schedules reached in the replicated runs. Second, it produces these best values with high consistency, or a lower variance. Thus, a well-parameterized GA converges to the global optimum rapidly, and consistently, regardless of the algorithm's starting condition. The last characteristic makes a good GA robust in its performance. A statistical experimental framework (Bagchi and Deb [6]) was used in the present study to discover the most appropriate values of the GA parameters P_c (the probability of crossover), P_μ (the probability of mutation) and P_s (population size). The experimental layout used is a three-level full factorial design (Montgomery [74]) involving parameter values judged to be the possibly correct values. Each *experimental run* involved parameterizing the GA according to the rows in the experimental table and then running the GA from certain fixed initial condition for a given number of generations (here 100), in a pilot run fashion. The makespan value of the best schedule produced in each such run indicates the parameters' combined performance. Further, to remove the bias of the starting sequence, several randomly produced initial starting (job) sequences were tested in each run, providing the necessary replications; these starting sequences were kept unaltered in each factorial run. Figure 5.9 indicates the marked difference (for a 100-job problem) in the convergence behavior of two typical sequencing GA runs using parameter values as shown. It is evident that inappropriate parameterization produces poor convergence or no convergence at all. Tables 5.6.2 and 5.6.2 display typical results obtained from the experimental three-factor full factorial GA runs for 100 generations with parameters as shown.

Interaction of parameter effects indicated that the parameters here could not be arbitrarily and individually *optimized*. Subsequently, the values $P_s = 1000$, $P_c = 0.95$ and $P_\mu = 0.05$ were selected as the most appropriate ones. These parameter values were implemented in the *production* GA runs for processing of the sequencing chromosome. Figure 5.10 shows the characteristic convergence of the shortest makespan produced by a well parameterized GA run.

Table 5.5: $P_s = 1000$

P_c	P_μ	SEED No.									
		1	2	3	4	5	6	7	8	9	10
0.9	0.01	4026	4024	4013	4033	4037	4078	4037	4044	3994	4082
0.9	0.05	4028	3985	4019	3975	3997	4015	3976	4007	4010	3996
0.5	0.01	4127	4137	4127	4128	4137	4119	4132	4128	4132	4118
0.5	0.05	4081	4111	4080	4091	4082	4120	4057	4108	4101	4140

5.7 Computational studies and comparisons of results

Two computations studies were done to evaluate the methods proposed in this paper. In the first one we randomly generated job processing times for the two stages of the $F2|blocking, MH, m_1 = 1, m_2 = 2|C_{max}$ flowshop as follows. For stage one and stage

Table 5.6: $P_s = 500$

P_c	P_μ	SEED No.									
		1	2	3	4	5	6	7	8	9	10
0.9	0.01	4085	4104	4057	4106	4135	4063	4129	4063	4064	4055
0.9	0.05	4045	4083	4039	4024	4020	4044	4022	4055	4068	4064
0.5	0.01	4283	4198	4159	4192	4213	4114	4152	4167	4180	4120
0.5	0.05	4161	4091	4143	4093	4126	4103	4115	4133	4158	4117

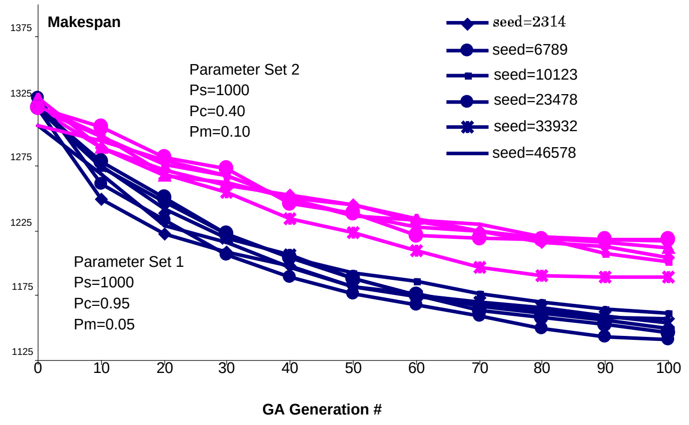

Figure 5.9: GA convergence for two different parameter settings with identical random starts.

two job processing times were randomly generated from $U \sim (1-10)$ and $U \sim (1-35)$ distributions, respectively. For the second study we obtained processing times from the same distribution–$U \sim (1-35)$–for both stages.

Therefore, for the first study, the total processing time of stage 2 is greater than that of stage 1. For the second study, this is roughly the same. Hence in the second study, the benefit if any of having an extra Machine in stage 2 would likely be marginal.

We considered three sets of problems in each case, with different number of jobs (25, 50 and 100). We solved ten different problems in each case. To set the reference, makespan was calculated for 1000 randomly generated job sequences. The sequences constructed by the two heuristics (DECf and GG) were also calculated. For the same problem a GA search was done to produce a good schedule. In the second study the

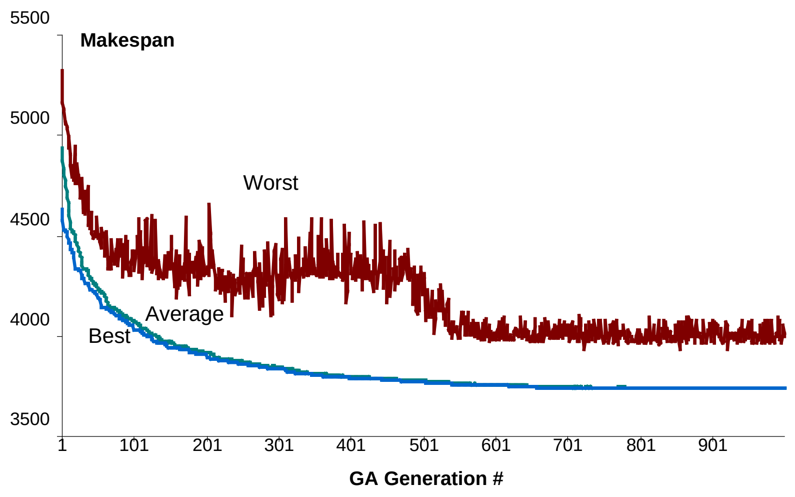

Figure 5.10: Typical convergence profile for a well-parameterized GA.

Gilmore-Gomory solution for the two-machine problem was also obtained.

For GA we used the following parameters: $P_s = 1000$, $P_c = 0.95$, and $P_\mu = 0.05$. The best sequence obtained upto 1000 generations of GA iteration was recorded in each GA run. The number of generations the GA took to find the best makespan with no further improvement showing in upto 1000 generations is indicated in parenthesis in the tables that follow.

In the first study we used the DECf solution as a member of the initial population of GA solutions. In the second set of studies the GA was hybridized by introducing the Gilmore-Gomory solution into the initial population. In the tables below the CPU times indicated are for an Intel pentium III, 722 MHz processor.

5.7.1 Study I–Summary of results when average job processing times are not equal

The numerical results are displayed in Tables 5.7.1, 5.7.1 and 5.7.1.

- In this study stage 2 is the bottleneck.

Table 5.7: Unequal job processing times: number of jobs=25

No.	Rand.Best	DECf	GG	Pure GA	Hybrid DECf+GA	CPU time(Pure GA)/Sec
1	288	287	311	254(596)	252(49)	124.759
2	250	238	262	227(35)	222(34)	125.800
3	283	274	308	258(999)	253(44)	112.752
4	329	317	352	299(304)	294(859)	109.016
5	317	307	353	286(58)	280(60)	128.705
6	263	248	280	231(442)	228(832)	112.962
7	220	224	229	206(28)	206(28)	104.089
8	282	292	324	259(428)	259(428)	114.514
9	217	228	237	192(812)	192(812)	104.570
10	257	251	296	226(320)	224(61)	112.401

Table 5.8: Unequal job processing times: number of jobs=50

No.	Rand.Best	DECf	GG	Pure GA	Hybrid DECf+GA	CPU time(Pure GA)/Sec
1	618	603	679	541(167)	544(97)	186.898
2	597	606	648	531(208)	531(208)	193.718
3	572	559	616	497(114)	485(505)	204.043
4	607	599	637	539(179)	540(78)	198.855
5	580	552	646	509(110)	505(448)	184.555
6	616	611	653	542(102)	540(500)	179.127
7	580	560	620	498(976)	501(101)	175.953
8	559	559	559	559(1)	559(1)	108.526
9	595	582	614	521(991)	523(289)	202.130
10	485	474	481	460(104)	460(42)	197.634

- DECf heuristic outperforms the Gilmore-Gomory heuristic. This is not unexpected since the processing time in stage 1 is not very relevant here. We also found that randomly generated *initial* solutions generally performed better than the Gilmore-Gomory solution.

- Pure GA results (without hybridizing) were always better than both the heuristics applied alone.

- The hybridized GA consistently performed better than the pure GA in terms of either finding a better solution for the same number of iterations (generations) or finding an equally good solution in fewer GA iterations.

Table 5.9: Unequal job processing times: number of jobs=100

No.	Rand.Best	DECf	GG	Pure GA	Hybrid DECf+GA	CPU time(Pure GA)/Sec
1	1232	1196	1319	1063(409)	1051(473)	356.021
2	1174	1138	1223	997(410)	993(595)	366.256
3	1178	1144	1264	1018(715)	1007(871)	347.479
4	732	714	741	697(121)	697(72)	308.552
5	1219	1155	1278	1022(352)	1016(960)	385.484
6	1234	1211	1360	1058(255)	1055(329)	366.646
7	1275	1270	1334	1096(269)	1097(325)	366.577
8	1081	1048	1067	920(299)	903(329)	365.104
9	1139	1119	1238	1019(171)	1016(889)	366.605
10	1291	1251	1392	1107(255)	1093(486)	391.723

5.7.2 Study 2–Results when job processing times are equal

The numerical results are displayed in Tables 5.7.2, 5.7.2 and 5.7.2.

Table 5.10: Equal job processing times: number of jobs=25

No.	Rand.Best	GG		DECf	Pure GA	Hybrid GG+GA	CPU time(Pure GA)/Sec
		2M	3M				
1	447	504	449	504	404(35)	404(35)	110.639
2	461	459	453	477	452(5)	452(5)	99.022
3	480	535	485	516	439(218)	439(218)	110.528
4	434	450	427	494	413(23)	413(13)	105.041
5	485	484	484	500	484(10)	484(1)	100.804
6	430	479	421	460	380(32)	380(35)	109.086
7	549	543	538	623	535(12)	535(9)	102.547
8	457	451	445	532	446(12)	444(8)	103.368
9	548	548	546	569	546(12)	546(1)	96.268
10	532	539	534	551	530(7)	530(7)	96.699

- As expected, the three-machine Gilmore-Gomory solution was always better than the two-machine case due to the availability of extra processing capacity in the second stage. But this improvement was marginal because processing times in the two stages were of comparable magnitude (both had the same averages).

- The Gilmore-Gomory heuristic performed better than DECf. This is explained

Table 5.11: Equal job processing times: number of jobs=50

No.	Rand.Best	GG		DECf	Pure GA	Hybrid GG+GA	CPU time(Pure GA)/Sec
		2M	3M				
1	1052	1036	1034	1096	1034(49)	1034(1)	172.828
2	1001	992	990	1052	990(5)	990(1)	173.849
3	900	900	858	948	826(29)	829(34)	191.154
4	928	898	886	985	884(80)	884(7)	171.186
5	1101	1070	1055	1127	1054(34)	1054(10)	171.536
6	929	896	873	979	862(43)	862(33)	181.050
7	1001	981	963	1065	954(38)	954(23)	171.025
8	945	895	895	1010	895(15)	895(1)	169.263
9	1005	998	993	1034	993(16)	993(1)	161.442
10	932	895	884	976	882(36)	882(14)	171.086

Table 5.12: Equal job processing times: number of jobs=100

No.	Rand.Best	GG		DECf	Pure GA	Hybrid GG+GA	CPU time(Pure GA)/Sec
		2M	3M				
1	1950	1843	1813	2051	1797(104)	1797(50)	339.277
2	1938	1861	1816	2030	1790(74)	1788(82)	508.601
3	1616	1953	1545	1761	1513(85)	1513(10)	310.656
4	1972	1887	1882	2063	1882(77)	1882(1)	363.743
5	2051	1984	1952	2167	1939(120)	1937(2)	370.062
6	1925	1825	1823	2028	1823(104)	1823(1)	641.582
7	2002	1948	1876	2044	1828(100)	1828(71)	353.137
8	1804	1758	1691	1897	1639(89)	1639(102)	345.867
9	1929	1814	1809	1995	1809(135)	1809(1)	337.014
10	1887	1802	1793	1994	1793(123)	1793(1)	334.601

by the fact that the processing time of stage one is relevant in this case. Also, the best solutions of randomly generated solutions generally performed better than DECf.

- Pure GA results were always better than either of the two heuristics.
- The hybrid GA performed better than pure GA for equal number of iterations run or in finding an equally good solution in fewer GA iterations.

5.8 Concluding Remarks

In this chapter we described heuristic methods for minimizing the makespan in a two-stage blocking flowshop with material handling constraints. Such problems are encountered in production system design (e.g. automotive paint shops), service facilities design (e.g. car wash), or specialty industries such as petrochemical processing. Surprisingly, even the two-stage version of this problem is not easy to solve due to the material handling restriction present. We prove that the problem is unary NP-hard. Computational studies indicate that metaheuristic methods such as genetic algorithms can be fruitfully used to solve practical instances of this problem. We also explore the special structural features of this problem and develop problem-specific solution construction heuristics for different job processing time scenarios. Such heuristics can provide efficiency in solution evolution by hybridizing–by providing good starting points for a method like the genetic algorithm, particularly when the problem is large and computational efficiency may be paramount.

CHAPTER 6

CONCLUSION AND DIRECTION FOR FUTURE RESEARCH

This chapter concludes the findings of this research, and discusses the scope and direction of future research. This study dealt with the optimization problems arising in practical and real world situations. We discussed three specific applications in production and operations management, telecommunications and scheduling areas. We developed analytical formulations and heuristics and meta-heuristic methods for these problems.

The first problem the study addressed was that of computing forecast horizons in multi-period decision environments. Specifically, we looked at the Dynamic Lot-size problem. In this study we demonstrated the viability of integer programming in solving these problems. Also, we introduced a new type of weak forecast horizon - forecast horizons subject to discrete future demand. There are several interesting directions worth pursing along this study. We propose the following:

- Most state-of-the-art general-purpose integer programming solvers use a variety of techniques - search strategies, e.g., depth-first-search, breadth-first-search, best-bound, etc; different classes of cutting planes, e.g., disjunctive cuts, clique cuts, flower cover cuts, etc; heuristics - to make the search for integer solutions efficient. Typically, given a specific instance, solvers differ in nature of these techniques and the extent to which they are used. Consequently, for a specific class of problems such as the one discussed in this Chapter, there does not seems to be a great deal of consistency across solvers. Our experience indicates that integer programming is a practicable approach to compute forecast horizons; a promising direction for future work is an investigation of the structure of these problems leading to a specialized integer programming solvers.

- Throughout this study, we have used the dynamic lot-size problem (and its variants) to illustrate our analysis of discrete forecast horizons. Minimal use of structural properties of the problem domain, and the constraint-based nature of integer programming lead us to believe that our approach can be extended to investigate forecast horizons for a wide variety of multi-period dynamic decision problems.

- As with continuous forecast horizons, a general characterization of demand vectors for which a discrete forecast horizon exists remains an open question. Due to its prevalence in practice, a characterization for integer future demands is especially relevant. The additional structure imposed by the discreteness assumption may help make such a characterization easier to obtain than that in the continuous demand case. Another fundamental topic is the robustness of discrete forecast horizons to changes in data.

- An important consequence of the discreteness assumption is the reduction (with respect to continuous demands) in the length of the minimal forecast horizon. While a significant reduction has been observed in our numerical experiments, there is no theoretical analysis available that provides an estimate of this reduction.

- This study investigates discrete forecast horizons under the assumption that all near-term demands – from the current period until the forecast horizon – are deterministic. In situations when these demands cannot be predicted with certainty, a scenario-based stochastic integer programming approach can be used. We are actively investigating the feasibility of this approach.

- Sethi and Sorger [99] have proposed a theory of rolling horizon decision making where the first-period decision consists of the length of the time interval for which demand must be forecasted and the production quantity. This process is repeated in each period. The objective is to maximize the total expected cost of inventory holding, shortage, and forecasting over a given finite horizon.

It would be of interest to see if their theory could be implemented with the integer programming framework developed in this paper.

The second problem this study looked at was that of traffic grooming for wavelength routed optical networks. We formulated mixed integer programs for this problem and developed heuristics based on column generation. Furthermore, we obtained bounds using Lagrangian relaxation. Our heuristic for obtaining a traffic maximizing set of routes is based on a Price-and-Bound algorithm to solve the LP relaxation, followed by Branch-and-Bound using only the columns generated at the root node. It might be possible to improve the solution presented in this study by employing a Branch-and-Price algorithm that generates columns at every node of the tree. This approach is a possible direction for future research. However, it has to be noted that the improved solutions may be obtained at the expense of increased computing time. Another extension could be to consider routing and wavelength assignment together in a single formulation. Since we only consider single-hop networks, traffic routing using multi-hops is another extension that is worth investigating.

The third and the final problem we discussed was the blocking flow-shop scheduling problem with material handling constraint. We analyzed the structural properties of this problem and developed heuristics and a search method using genetic algorithms. Furthermore, we analyzed the efficiency of hybrid-genetic algorithms by combining heuristic solutions with the genetic algorithm. Interesting future research work is to obtain exact solutions using time-index integer programming formulations. Also, other meta-heuristic approaches such as simulated annealing and tabu-search techniques are worth perusing.

REFERENCES

[1] Abadi, I.N.K., N.G. Hall and C. Sriskandarajah, C. 2000. Minimizing Cycle Time in a Blocking Flowshop, *Operations Research*, 48, 177–180.

[2] Ahuja,R.K., Magnanti, T.L., and Orlin, J.B., 1993. Network Flows: Theory, Algorithms and Applications, *Prentice Hall*.

[3] Ajmone, M., Leonardi, E., Mellia, M. and Nucci, A., 2002. Design of Topologies in Wavelength Routed IP Networks, *Journal of Photonic Network Comminications,* Vol. 3, No. 3, 423–442.

[4] Aronson, J. E., Morton, T. E., and Thompson, G. L. (1984) A Forward Algorithm and Planning Horizon Procedure for the Production Smoothing Problem Without Inventory, *European Journal of Operational Research,* Vol. 15, 348-365.

[5] Arthanari, T.S. 1974. On Some Problems of Sequencing and Grouping, Ph.D. Thesis, Indian Statistical Institute, Calcutta, India.

[6] Bagchi, T. P., and Deb, K. 1996. Calibration of GA parameters: the design of experiments approach, *Computer Science and Informatics*, 26(3).

[7] Bagchi, T.P., Naranpanawe, S., and Sriskandarajah, C., 2003. A A Blocking Flow-shop Scheduling Problem with Material Handling Constraint, *International Journal of Operations and Quantitative Management*, Vol.9 No.1.

[8] Baker, K.R. 1974. *Introduction to Sequencing and Scheduling.* Wiley, New York.

[9] Baker, K.R. 1992. *Elements of Sequencing and Scheduling.* Amos Tuck School of Business Administration, Dartmouth College, Hanover, NH.

[10] Balas, E., Ceria, S. and Cornuéjols, G. (1996) Mixed 0-1 Programming by Lift-and-Project in a Branch-and-Cut Framework, *Management Science,* Vol. 42, 1229-1246.

[11] Banerjee, D. and Mukherjee, B., 1996. A Practical Approach for Routing and Wavelength Assignment in Large Wavelength-Routed Optical Networks, *IEEE J. Select Areas Commun.* , Vol. 14., No. 5, 903–908.

[12] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1998) Branch-and Price: Column Generation for Solving Integer Programs, *Operations Research,* Vol. 46, 316-329.

[13] Baroni, S. and Bayvel, P., 1997. Wavelength Requirements in Arbitrarily Connected Wavelength-Routed Optical Networks, *Journal of Lightwave Technology,* Vol. 15, No. 2, 242–251.

[14] Bastian, M. and Volkmer, M. (1992) A Perfect Forward Procedure for a Single Facility Dynamic Location Relocation Problem, *Operations Research Letters,* Vol. 12, 11-16.

[15] Bean, J., Smith, R. L., and Yano, C. (1987) Forecast Horizons for the Discounted Dynamic Lot Size Model Allowing Speculative Motive, *Naval Research Logistics Quarterly,* Vol. 34, 761-774.

[16] Bensoussan, A., Crouhy, J., and Proth, J.M. (1983) Mathematical Theory of Production Planning, *Advanced Series in Management, North-Holland, Amsterdam.*

[17] Bes, C. and Sethi, S. P. (1988) Concepts of Forecast and Decision Horizons: Applications to Dynamic Stochastic Optimization Problems, *Mathematics of Operations Research,* Vol. 13, No. 2, 295-310.

[18] Blocher, J. D. and Chand, S. (1996) A Forward Branch-and-Search Algorithm and Forecast Horizon Results for the Changeover Scheduling Problem, *European Journal of Operational Research,* Vol. 91, 456-470.

[19] Buten, R.E. and V.Y. Shen. 1973. A Scheduling Model for Computer Systems with Two Classes of Processors, *Proceedings Sagamore Computer Conference on Parallel Processing*, 130–138.

[20] Chand, S., Hsu, V.N., Sethi, S.P. (2002) Forecast, Solution, and Rolling Horizons in Operations Management Problems: A Classified Bibliography, *Manufacturing & Service Operations Management,* Vol. 4, No. 1, 25-43.

[21] Chand, S., and Morton, E. (1982) A Perfect Planning Horizon Procedure for a Deterministic Cash Balance Problem, *Management Science,* Vol. 28,No. 6, 652-669.

[22] Chand, S., and Morton, E. (1986) Minimal Forecast Horizon Procedures for Dynamic Lot Size Models, *Naval Research Logistics Quarterly,* Vol. 33, 111-121.

[23] Chand, S. and Sethi, S. P. (1982) Planning Horizon Procedures for Machine Replacement Models with Several Possible Replacement Alternatives, *Naval Research Logistics Quarterly,* Vol. 29, 483-493.

[24] Chand, S., Sethi, S.P., and Proth, J.M. (1990) Existence of Forecast Horizon in Undiscounted Discrete-Time Lot Size Models, *Operations Research,* Vol. 38, No. 5, 884-892.

[25] Chen, A., Vempati, V. S., and Aljaber, N. 1995. An application of genetic algorithms for owshop problems, *European Journal of Operational Research*, 80, 389–396.

[26] Chiu, Angela L., and Modiano, Eytan H., 2000. Traffic Grooming Algorithms for Reducing Electronic Multiplexing Costs in WDM Ring Networks , *Journal of Lightwave Technology* , vol.16(1), 2–12.

[27] Chu Xiaowen, Bo Li,Chlamtac I., 2003. Wavelength converter placement under different RWA algorithms in wavelength-routed all-optical networks *IEEE Transactions on Communications,* , Volume: 51 Issue: 4 , April 2003, 607–617.

[28] Cinkler, T., 2003. Traffic and λ Grooming, *IEEE Network*, March/April 2003.

[29] Coffman E. G.,Jr., Garey M. R., and Johnson D. S., 1997. Approximation Algorithms for Bin Packing: A Survey, *Approximation Algorithms for NP-Hard Problems, D. Hochbaum (editor)*, PWS Publishing, Boston, 46–93.

[30] Connoly, D. 1992. General purpose simulated annealing, *Journal of Operational Research Society*, 43(3), 495–505.

[31] Conway, R.W., W.L. Maxwell and L.W. Miller. 1967. *Theory of Scheduling.* Addison-Wesley, Reading, MD.

[32] Daskin, M. S., Hopp, W. J., and Medina, B. (1992) Forecast Horizons and Dynamic Facility Location Planning, *Annals of Operations Research,* Vol. 28, 125-151.

[33] Davis, L. 1991. *Handbook of genetic algorithms*, New York, Van Nostrand Reinhold.

[34] Dawande, M., Gavirneni S., Naranpanwe S., and Sethi S.P. (2004) Integer Programs for Computing Minimal Forecast Horizons, *Journal of Mathematical Modeling and Algorithms,* forthcoming.

[35] Dawande, D., Gavirneni, S., Naranpanawe, S., and Sethi, S., 2005. Discrete Forecast Horizons: Analysis and Computations, *working paper.*

[36] Dobson, G. and C.A. Yano. 1994. Cyclic scheduling to minimize inventory in a batch flow line. *European Journal of Operational Research*, (75)2, 441-461.

[37] Dutta, Rudra, and Rouskas, George N., 2002. Traffic Grooming in WDM Networks: Past and Future, *IEEE Network*, Nov-Dec 2002, 46–56.

[38] Erben, W. 1995. Time tabling using artiRcial neural nets and genetic algorithms, *Proceedings of the International Conference in Ales*, France, 30–32.

[39] Even, S., Itai, A., Shamir, A., 1976. On the complexity of timetable and multicommodity flow problems, *SIAM Journal of Computing,* Vol. 5, 691–703.

[40] Federgruen, A. and Tzur, M. (1994) Minimum Forecast Horizons and a New Planning Procedure for the General Dynamic Lot Sizing Model: Nervousness Revisited, *Operations Research,* , Vol. 42. No. 3, 456-468.

[41] Federgruen A. and Tzur, M. (1995) Fast Solution and Detection of Minimal Forecast Horizons in Dynamic Programs with a Single Indicator of the Future: Applications to Dynamic Lot-Sizing Models, *Management Science,* Vol. 41, 874-893.

[42] Federgruen A. and Tzur, M. (1996) Detection of Minimal Forecast Horizons in Dynamic Programs with Multiple Indicators of the Future, *Naval Research Logistics Quarterly,* Vol. 43, 169-189.

[43] Fisher, M.L., 1985. An application oriented guide to Lagrangian relaxation, *Interfaces*, 15,1–18.

[44] Gaither, N. 1990. *Production and Operations Management*, fourth edition. The Dryden Press, Chicago.

[45] Garey, M.R. and D.S. Johnson. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco.

[46] Garey, M.R., D.S. Johnson and R. Sethi. 1976. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research* 1, 117-129.

[47] Gerstel, Ornan, Ramswami, Rajiv, and Sasaki, Galen H., 2000. Cost-Effective Traffic Grooming in WDM Rings , *IEEE/ACM Transactions on Networking*, vol.8(5), 618–630.

[48] Gilmore, P. and R. Gomory. 1964. Sequencing a One-state Variable Machine: A Solvable Case of the Travelling Salesman Problem, *Operations Research*, 12, 665–679.

[49] Glover, F. and Laguna, M. 1997. *Tabu search*, Dordrecht, Kluwer Academic Publishers.

[50] Goldberg, D. E. 1989. *Genetic algorithm in search, optimization and machine learning*, Reading, MA, Addison-Wesley.

[51] Graham, R.L., E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan. 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics* 5, 287-326.

[52] Grotschel, M., Junger, M. and Reinelt, G. (1984) A Cutting Plane Algorithm for the Linear Ordering Problem, *Operations Research,* Vol. 32, 1195-1230.

[53] Gupta, R., Naranpanawe, S., and Sriskandarajah, C., 2004. A Traffic Grooming Algorithm for Wavelength Routed Optical Networks, *working paper.*

[54] Hall, N.G. and C. Sriskandarajah. 1996. A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, 44, 510-525.

[55] Holland, J. H. 1975. *Adaption in natural and artiRcial system.* Ann Arbor, University of Michigan Press.

[56] Hu, J.Q., and Leida, B., 2002. Traffic Grooming, Routing, and Wavelength Assignment in Optical WDM Mesh Networks, *presented at the Sixth INFORMS Telecommunications Conference, Boca Raton, Fla., 10-13 March 2002* .

[57] Jia, X., 1998. A Distributed Algorithm of Delay-Bounded Multicast Routing for Multimedia Applications in Wide-Area Networks, *IEEE/ACM Transactions on Networking,* Vol. 6, 828–837.

[58] Jia, X., Du, D., Hu, X., Lee, M. and Gu, J., 2001. Optimization of Wavelength Assignment for QoS Multicast in WDM Networks, *IEEE Transactions on Communications* Vol. 49, No. 2, 341–350.

[59] Jia, X., Hu, X., Ruan, L. and Sun J., 2002. Multicast Routing, Load-Balancing and Wavelength Assignment on a Tree of Rings, *IEEE Communication Letters* , Vol. 6, No. 2, 78–81.

[60] Johnson, R. E. and McClain, J. O. (1978) On Further Results on Planning Horizons in the Production Smoothing Problem, *Management Science,* Vol. 24, 1774-1776.

[61] Kamoun, H. and C. Sriskandarajah. 1993. The complexity of scheduling jobs in repetitive manufacturing systems. *European Journal of Operational Research* 70, 350-364.

[62] Karabati, S. and P. Kouvelis. 1994a. The interface of buffer design and cyclic scheduling decisions in deterministic flow lines. *Annals of Operations Research* 50, 295-317.

[63] Karabati, S. and P. Kouvelis. 1994b. Cyclic scheduling in flow lines: modeling observations, effective heuristics and a cycle time minimization procedure. Working paper, Fuqua School of Business, Duke University, Durham, N.C.

[64] Kleindorfer, P. and Leiber, Z. (1979) Algorithms and Planning Horizon Results for Production Planning Problems with Separable Costs, *Operations Research,* Vol. 27, 875-887.

[65] Langston, M.A. 1987. Interstage Transportation Planning in the Deterministic Flowshop Environment, *Operations Research*, 35, 4, 556-564.

[66] Lawler, E.L., J.K, Lenstra, A,H.G. Rinnooy Kan and D.B., Shmoys 1993. Sequencing and scheduling: algorithms and complexity. In Graves SC et al. (eds). *Handbooks in Operations Research and Management Science* 4, pp 445-552.

[67] Lee, Kyungsik Lee, Kang, Kug Chang, Lee, Taehan,and Park, Sungsoo, 2002. An Optimization Approach to Routing and Wavelength Assignment in WDM

All-Optical Mesh Networks without Wavelength Conversion, *ETRI Journal*, vol.24(2), April 2002, 131–141.

[68] Lundin, R.A., and Morton, E. (1975) Planning Horizon for the Dynamic Lot Size Model: Zabel vs. Protective Procedures and Computational Results, *Operations Research,* Vol. 23, No. 4, 711-734.

[69] Matsuo, H. 1990. Cyclic sequencing problems in a two machine permutation flowshop: complexity, worst case and average case analysis. *Naval Research Logistics* 37, 679-694.

[70] McCormick, S.T., M.L. Pinedo, S. Shenker and B. Wolf. 1989. Sequencing in an assembly line with blocking to minimize cycle time. *Operations Research* 37, 925-935.

[71] Mexatiotis, K.S. and J.E. Psarras. 2001. Building Ontology for Production Scheduling Systems: Towards a Unified Methodology, *Journal of Information Technology Theory & Application*, 3(1), 1–12.

[72] Michaelewicz, Z. 1992. *Genetic algorithms + data structures = evolution programs.* Berlin: Springer-Verlag.

[73] Modiano, Eytan, and Lin, Philip J., 2001. Traffic Grooming in WDM Networks , *IEEE Communications Magazine*, July 2001, 124–129.

[74] Montgomery Douglas, C. 1991. *Design and analysis of experiments (3rd ed.).* New York, Wiley.

[75] Mukherjee, B.,1997. Optical Communication Networks, McGraw-Hill, Series on Computer Communications.

[76] Murata, T. and Ishibuchi, H. 1994. Performance evaluation of genetic algorithms for Flowshop scheduling problems, *Proceedings of the IEEE Conference on Genetic Algorithms*, pp. 812–817

[77] Nair, S. K. and Hopp, W. J. (1992) A Model for Equipment Replacement Due to Technological Obsolescence, *European Journal of Operational Research,* Vol. 63, 207-221.

[78] Nawaz, M., E.E. Enscore, Jr. and I. Ham. 1983. A Heuristic Algorithm for m-machine, n-job Flowshop Sequencing Problem. *Omega* 11, 91-95.

[79] Nemhauser, G.L., Savelsbergh, M.W.P and Sigismondi, G.S. (1994) *MINTO, a Mixed INTeger Optimizer,* Oper. Res. Letters 15, 47-58.

[80] Nemhauser G.L., Wolsey L.A.,1988. Integer Programming and Combinatorial Optimization, *Wiley-Interscience.*

[81] Osman, I.H. and C.N. Potts. 1989. Simulated annealing for permutation flowshop scheduling. *Omega* 17, 551-557.

[82] Ozdalgar, Asuman E. and Bertsekas, Dimitri P., 2000. Routing and Wavelength Assignment in Optical Networks *IEEE/ACM Transactions on Networking*, vol.11 No.2.

[83] Padberg, M. W. and Rinaldi, G. (1987) Optimization of a 532-city Symmetric Travelling Salesman Problem by Branch and Cut, *Operations Research Letters,* Vol. 6, 1-7.

[84] Pinedo, M. 1995. *Scheduling: Theory, Algorithms and Systems.* Prentice-Hall, Englewood Cliffs, N.J.

[85] Portmann, M. C. 1996. Scheduling methodology: optimization and compu-search approaches I. In A. Artiba, and S. E. Elmaghraby, *Production and scheduling of manufacturing system.* London: Chapman and Hall.

[86] Potts, C.N. 1980. An adaptive branching rule for the permutation flowshop problem. *European Journal of Operational Research* 5, 19-25.

[87] Rajagopalan, S. (1992) Deterministic Capacity Expansion Under Deterioration, *Management Science,* Vol. 38, 525-539.

[88] Rajagopalan, S. (1994) Capacity Expansion with Alternative Technology Choice, *European Journal of Operational Research,* Vol. 77, 392-403.

[89] Ramaswami, Rajiv and Sivarajan, Kumar N., 1995. Routing and Wavelength Assignment in All-Optical Networks *IEEE/ACM Transactions on Networking*, vol.3 No.5.

[90] Rempala, R. (1989) Forecast Horizons in Nonstationary Markov Decision Problems, *Optimization,* Vol. 20, 853-857.

[91] Rinnooy Kan, A.H.G. 1976. *Machine Scheduling Problems.* Martinus Nijhoff, The Hague.

[92] Röck, H. 1984. The three machine no-wait flowshop problem is NP-complete, *Journal of the Association for Computing Machinery* 31, 336-345.

[93] Salvador, M.S. 1973. A Solution of a Special Class of Flowshop Scheduling Problems, *Proceedings of the Symposium on the Theory of Scheduling and its Applications*, Springer-Verlag, Berlin, 83–91.

[94] Sethi, S. (1971) A Note on a Planning Horizon Model of Cash Management, *Journal of Financial and Quantitative Analysis,* Vol. 6, 659-665.

[95] Sethi, S. P. and Bhaskaran, S. (1985) Conditions for Existence of Decision Horizons for Discounted Problems in a Stochastic Environment: A Note, *Operations Research Letters,* Vol. 4, No. 2, 61-64.

[96] Sethi, S. P. and Sorger, G. (1991) A Theory of Rolling Horizon Decision Making, *Annals of Operations Research,* Vol. 29, 387-416.

[97] Sriskandarajah, C. 1986. L'ordonnancement dans les Ateliers: Complexité et Algorithmes Heuristiques, Doctorate Thesis, Laboratoire d'Automatique, École Nationale Supérieure d'Ingénieurs Électriciens de Grenoble, France.

[98] Sriskandarajah, C. and P. Ladet. 1986. Some No-Wait Shops Scheduling Problems: Complexity Aspect, *European Journal of Operational Research*, 24, 424–445.

[99] Sriskandarajah, C. and S.P. Sethi. 1989. Scheduling Algorithms for Flexible Flow shops: Worst and Average Case Performance, *European Journal of Operational Research*, 43, 143–160.

[100] Turner, S. and D. Booth 1987. Comparison of heuristics for flowshop sequencing. *Omega* 15, 75-78.

[101] Uckun, S., Bagchi, S., Kawamura, K., and Miyabe, Y. 1993. Managing genetic search in job shop scheduling, *IEEE Expert*, 8(5), 15–24.

[102] Wagner, H. M. (2004) Comments on "Dynamic Version of the Economic Lot Size Model", Vol. 50, No. 12, 1775-1777.

[103] Wagner, H.M, and Whitin, T.M (1958) Dynamic Version of the Economic Lot Size Model, *Management Science,* Vol. 5, No. 1, 89-96.

[104] Wittrock, R.J. 1988. An Adaptable Scheduling Algorithm for Flexible Flow Lines, *Operations Research*, 36, 445–453.

[105] Waxman, B. M., 1988. Routing of Multipoint Connections, *IEEE Select. Areas Communications,* Vol. 6, 1617–1622.

[106] Yoon, Moon-Gil.,2001. Traffic Grooming and Lightpath Routing in WDM Ring Networks with Hop-count constraints , *ICC 2001,IEEE International Conference on Communications* , 731–737.

[107] Zang, Hui, Jue, Jason P., and Mukherjee Biswanath, 2000. A Review of Routing and Wavelength Assignment Approaches for Wavelength-Routed Networks, *Optical Networks Magazine*, January 2000, 47–59 .

[108] Zegura, E. W ., Calvert, K. L., and Bhattacharjee, S., 1996. How to model an Internetwork, *Proceedings of the 15th IEEE INFOCOM Annual Joint Conference of the IEEE Computer and Communications Societies, San Francisco, CA.*

[109] Zhu, Hongyue , 2003. Zang, Hui, Zhu, Keyao and Mukherjee, Biswanath, Novel Generic Graph Model for Traffic Grooming in Heterogeneous WDM Mesh Network *IEEE/ACM Transactions on Networking*, vol.11 No.2.

[110] Zhu, K., and Mukherjee, B., 2002. Traffic Grooming in an Optical WDM Mesh Network, *IEEE Journal on Selected Areas of Communications* , 20(1), 122–133.

VITA

Sanjeewa Naranpanawe was born in the city of Kandy, Sri Lanka, the youngest son of Mr. Ranasinghe B. Naranpanawe and Mrs. Chandralatha L. Naranpanawe. He completed his primary education at St.Anthony's college, Kandy. He received his Bachelor of Engineering degree in Production Engineering from the University of Peradeniya, Sri Lanka in 1998. He obtained his masters degree in Management and Administrative Sciences from the University of Texas at Dallas in 2004.