ADVANCED PLANNING SYSTEMS FOR HARD DISK DRIVE ASSEMBLY

NG TSAN SHENG

NATIONAL UNIVERSITY OF SINGAPORE

2004

ADVANCED PLANNING SYSTEMS FOR HARD DISK DRIVE ASSEMBLY

NG TSAN SHENG

(B.Eng.(Hons), National University of Singapore)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgments

This work owes much credit to the guidance of my research supervisors: Dr Lee Loo Hay and A. P. Chew Ek Peng. I am thankful to them for their invaluable advice and also the many hours of discussions and brainstorming despite their very hectic schedules.

Special gratitude goes out to the staff of the Production Planning and Control Department and the New Business Development Department of Maxtor Singapore for their generosity and help during my attachment. I am also thankful to the Department of Industrial and Systems Engineering in the university for the provision of a very conducive research environment. I would like to extend my acknowledgments to: Lai Chun, for her kind assistance and patience in handling my administrative demands. Teng Suyan, for her help and discussions in my research project. Wee Tat, for providing much help in the typesetting of this thesis. Mr Lau Pak Kai and Ms Yao Qiong, for their assistance in using the laboratory facilities and resources. Yew Loon, Mong Soon, Ivy, Yenping, and also the colleagues in Quality and Reliability laboratory for their friendship through these few years in the Department.

Also deserving of gratitude are my parents and family, for their support and encouragement in pursuing my post-graduate studies. Finally, to Grace, for her love and understanding, and for leading me back to know God, without whom none of these would have been possible.

Contents

List of Figures

List of Tables

Notation For Problem Parameters

t	production period, $t = 1, \cdots, T$
p	product component
θ	build-type
k	customer
v	component vendor
d_k	random customer k demand in units of product
d_t^k	deterministic customer k demand in units of product due in t
c_t	manpower resource available in units of products built in t
g	per unit shortage cost
h	per unit holding cost
g_t^k	per unit tardiness cost of k in t
r_k^p	number of units of p required to build per unit of k
m_v	component supply from vendor v
$m_{v,t}$	component from vendor v arriving in t
P	set of all components p
V_p	set of all vendors of component $p \in P$

V_p^k set of all vendors of $p \in P$ that is acceptable in the AVM for customer k

ϑ set of all possible build-types θ

K set of all customers k

K_v set of all $k \in K$ that can use vendor $v \in V_p$ for component $p \in P$ to make the final product

$\phi(\cdot)$ standard normal density function

$\Phi(\cdot)$ standard normal distribution function

$G(\cdot)$ standard normal 'loss' function, i.e. $G(\kappa) = \int_{\kappa}^{\infty}(z-\kappa)\cdot\phi(z)dz$, where κ, $z \in \Re$

μ_k mean of customer k demand

σ_k standard deviation of cusomer k demand

$$\Gamma_{k,v,v'} = \begin{cases} 1 & \text{if for } k\text{, vendor } v \text{ of component } p \text{ cannot be used together} \\ & \text{with vendor } v' \text{ of component } p', \text{ where } v \in V_p,\ v' \in V_{p'}\ \forall p,\ p' \in P. \\ 0 & \text{otherwise.} \end{cases}$$

Summary

This dissertation studies a new class of production planning and scheduling problems motivated by an actual manufacturer of hard-disk drives. In order to distinguish itself in the technologically saturated and highly competitive electronic goods market, the manufacturer offers its customers the *approved vendor matrix* as a competitive advantage. An approved vendor matrix allows each customer to pick and choose the various product component vendors for individual or pairs of components constituting their product. Two main problems are considered in this work: a multi-period production scheduling problem, and a stochastic production planning problem. We also study various extensions of these two problems. In the case when the presence of the approved vendor matrices are not considered, these problems can be modeled and solved easily using linear and integer programming techniques. The approved vendor matrices however, complicate these formulations, and render their solution via general-purpose solvers extremely inefficient for realistic problem sizes. This work presents the appropriate mathematical models for the problems studied, and then develop the specialized methods and algorithms to solve them. In particular, our algorithms involve novel applications of column

and cut generation, decomposition, branch-and-bound and branch-and-price methods. We demonstrate that our algorithms are able to outperform general purpose techniques significantly in terms of the computation times required to solve the problems. This is a valuable and practical contribution for the decision makers, who may be looking to apply optimization to solve their planning problems but cannot afford the enormous amounts of computational resources often required by general purpose methods. To the best of our knowledge this work is also the pioneering effort that investigates this class of problems in production planning research.

Chapter 1

Introduction

This work is about optimization models for production planning and scheduling (PPS) systems. Our focus is on a specific class of PPS problems characterized by the hard-disk drive (HDD) industry. Proponents of highly successful manufacturing practices such as *lean production* tend to regard 'operations research approaches' in manufacturing planning as rigid and contrary to world class manufacturing practices [94]. In the next section, we will first motivate the relevance of optimization models for PPS problems in the modern-day manufacturing environment. This motivation is then applied to the case of HDD manufacturing in §1.2. We highlight the essential characteristics of HDD industry, and in particular how the *modular design* of HDDs provides opportunities to build new competitive advantages for the company. These often translate into new business rules that may cause ramifications on downstream activities like production planning. The impact of one such business rule on an actual HDD manufacturer leads to the class of PPS

problems that is the scope of our research, which is discussed in §1.3. Finally in §1.4 we outline the presentation of this thesis.

1.1 The Relevance of Optimization in Production Planning With Modern Business Rules

It is well-known in both the industry and academia that the competent management of logistics provide valuable cost-saving opportunities for manufacturers. For example, if 4% of the total accumulated inventories in China (forming 50% of its national GDP as of year 2000) can be shaved off, an estimated US 495 billion dollars can be saved [62]. A company with well-managed in supply chain operations can potentially have up to 50 % cost advantage over competitors [99]. The challenges of exploring cost-reduction strategies, of solving problems in managing and optimizing logistical systems has continued to prompt research interest in areas of distribution planning, inventory management and PPS. In recent years, manufacturing practices such as JIT (*just-in-time* [53]) and lean production [114] have seen much success over traditional planning systems such as MRPII (manufacturing resources planning), especially in repetitive manufacturing organizations around the world. While MRPII was developed by data processing professionals and did not begin as an optimization model, it is often confused with the operations research approach of problem-solving. This has led to the misconception among some proponents of practices such as lean production that PPS using operations research

approaches are "*complex, inhibit change, foster mediocrity, and are inflexible*" [94].

On the other hand, heightened competition in the electronics goods industries to sustain or expand market shares has often compelled manufacturers to re-evaluate business strategies. For instance, in discussing the limits of applying lean production principles, Cusmano [35] remarks that "*the parity of performance in core processes is forcing manufacturers to seek competitive advantage not simply by following lean principles that everyone will know and be implementing, but by defining other domains of competition*". Hence, in commodity-industry situations such as HDD manufacturing, where there is a saturation of product and process technologies, the ability of a firm to compete with fellow incumbents frequently lies in its capability to distinguish itself through innovative marketing initiatives [25]. These include among examples, the provision of product differentiation for customers (variety and grades), competing on product attributes other than the basic ones, building customer loyalty (e.g. good delivery services) and brand sensitivity etc. However, these marketing initiatives to create new business opportunities, which translate into business rules in the company, often complicate the downstream activities of PPS and inventories management. For example, vehicle manufacturers employing *mass customization* to provide product variety creates conflicts in the manufacturing system that has been optimized for high conformance, low cost and low variety [3]. Logistics systems thus becomes increasingly complex and existing models and planning methods will need to be continually modified to adapt to new business rules.

All this leads to the surge of interest and a growing market for optimization. Manufacturers are turning to vendors of 'advanced planning systems' (APS) that promise to provide optimization of their supply chain components [125]. Optimization is regarded as: "*the technology in a supply chain management system that can have the single greatest impact on reducing costs, improving product margins, lowering inventories and increasing manufacturing throughput...planning and scheduling modules that depend on optimization technology have generated 30 to 300 % ROI (return on investment) within companies that have already used them*" [92]. Generally, companies are looking for planning solutions that consider major supply constraints, in contrast to traditional MRP solutions which do not consider supply (especially materials) constraints and frequently generate unrealistic supply plans [99]. In a similar spirit our work will also focus on PPS optimization models that acknowledge supply limitations as hard constraints.

1.2 The Case of Hard-Disk Drives

In recent years, the HDD industry suffers a persisting trend of declining profits as the prices per megabyte continue to fall[25]. HDD manufacturers compete in a highly commoditized industry and face tremendous cost pressures. In many cases manufacturing has achieved high levels of efficiencies, and there is often little room for reducing costs further by improving manufacturing. One area that is still worthy of exploration is the design of the HDD itself as a cost-savings measure.

Modular product design[131] has recently received much renewed interest both

among managers and academics [10, 11, 119, 120], as it presents opportunities in reducing the time to develop new products[119] and helps in better manufacturing and vendor relations[119]. The modular design approach enables designers to focus on components and subsystems of the product, rather than on the interactions between the components and the product itself. A product with a high degree of modularization is defined as one in which the majority of components are independently or *loosely coupled*[46]. This implies that component substitutions can be made without major changes to the product design itself. Modular designs provide several advantages, among which includes: 1) the ability to market a large variety of products, resulting from different combinations of the components, 2) shorter times-to-market of products, 3) the ability to implement rapid incremental technological improvements, as newly upgraded products can be introduced to the market as soon as the new component technology is available, and 4) lower costs of design, production, manufacture and distribution[43].

Most product designs tend towards modular systems as its technology matures. As understanding of the product and its components increase, it is possible to define the necessary interfaces so that a component's design could be independent of the product design. A good example of this is the automobile industry [10]. The HDD is basically an assembly of a number of *critical* components, and can essentially be considered to be a modular design too. This has enabled many technological innovations to be incorporated in the HDD over the years. For example, the disc platter component of the disk drive was made of aluminum in the earlier days. In

1992, IBM introduced glass platters, which are more reliable, smoother, can hold more data, and can spin faster resulting in faster access time and data transfer rates. Because of the modular design of HDDs, this technological innovation can be easily incorporated into new HDDs that were marketed. Balachandra [12] provides a detail discussion on the modular design of HDDs.

1.3 PPS Problems in Hard-Disk Drive Assembly

1.3.1 Build-pack PPS Problems

As mentioned in the preceding section, one of the advantages of adopting modular product designs is the potential of achieving lower production and manufacturing costs. However, it is clear that these advantages can only be exploited if there is a proper design of the corresponding production, manufacturing and distribution planning systems to aid decision-makers. There is an abundance of academic research devoted to the study of various components of the planning systems. In this work we study a new class of production planning problems of emerging importance based on HDD assembly. The characteristics of this class of problems were motivated by an actual HDD manufacturer, whose customers are largely original equipment manufacturers (OEMs) and reputable PC-makers. The HDD manufacturer purchases all the critical components from multiple vendors on a long-term contract basis. It then assembles, tests and packs the drives for its customers. The manufacturer adopts the modular design approach of HDDs as a cost-savings

measure as mentioned above. Based on the modular design of the HDD, the manufacturer also implements a scheme called the *approved vendor matrix* (AVM). The AVM allows customers to restrict the combinations of pairs of preferred vendors supplying the components in their products. In the HDD industry, products are largely undifferentiated in the eyes of the purchasers [25]. The AVM scheme is thus positioned as a competitive advantage for this manufacturer as it provides its customers the opportunity to participate in defining their products. A *build type* in this work is defined as the set of all HDD that uses the same combination of component vendors. A build type can be *packed* (assigned) for a customer order only if it complies to the AVM specified by the customer. In general more than one build-type can satisfy the AVM requirements of a customer and vice versa.

As have been mentioned in §1.1, the presence of certain new business requirements like mass customization can create complications in the current practice of PPS. The AVM is in fact such a business requirement. In many cases such as this, manufacturers are realizing that "*the proliferation of product variety and the complexity of the manufacturing environment has exceeded their ability to do production planning on spreadsheets, using the guidelines, rules-of-thumb and experience developed over the years* " [125]. The study of the production planning problems in the face of business requirements of the AVM is thus timely and relevant.

We define production planning problems with AVM requirements as the class of *build-pack PPS problems*. In summary the build-pack PPS problem can be simply stated as follows. Given a fixed set of available capacities, component supplies and

the AVM requirements, develop the build and pack schedules that minimizes the total production costs. These production costs that we aim to minimize are in line with some of the most important supply chain performance measures of the industry. In a white paper by Valdero [135] that discusses supply chain management of high technology firms, the following serious business risks were identified: i) profit lost to excess and obsolete inventory, ii) revenue lost to unexpected fulfillment demands or incorrectly managed allocation, iii) customers lost because of unforeseen shortages or mismanaged expectations, iv) partnership opportunities lost because of inability to deliver on time or in sufficient quantities. The implications of these problems are widespread, as it impacts a company's immediate customers and partners, translate directly to the company's revenue growth and even affect their stock prices and valuations. Similarly, AMR Research's three-tiered hierarchy of supply chain metrics[66] rates *perfect order fulfillment* and supply chain management costs as two key performance metrics in a manufacturing organization. These management level metrics translate to the ground level as the detail metrics of *finished goods inventory*, *order cycle time* and *perfect order details*. These metrics indicate the level of the operational readiness of the company. To reflect these metrics, the production costs in our models thus consists penalty costs for the inability to fulfill customer demands, and the costs of production in excess of demands. The scope of our research focuses on the mathematical modeling and solution development for this class of problems. We consider two main problems in this work: a multi-period production scheduling problem, and a stochastic production planning problem. We

also study various extensions of these two problems.

1.3.2 Reduction of Planning Cycle

From the perspective of the user of an APS, reducing the planning cycle and achieving real-time planning and execution is desirable as it leads to improvements such as reduction of supply chain inventories, increase in responses of the operations and improved customer service. Extensive planning cycles are also undesirable as they result directly in production time lost that were intended to compensate for operational uncertainties [135]. Further, as noted by Manugistics' Heaghney and Noden [59], the push to shorten decision cycle times, especially at the tactical and operational levels of planning, has made consistent and balanced decision-making increasingly difficult.

Up to now, the reduction of planning cycles has been limited by the speed at which an optimized plan can be generated [125]. A key element in APS systems that embed optimization processes is the solver, which solves the planning model for the optimal solution. Many application vendors of APS believe that core competencies can be built on the internally developed solvers or other optimization components. The mathematical model and the solution algorithms are in fact valuable avenues which can directly help in the reduction of the planning cycles. Algorithmic performance, in particular the computational speed of the solution process, is a major concern and motivation of our work.

In the case where there are no AVM requirements, the same PPS problems can

be modeled and solved easily using linear and integer programming techniques. The AVM however, complicate these formulations in a non-trivial manner, and render their solution via general-purpose solvers extremely inefficient for realistic problem sizes. The algorithms we present in this work, on the other hand, are able to outperform general purpose techniques significantly in terms of solution times. This directly contributes to the reduction of the planning cycle of the end-user. Our algorithms also require modest amounts of computational resource, and is appealing to decision makers looking to apply optimization to solve their planning problems, but cannot afford the enormous amounts of computational resources often required by general purpose methods. To the best of our knowledge this work is also the pioneering effort that investigates this class of problems in production planning research.

1.4 Outline of Thesis

The organization of the rest of this thesis is as follows. In Chapter 2, we provide the essential background and motivation of our work. A description of the class of AVM requirements that is central to all our problems is given. We then provide the scenarios and motivations of the two problems that will be studied. These are: (1) the multi-period build-pack scheduling problem, and (2) the build-pack planning problem with stochastic demands. We will also consider various extensions of both problems. In the last section of Chapter 2 a survey of some related literature in production planning research is provided. Chapter 3 presents a formulation and

solution approach for Problem (1) using the column generation method. This lays the foundation for designing the solution algorithms of the rest of the problems considered. In Chapter 4 we provide an alternate formulation and solution method for the same problem (1), using the generation of cut constraints in a multi-stage formulation of the problem. This is essentially a dual approach, in contrast to the primal approach in Chapter 3. In Chapter 5, an extension of Problem (1) is considered, in particular when the number of setups are limited. The formulation presented in Chapter 3 is modified to account for this, and we then provide some simple heuristics based on linear programming (LP) rounding to generate good quality solutions using only modest amounts of computation time.

In Chapter 6 we turn to the formulation and solution method of problem (2), where the customer demands are assumed to be random. The formulation we use is essentially a set-partitioning type model with side constraints for the components supplies limitations. A column generation method is developed to solve the linear relaxation and approximation of the problem, and a branch-and-price method is used to achieve the optimal solution. Lastly in the chapter some computational results from our implementation are presented. Chapter 7 considers some special and realistic extensions to problem (2), namely when there are homogeneous lots requirements, and when demands follow arbitrary discrete distributions. We propose some modifications to adapt the solution framework for Problem (2) for these extensions. Finally Chapter 8 concludes our work, and throws open some possible challenges for future research.

Chapter 2

Background

This chapter presents the essential background information of our work. In the following section, we first describe, using examples in the HDD context, the class of AVM requirements that are central to our problem models. §2.2 provides the scenarios and motivations of the two problems we consider in this work, i.e. the multi-period build-pack scheduling problem, and the build-pack planning problem with stochastic demands. As their names suggest, the first problem concerns itself with planning in smaller time-buckets, whereas the second problem is concerned with tactical planning over a longer time horizon. Only the basic versions of the two problems are presented here. Extensions of the problems will be described later in chapters which consider them. In §2.3 we survey the ideas of *mass customization* and related models in common component problems to draw comparisons to our problem. Finally in §2.4 a survey of related PPS research is presented for the purpose of positioning our work in the scheme of things.

2.1 Approved Vendor Matrices

As a competitive advantage, the HDD manufacturer allows its customers to choose component vendors for their products using the AVM. As have been mentioned in §1.2, the AVM is a scheme that offers product variety to its customers. This is possible because the HDD can be regarded as a highly modular product (see §1.1). The HDD is essentially an assembly of a number of critical components including, for example: the headstack assembly (HSA) which mounts the read/write head, the disc platter(s), the printed circuit board(PCB) that mounts the microprocessor, the spindle motor, the bearings and the case and cover. Because of the high degree of modularity in HDDs, the majority of the product components are regarded as independent. For example, at the current stage in the life cycle of HDDs, upgrading the spindle motor does not influence the performance of the drive other than itself, since its performance does not interact with the other components. On the other hand, the performance of HDDs is well-known in magnetic recording technology to be highly sensitive upon the interaction between the HSA and disc components. In particular, the choice of the coating on the disc platter influences the performance of the read/write head. Additionally, the choice of the head may also affect the firmware (microprocessor) that controls the read/write operations. Some customers such as OEMs often have their own engineering evaluations on the performances of various combinations of the HSA and disc components. To account for such interactions the AVM allows the customer to specify combinations of vendors for *pairs* of components. Tables 2.1 and 2.2 shows a typical AVM of a customer.

H_1-H_3, D_1-D_3 and P_1-P_3 here denote the different suppliers of the HSA, disc and PCB components respectively. In Table 2.1 a value of zero (one) is assigned to combinations of HSA and disc vendors that cannot (can) be used to build the HDD for the customer. Similarly in Table the customer specifies a value of zero (one) to combinations of HSA and PCB vendors that cannot (can) be used. All build types that does not violate the specifications of Tables 2.1 and 2.2 can be assigned towards this customer. For example, the build type comprising of components H_1, D_2 and P_2 can be used to fulfill the demand of the customer, while H_1, D_1 and P_2 is not allowed to.

Table 2.1: AVM For Head-Disc Combination For a Customer

DISC/HSA	H_1	H_2	H_3
D_1	0	1	1
D_2	1	1	0
D_3	0	1	1

Table 2.2: AVM For Head-PCB Combination For a Customer

PCB/HSA	H_1	H_2	H_3
P_1	1	0	1
P_2	1	0	0
P_3	1	0	1

2.2 Problem Descriptions

In this section we describe the basic scenarios of the two problems considered in this work. The problems were adapted from the production planning and scheduling environment of the HDD manufacturer. As have been mentioned this manufacturer performs the final assembly and testing of the disk-drives for the customers, with the components supplied by multiple vendors on a long term basis. The problem descriptions that follow are based on a technical documentation[101] of the detailed process flows of the production planning operations of the company. The document was developed by the author and verified with the company during a period of under-study with the company.

2.2.1 Multi-Period Build-Pack Scheduling

The problem scenario starts with the release of the Master Production Schedule (MPS), which is a schedule of order types (by demand quantity and due date) to be fulfilled in the current week. However, to be implemented at the shop floor level, the master schedule needs to be broken down into even more detailed schedules. A *build schedule* schedules the run quantities of build types in each production period, while a *pack schedule* assigns the build types towards the fulfillment of customer orders in the MPS.

Once the build and pack schedules are drawn, the rest of the production process is relatively straightforward. At the beginning of each production period, production supervisors refer to the build plans to draw components from the parts store,

and these components are fed into the manufacturing cells to be assembled into the specified disk-drive types. These are then passed into the test cells for software coding and power-up tests. Finally, the drives are labeled and packed for the customers as specified in the pack schedule and these finished goods are shipped out of the factory everyday to regional distribution centers (D.Cs).

In the current system, a team of human planners manually draft the build and pack plans using the MPS as reference. When production volume and finished products proliferation becomes high, it becomes increasingly difficult and time-consuming for the planners to co-ordinate and draft feasible schedules that makes the best use of the common manufacturing resources. In this work, our prescriptive scheduling model takes the MPS and translates it into optimal build and pack production plans. We consider in our problem the limited availability of both manpower capacity and components availability. In the company, it is not uncommon that in the course of production planning, although the manufacturing resources meets the requirements to fulfill the MPS in an aggregate sense, daily availability of resources may not be fully synchronized with the build plans, and cannot be changed in the short term, hence causing production 'misses'or so-called *underpacks*. These under-packs are costly as they contribute to the direct failure to fulfill committed delivery to customers on time. An underpack of an order is the number of units short of the demanded quantity that is due. Underpacks are accumulated into the next production period as backlogs, and penalties are charged towards the backlog. If they cannot be fulfilled by the end of the planning horizon

then they are penalized as shortages, since backlogging of demand into the next work week indicates failure to fulfill the total demand bucket in the current week and is considered as poor operational performance on the company's part. Our objective is to schedule daily production in a manner so as to minimize the total daily production backlogs and shortages within the planning horizon.

2.2.2 Build-Pack Planning With Stochastic Demands

For this problem the description is as follows. The current practice of production planning can be seen to consist of two main phases. In the first phase called *build-planning*, the manager determines the total build-type levels to run in the entire weekly demand bucket, subjected to limited availability of the component supplies. Due to the volatility of the electronics goods market, the build-plan must be determined prior to full knowledge of the customers' future demand. In the current practice, a simple product-mix linear program is used to generate the build-plan. The unknown customer demand is estimated using a point forecast, and the AVM restrictions are ignored. In the second phase called *pack-planning*, which occurs after demand realization, production planners assign the build-types to fulfill these demands using spreadsheets, observing the AVM requirements of the customers. It should be noted that in actual operation, the build-plan is not used rigidly as a decree to drive detailed scheduling, but rather as a tool for management to accomplish several other important purposes, including: 1) to estimate the capability of customer demand fulfillment with the current components supplies over the larger

time bucket, 2) to drive capacity requirements, 3) to negotiate for component supply changes, and 4) to serve as a guideline for short-term planning.

In this work we are concerned with the development of a more rigorous approach to the build-planning phase. Although the current practice of using the product-mix LP is simple and requires little computational effort, the solutions generated may be quite imprecise, since it only uses a point estimate for the demand and does not take the AVM restrictions into account. Such an approach may be justifiable in the past due to limited computational resource, but with the current availability of high-speed processors readily at disposal, it seems motivating to devise more realistic planning models which are capable of providing more precise estimates. In particular it is desired that the new planning model takes into account the variability of the demands, and also to respect the AVM restrictions. To define this planning model we first state the *build-pack planning problem* as follows: given some limited information of the future customer demands (i.e. for our modeling purposes some fitted probability distributions of the demands), determine the set of build-plans prior to demand realization that minimizes the total shortage costs for unfulfilled demand and holding costs for excess production on expectation, subject to limited components availability and the AVM restrictions.

2.3 Mass Customization Literature

With increasing demand for product variety and customization, and shortening of product life cycles, companies face tremendous cost pressures and are forced

to revisit their operations strategy. While the Marketing literature[7, 33] indicates that broader specialized product lines lead to higher market share, the Operations Management literature predicts that cost and complexity may increase with greater variety. Manufacturers have adopted various strategies to reduce costs and improve customer satisfaction. *Mass customization* (MC) is one such competitive strategy that has become a major objective of many Fortune 500 companies. MC refers to a process of production of goods and services tailored to suit the needs of customers in a mass market. Davis[38] promotes MC as: "*the ability to provide individually designed products and services to every customer through high process agility, flexibility and integration.*" A survey by Ablstrom and Westbrook[1] reported several benefits that companies have experienced from using mass customization, including: increased customer satisfaction, increased market shares, increased customer knowledge, reduced response time and manufacturing costs, and increased profit. Identification and the classification of MC is widely varied in practice. In this section we discuss some aspects of MC addressed in the literature, and how the AVM scheme of the HDD manufacturer fits in the framework of MC, and its similarities and differences with other models of MC in practice.

One of the most successful build-to-order (BTO) companies that employs MC was Dell Computers, which gained market share by building customized computers using the Internet as an order fulfillment vehicle. The personal computer system is defined in terms of specifications such as memory size, processor speed, hard disk drive, software and other peripherals. Dell provides a variety of these specifications

for the customers to choose from. The customer selects from the various options for the different aspects of the computer system according to his choice. With successful manufacturing and delivery of the finished products within 5 days of lead time, Dell was able to generate 160 % ROI[48] . Other major manufacturers like Motorola, Hewlett-Packard, General Motors, Ford and Chrysler are also specifically using mass customization processes in their production facilities.

Swaminathan[133] identified five methods or approaches to facilitate mass customization in practice: part standardization, process standardization, product standardization, procurement standardization and partial standardization. The use of standardized parts to serve different product items derives benefits of lower costs due to economies of scale, reduced inventories, and improved forecasts of the component needs. With process standardization, the customization can be delayed as late as possible. With product standardization, companies may advertise a wide variety of products but only stock a few of the items. Downward substitution is then used to produce unstocked items when there is a demand for them. With procurement standardization, companies acquire common equipment and components to carry out their operations, thereby enabling benefits of cost-savings from buying standardized materials and equipment. Lastly, the partial standardization approach offers customers a limited number of options to choose from while keeping their products mostly standardized. Dell Computers uses this approach effectively by allowing its customers to choose a standardized computer system along with selective options for the various categories of the product. Similarly for the HDD

manufacturer, the AVM can be seen a scheme to offer customers component selection options for the standard system, which in this case is the disk drive. The build-types in this sense constitute the product variety that is offered to satisfy certain customer requirements. In general more than one build-type can satisfy a customer's requirements.

The definition of the levels of individualizing a product that characterizes mass customization varies among authors. Gilmore and Pine[51] for example identified four customization levels based mostly on empirical observations: collaborative (designers dialogue with customers), adaptive (standard products can be altered by customers during use), cosmetic (standard products packaged specially for each customer) and transparent (products are adapted to individual needs). In this sense, for the HDD manufacturer, the AVM can be considered to be a customization at the collaborative level. As have been mentioned, the customers of the HDD manufacturer are largely OEMs and reputable PC-makers on a long-term working relationship with the HDD manufacturer. Based on past experience, these customers have developed some technical knowledge on the HDD component performances and thus have their own engineering evaluations and preferences. The AVM in this way allows certain latitude for the customers to participate in the design of the their HDDs, although to end-users such a customization is usually invisible.

To justify the use of MC as a competitive strategy the following factors are commonly emphasized in the literature. An existence of customer demand for variety

and customization[81, 84] , appropriate market conditions[82] , readiness of the value chain[43, 81] , available technology[61, 75] , customizable products[43, 84] and knowledge-sharing[81, 109] . For the HDD manufacturer, the demand for customization apparently exists, although it was not the original intention of the manufacturer to provide variety by multiple vendor purchase of their components. Customers are becoming more knowledgeable about their HDDs and the various performances of the components' interfaces and prohibits combinations of interfaces which produce inferior quality drives. Being the first to offer such a scheme in the HDD industry, the market conditions for the AVM to be transformed into actual competitive advantage is also appropriate. To improve the readiness of the supply chain, the manufacturer is also working towards closer supplier and customer relationships in both positioning of the physical supply networks closer and establishing an efficient information network. The HDD, as have been mentioned in §1.2, is highly modular in design, and hence is appropriate for implementation of customization.

The successful application of MC like in the case of the Dell Computers also relies strongly on the tight integration of the upstream supplier of parts, the mid-stream manufacturer and assembly of components, and the downstream distributor of finished goods in the supply chain[29] . The problems that are the motivation of our work is the final assembly process of the HDDs. The production planners essentially face a problem of assigning build-types to customer requirements under the AVM restrictions. Because components from a particular vendor can be assigned to more than one customer in general, our problem bears some similarity to the

component commonality problem[9, 47] . This problem arises from assemble-to-order systems where product-specific components are present alongside with parts shared by several products. This is also a specific form of MC when applied to the situation where there are a large variety of products. Simple inventory models of the component commonality problem with either stock-out or service level constraints were considered by works like Baker et al[9] , Collier[32] and Gerchak et al[47] . These models are basically two-stage decision models, where in the first stage the stock levels of the common and product-specific parts are determined prior to demand realization, and in the second stage the sales of the products are determined. The build-pack scheduling problem in our case is similar to the second-stage problem, where the components levels are fixed and the demands are realized. However in our case demands are not specified for individual products, rather groups of products,i.e. build-types that satisfy the AVM requirements. Multi-period extensions were considered by Tayur[132] and Srinivasan et al.[130] using a build-to-level policy. All these models however do not consider the constraints on limited components supply, as is faced by our problem. Further, the issues of handling large problem sizes and solution efficiency have not be addressed.

2.4 A Survey of Production Planning Models

In this section we provide a survey of the existing research literature in production planning. Production planning mathematical models is an extensive area of research and practical application because of their powerful optimization capabil-

ities. This survey is by no means exhaustive, and only aims to introduce some common modeling approaches and frameworks that have been considered.

According Bitran and Hax[23], production planning problems in manufacturing may always be formulated as mixed-integer programs (MIP) or linear programs (LP). However, this approach is often undesirable because firstly, the size of the problem is usually too large, and secondly, this approach does not conform to industrial practice, which requires hierarchical and functional decision units with different responsibilities. In a hierarchical decision procedure, typically a set of problems is solved in a sequential manner, with the planning horizon decreasing and the level of detail increasing as one moves down the hierarchy. The high level decision thus impose constraints on lower level actions, and the lower level decisions providing feedback to the higher levels. By definition, it is clear that a hierarchical approach is suboptimal. In practice, the planning process begins at which output, inventory and manpower are determined in aggregate figures. These figures are then used as inputs for lot sizing, scheduling and resource allocation at the level of individual items. This process implies also that appropriate disaggregation schemes will be required for consistency and feasibility.

In the rest of this section we survey various works in production planning research. We classify the survey under the umbrella terms of *aggregate production planning*, MRP models, *earliness-tardiness* models and stochastic planning models. Note that this categorization is used here only to facilitate the presentation of the material and does not imply a strict division between the categories. The models

and assumptions that has been developed and undertaken by different authors are wide and varied, as different manufacturing systems and practices have emerged and changed over many decades. In the discussion we also point out the similarities and differences in the various modeling assumptions that has been considered in other work and ours.

2.4.1 Aggregate Production Planning Models

Generally, the aggregate production planning (APP) problem concerns itself with the utilization and allocation of production resources to satisfy customer demands at minimum production costs. Typical decisions made are the determination of workforce level, scheduling of overtime, determination of run quantities. In manufacturing, planning and control systems the APP serves as a constraint on the master production schedule (MPS). To justify the use of APP, it is necessary that grouping of product families into an aggregate product is possible. This of course assumes some degree of homogeny in the product families. For example, products sharing similar setups are grouped into a product type. Product types of the same seasonal demand pattern can then be grouped into a product family, and a product type can only belong to one product family. The aggregated families of products are then used as input in conjuction with various APP techniques to ensure that resource and capacities are adequate to meet customer demands.

Many pioneering works[67, 72, 107] since the 1950s have used MIP or LP models for the APP problem. Various techniques that exploit the problem structure are

applied to solve the problems efficiently, including transportation formulations[22, 26], range programming[85] and separable programming[96]. For most of these models, the production costs consists of linear or piece-wise linear representations of compromises between inventory costs and overtime costs. Very few models allow backlogging of orders, with exceptions such as Posner and Szwarc[111] and Singhal and Adlakha[128].

Besides the LP or MIP approach, the linear decision rule[68, 69] (LDR) method was also one of the early approaches developed to deal with non-deterministic demands in APP. LDR relies on linear rules to set the workforce size, production rates and inventory levels. The total expected costs is quadratic as opposed to (piece-wise) linear in functional form. Basic calculus approach is used to obtain the optimal solutions. The clear drawback of this method is the inability to deal with integer-valued variables or constraints.

To characterize *batch processing* manufacturing systems in contrast to continuous assembly line systems, *lot size models* have been developed and explored by several different authors. The central problem considers the trade-off between lost productivity from frequent set-ups and short runs and higher inventory costs arising from longer runs. There are two main lines of development in lot-sizing research: the capacitated lot sizing model, pioneered by Manne[93] and uncapacitated lot sizing derived from the work of Wagner and Whitin[138]. In the former, production items compete for limited capacity resource, and set-up costs become an important element to be minimized. Most works in this direction also consider also the planning

for multiple items, using MIP formulations. The common solution approaches used are decomposition[37] , lagrangean relaxation[41, 86] , branch-and-bound[73] or heuristic decision rules[139]. On the other hand, works extending Wagner and Whitin's algorithm[138], for example Baker[9] and Kao[77], usually approach the problem using dynamic programming methods. A major challenge in lot-sizing decision models is the computational inefficiency in solving realistically-sized problems.

Other approaches in the area APP problems include goal programming (GP), which is first introduced by Lee and Moore[88]. The basic idea is to incorporate managerial objectives as constraints in the model. The managerial objectives are of different priorities, and the solution procedure that follows is iterative in nature. Highest priority goals are first achieved, then the next and so on. As higher priority goals are achieved, the feasible space for the remaining goals is reduced, until subsequent solutions become infeasible. Some extensions of APP problems considered using the goal programming approach include Deckro and Hebert[39], Lockett and Muhlemann[90], and Rakes et al.[112].

Several heuristic approaches for the APP have also been developed over time by different authors. The search decision rule approach combines simulation with standard neighborhood search techniques to gain local optimality. An example is the *parametric production planning*[74] method, where two decision rules addressing work force and production levels are assumed to exist. The forms of the rules are suggested based on several experiments, and the parameters of the decision rules are optimized using search techniques. Taubert's[134] approach combines a

branch-and-bound procedure with search techniques. Another popular heuristic considered is the *production switching heuristic*[97] (PSH), where the purpose is to avoid frequent rescheduling of production and work force size over the planning horizon. A small number of discrete levels of production and work force rules are hence used, with switching of one level to another depending on the inventory and demand levels. Several works based on the PSH has been developed, for example, Oliff and Burch[105] used the PSH for a fiberglass manufacturer, Hall[56] developed a graphical procedure for deciding when to change the production level, and O'Grady and Byrne[104] combined PSH with the LDR method.

In many aggregate models[86, 93, 103] little attention was paid to assure consistency between the aggregate and detailed production planning levels. In practice, aggregation would be of little value if it was not possible to disaggregate back to the detailed level to obtain good solutions. The coordination between aggregate and disaggregate planning is thus another area of focus of many other works in APP. For example, Bitran and Hax[23], Graves[52] and Hax and Meal [58] pioneered the widely accepted hierarchical production planning (HPP) framework. The basic idea is to couple an LP-type aggregate model and a relatively simple jobshop simulation at the detailed level. A common strategy is then to apply lagrangean decomposition[52] to de-couple the two levels and solve the problem in an iterative manner. Zipkin[145] examines the effect of bounding the aggregating variables in LPs, and Rogers et al[117] analyzed aggregation and disaggregation in optimization models. Axsater[5, 6] discussed the conditions for 'perfect' aggregation and includes

multi-stage production. 'Perfect aggregation' refers to an aggregation scheme that ensures consistency between the aggregate and detailed models for all possible sets of production levels. Aggregate feasibility can result in infeasibilities at the detail level if more components are required than available, or if some production capacity constraints are violated. Ritzman et al[115] present an extensive collection of papers on aggregation and disaggregation in manufacturing and service systems.

As mentioned, to use APP, it is necessary that grouping of product families into an aggregate product is possible. This is the basis of aggregation and hence reduction of the problem size. In our problem the set of products using the same component vendors are grouped as a build type, where each build type can typically be used to fulfill more than one customer's demand. On the other hand, each customer's demand can in general be fulfilled by more than a single build type. If there is no AVM restrictions, then we can treat customer orders as the aggregate products, and build-types as the disaggregated products. In fact, the exact composition of the build-types is irrelevant in this case, and only the solution to the aggregated problem is required. The AVM restrictions however, necessitate the identification of the build-types serving the customer orders in the solution. This makes the definition of aggregated products and their corresponding disaggregation schemes in APP models unclear in our problem. In the next sub-section we shall discuss another class of models, i.e. MRP models. These models arose from the study of discrete parts manufacturing systems where the product structures can be quite complex in general, with sharing of components among different products.

2.4.2 MRP Models

MRP and MRPII (manufacturing resource planning) systems are generally regarded as accepted tools of decision support systems (DSS) among production management and control for the management of complex high-volume production with hundreds of products from hundreds of individual components. The conventional MRP system approach can be stated as follows. "*A materials requirements system consists of a set of logically related procedures, decision rules, and records designed to translate a master production schedule into time-phased net requirements for each component inventory item needed to implement this schedule. A material requirements planning system replans net requirements as a result of changes in wither the master production schedule, or inventory status, or product composition.*"[106].

The basic form of MRP does not consider capacity constraints. That is, they perform 'infinite loading', and clearly this approach often does not produce feasible production plans in practice. To circumvent this the MRPII system[17] was developed to incorporate a 'rough-cut' capacity planning phase prior to the bill-of-materials explosion. This serves to reveal the productions that are causing the capacity overloading. Management then takes action, typically by modifying the master production schedule to rectify the capacity issues. In theory MRP and MRPII attempt to produce a feasible production plan to meet external demands. In practice, there are several complicating issues that make the successful application of MRPII systems difficult. The problems associated with lead-times, lot

size requirements and capacity constraints are inter-related and difficult to resolve by human judgment. This has motivated much research interest to study MRP systems using mathematical models in the area of *multi-stage discrete parts manufacturing*.

Works in multi-stage production planning scheduling is abundant and varies in the class of systems studied. For a single-product *series system*, Love[91] and Zangwill[144] developed efficient solution techniques when there are no capacity constraints. In the case of capacity constraints, Lambrecht and VanderEecken[83] and Ramsay[113] present heuristic approaches to the lot-sizing. For *parallel systems*, Gabbay[44] studied the case where each level is constrained. Zahorik et al[143] presented heuristics for an n-period problem using a network formulation. For *assembly systems*, each product is composed of a number of *predecessor* sub-assemblies, and each sub-assembly will have only one *successor*. Examples of lot-sizing research for this system are Crowston et al[34] and Veinott[137] . Finally, in the general case when components can be shared for different sucessor products, i.e. components commonality, the system is that of the general form of the MRP problem.

Lot-sizing problems for this general case have been studied by several authors[18, 19]. For capacity planning, the Collier[32] finds that certain lot-sizing techniques can lead to erratic capacity usage. Billington et al[21] on the other hand pointed out that batching may either help or hurt the capacity usage pattern. Further, works such as Caie and Maxwell[28] and Maxwell and Muckstadt[95] developed heuristic methods for the problem of sequence dependent setups in the multi-stage production setting.

Billington et al[20] developed a MIP formulation to simultaneously determine lot-sizes, lead-times and capacity utilization plans in a capacity-constrained multi-level MRP system. The authors then attempt to reduce the problem size by ignoring production facilities which are not supposed to be bottlenecks. Sergerstedt[123] reformulated the problem for a dynamic programming solution procedure. The formulation was also extended to account for scheduling constraints. Hoover and Perry[70] discussed the use of simulation models in conjuction with LP models in complex production situations. Byrne and Bakir[27] adopted this hybrid simulation-analytical approach and showed that solution from the classical LP planning model may be infeasible for real production systems due to non-linear behavior of the workloads at the machines. They proposed adjustments of the capacity constraints based on simulation results to obtain more realistic capacity constraints. Many other works have also appeared in the literature dealing with various aspects of MRP and MPS systems. These include studies on the effects of lead times on backlogs and finished components inventories[76], efficient adaptations of the simplex method by exploiting the property of triangularized basis of linear programs of certain MRP systems[63], linear programming techniques for plan scheduling considering various system characteristics like finite capacity and realistic lead time[45].

As have been mentioned, research in MRP-related problems have focused largely on capacity-planning, lot-sizing and lead-time issues. Because of the complexities of these issues in the multi-stage structure, supply constraints, in particular, the case where the components availability is fixed and cannot be changed in the short

term, are rarely considered. Further, it is not the character of the MRP approach to deal with such situations, since the MRP idea is to develop a production plans for the components rather than to treat the components as fixed resources. However, such constraints reflect the nature of the build-pack scheduling problem, where supply quantities from the vendors are already in place and cannot be changed. The production planners task is then to work around these supplies to decide what build-types to schedule for production and how to assign them to the customer orders.

In discussing the mathematical programming approach to master production scheduling, Chu[30], addresses the issues of having limited 'non-perishable' production resources, i.e. component supplies, and a possibility of the presence of a 'vendor-approved list' for components, which may increase the problem size enormously. The differences are that our problem has a more complicated form of vendor-approved list to be observed, and attempts to minimize backlogs and shortages, while in Chu[30] a linear profit associated with production is maximized. The model presented determines production quantities for individual products under the restrictions of limited parts supplies and production capacities in order to maximize total profit. The results of the model can be interpreted as the resulting MPS and MRP. Also, while Chu[30] suggest the use of an interesting myopic decomposition in the temporal dimension to attack the size of the problem, it only gives feasible-optimal solutions when capacity is slack. In our work we tackle the problem size by exploiting the characteristics of the AVM. Further, while we do

not consider set-up costs and hence the issue of lot-sizing in our basic model, we shall extend our model (Chapter 5) to consider the case where there is a constraint on the number of setups performed in each period.

2.4.3 Earliness-Tardiness Planning Models

The majority of the production planning models discussed thus far are used to minimize total production costs or to maximize product output, and due dates if considered are simply taken as constraints of production planning. Earliness/tardiness (E/T) production planning problems, on the other hand, recognizes on the outset that available capacity cannot meet customer requirements at all times, and there are always capacity shortages in some periods and surplus in other periods. Trying to meet the due-dates as precisely as possible thus becomes the goal of E/T planning problems. E/T production planning arose from the surge of research interest in trying to use the highly popular just-in-time (JIT) manufacturing philosophy to improve the production planning approach of MRP. Work in machine scheduling to minimize E/T penalties is not new (Ghosh[49, 50] , Hall[54, 55]), but such research is much focused on single-machine and parallel multi-machine scheduling problems. Also, the processing capacity in such problems are assumed to be constant, whereas in a production planning problem for the whole manufacturing facility across a longer planning horizon, the aggregated capacity cannot be taken as constants, but varies from period to period due to maintenence and renewal of facilities. Wang[140] presents two basic models for E/T production planning, one

in a mass or repetitive manufacturing system via a LP formulation and the other for a one-of-a-kind product manufacturing via a MIP formulation. Li[89] and Hao[57] further discusses techniques to ease the computational burden of solving the E/T planning problem. Wang[141] models the E/T planning problem whereby customer demands comes with associated 'due-windows' rather than due-dates. In this case, the scenario is such that the production for a customer demand carried out within its due-window will not incur any E/T penalties. Our problem is closest thus far to the repetitive manufacturing model in Wang[141] , and we may regard the customer-specified due-date in our case to be a special case of 'due-window', i.e. from the beginning of the planning horizon up to the specified due-date. Production occuring after the due-window is penalized as backlog, or a tardy production. The abovementioned models[57, , 89, 140, 141] however does not consider 'non-perishable' resources like components supplies. Also, the techniques to reduce problem size introduced in these works do well in the case when there are only a few end products. Our problem, on the other hand, needs to consider the case of a high proliferation of end products, with a customer-specified AVM to be observed. To model the multi-period deterministic build-pack scheduling problem we adopt the framework of the E/T formulation. Our solution approach however will be based on a column generation procedure.

2.4.4 Stochastic Planning Models

To account for stochastic demands in the planning problems, a general approach commonly used by the mathematical programming community is the scenario-based *stochastic linear-programming* (SLP) approach. This is basically an extension of the deterministic models to permit production planning and scheduling uncertainties to be explicitly modeled and evaluated. These models consider simultaneously multiple scenarios of an uncertain future. Optimal contingency plans for each scenario are computed along with here-and-now strategies that optimally hedge against these plans.

SLP has been widely applied in production planning research. Examples include product mix planning when the requirements are stochastic[36, 116], multi-period production planning[108] and manpower planning[78] etc. Bitran et al[24] presented a MIP model for production planning of style goods with high set-up costs and forecast revisions. Hiller[60] generalized the deterministic multi-stage models (see §2.4.2) to a stochastic programming model with recourse for the case when demand for finished goods is uncertain. The model rationalizes the computations of safety stocks at all stages in a multi-stage environment taking in account of capacity constraints that limit the buildup of such stocks. Beale et al[15] reported on computational experiments with a similar class of models. Kira et al[80] extended Bitran and Hax's[23] hierarchical production planning model to include random demand variables. Swaminathan and Tayur[132] proposed a two-stage integer program that models the problem of designing *vanilla boxes*, or semi-finished products in the

delayed product differentiation approach of managing product variety. The first stage of decisions involve deciding the production levels of the vanilla boxes prior to demand realization. In the second stage the demand is realized, and final customization and assignment is made to fulfill the customer demands. The number of possible vanilla-boxes increases exponentially with the number of components due to the possible combinations of the components. In these ways, this problem is similar to our build-pack planning problem with stochastic demands (§2.3). However, the vanilla box model does not consider components supply constraints, while in our work the supply constraints are of major importance. Also the SLP approach required full enumeration of the vanilla box configurations in the first stage, which results in a large number of variables and constraints in the formulation. This renders the solution algorithm unsuitable for large problem sizes.

In SLP future uncertainty is modeled as a finite set of possible outcomes or *scenarios*, each with an associated probability of occurrence. The objective is typically to minimize some total costs of production on expectation over all the possible outcomes. In this manner the problem can be formulated as a large-scale linear program or *deterministic equivalent program*[142] , and solution approaches like the *L-shaped method*[129] have been developed to solve such formulations. Such an approach, however, is known to be computationally efficient only when the number of possible outcomes(the sample space) are of limited size. Furthermore, the application of the SLP modeling approach presents an additional difficulty in our case due to the high build-type proliferation in our problem. We will further elaborate this

point in Chapter 6 when we consider modeling demands with arbitrary discrete distributions in our problem.

On the other hand, works like Metters[98] , which considers production planning with stochastic seasonal demand, and Hodges and Moore[65] , which considers stochastic product-mix planning, uses the news-vendor model in classical inventory theory as the basic approach for considering stochastic demands. Similarly, for the build-pack problem with stochastic demands, we will adopt a solution framework based on a multi-item, capacitated news-vendor problem. Additionally, to generate feasible production plans, we will employ the use of a set partitioning formulation that groups the customer orders prior to demand realization. As in the deterministic case column generation is applied to address the high proliferation of build-pack decisions. We then use the branch-and-price method to achieve a good set partition, and to allow new columns to be generated at each branch-and-bound node.

Chapter 3

The Multi-Period Build-Pack Scheduling Problem

In this chapter we focus on the *multi-period build-pack scheduling problem* described in §2.2.1. First a formulation of the problem is presented in §3.1. Our formulation is a linear program with an exponential number of columns. §3.2 then develops the column generation method for solving this formulation. Computational results are presented in §3.3, which compares our algorithms with the CPLEX general-purpose solver. Lastly in §3.4, we provide an alternate modeling viewpoint of the problem, by casting the same problem as a multicommodity network flow model.

3.1 Problem Formulation

We formulate the build-pack planning problem using the basic framework of the E/T planning model for mass repetitive manufacturing systems introduced by

Wang[140] . We denote a build-type as θ, and customer k. A *pack type* is defined as an assignment of a build type θ towards customer k. Thus $s =< k, \theta >$, where s denote the index of a pack type. Note that by definition the build-type θ in two different pack types may be the same in general. In reality, a build type is distinguished into two or more pack types only when it is being packed and labeled for different order types.

Denote S as the set of all feasible (in compliance with the AVM) pack types, and S^k as the set of all feasible pack types dedicated to customer k respectively. Customer demand d_t^k, where $t = 1 \cdots T$, can therefore be satisfied only by the production of pack types $s \in S^k$. The objective of the multi-period build-pack scheduling problem is to minimize the total costs of backlogged orders, with g_t^k denoting the penalty cost per unit of order k backlogged in period t. Let p denote the a generic product component, e.g. the HSA, the disc platter, spindle motor etc in the HDD, and P the set of all parts. Let $v \in V_p$ be a vendor supplying component p, where V_p is the set of all vendors supplying p. Production capacity e.g. manpower available in period t is denoted by c_t, and the components supply from v arriving in period t is denoted by $m_{v,t}$. Further, r_k^p is the bill-of-materials data, i.e. the number of parts p in the product of the customer k. The decision variables are $B_{s,t}$, i.e., the production level of pack type s in period t, $s \in S$, $t = 1, \cdots, T$. The formulation $\mathcal{LP}$ is as follows.

Problem $\mathcal{LP}$:

$$\text{Minimize } Z = \sum_{k \in K} \sum_{t=1}^{T} g_t^k \cdot \left[\sum_{t'=1}^{t} \left(d_{t'}^k - \sum_{s \in S^k} B_{s,t'} \right) \right]^+ \tag{3.1}$$

subject to:

$$\sum_{s \in S} B_{s,t} \leq c_t \qquad t = 1 \cdots T \tag{3.2}$$

$$\sum_{t'=1}^{t} \left(\sum_{k \in K_v} r_k^p \sum_{s \in S^k} B_{s,t} \right) \leq \sum_{t'=1}^{t} m_{v,t'} \qquad \forall\ v \in V_p,\ \forall\ p \in P,\ \ t = 1 \cdots T \tag{3.3}$$

$$\sum_{t=1}^{T} \sum_{s \in S^k} B_{s,t} \leq \sum_{t=1}^{T} d_t^k \qquad \forall\ k \in K \tag{3.4}$$

$$B_{s,t} \geq 0, \quad B_{s,t} \in \Re \qquad s \in S,\ \ t = 1, \cdots, T \tag{3.5}$$

The objective function (3.1) minimizes the total cost of tardy production for customer k due in each period t. Note that in this formulation the cost element g_T^k is the shortage cost per unit of each customer k. Constraints (3.2) are the manpower capacity limitations for each period, and (3.3) are constraints on components availability in each period. Constraints (3.4) disallows 'overpacking' to occur and restricts total production to be less than the total demand over the planning horizon. Constraints (3.5) are nonnegativity conditions on production.

3.2 Solution Procedure

The column generation technique[51,102] is a specialization of the simplex method which proceeds by solving a restricted form of the original problem (called the master problem) by considering only a subset of all the possible columns (variables). Non-basic columns which have the potential to contribute to the objective function

advantageously are generated at the pricing stage by solving a separate *pricing problem*. The potential of a non-basic column is determined by some criterion, usually the reduced cost of the column. The new columns found are augmented to the master problem and re-solved for a new set of dual variables. The procedure iterates between solving the master problem and the pricing problem until no more columns of potential contribution can be found.

Let α_s^k and $\beta_s^{v,p}$ $\forall\ k \in K$, $\forall\ v \in V_p$ and $\forall\ p \in P$ be the set of *indicator parameters* so that

$\alpha_s^k = 1$ if pack type s is being assigned as customer k, 0 otherwise.

$\beta_s^{v,p} = 1$ if pack type s is built using vendor v for component p, $v \in V_p$.
0 otherwise.

Each pack type $s \in S$ is defined by a unique setting of the α_s^k and $\beta_s^{v,p}$ indicator parameters. In particular, $\alpha_s^k\ \forall\ k \in K$ defines one order type assigned, whereas $\beta_s^{v,p}$ $\forall\ v \in V_p$, $\forall\ p \in P$ defines one build type used. In accordance with the definition of s, only one α_s^k value can be set to 1 over all k, and only one value of $\beta_s^{v,p}$ can be set to 1 over all $v \in V_p$ for each part $p \in P$. Further, the build type as defined by $\beta_s^{v,p}$ must comply with the AVM restrictions for the order type defined in α_s^k.

Let R_t be some set of pack types associated with period t, i.e., $R_t \subseteq S$, $t = 1, \cdots, T$. Let G_t^k be 'backlog' variables denoting the total number of units of unfulfilled demand for customer k in period t. G_t^k is defined such that $G_t^k = 0$ in the case where there is no backlogging in period t. The *restricted master program* can then be written as:

Problem $\mathcal{MP}$:

$$\text{Minimize } Z = \sum_{k \in K} \sum_{t=1}^{T} g_t^k \cdot G_t^k \tag{3.6}$$

subject to:

$$\sum_{s \in R_t} B_{s,t} \leq c_t \qquad t = 1 \cdots T \tag{3.7}$$

$$\sum_{t'=1}^{t} \sum_{k \in K_v} r_k^p \cdot \sum_{s \in R_{t'}} \alpha_s^k \cdot \beta_s^{v,p} \cdot B_{s,t'} \leq \sum_{t'=1}^{t} m_{v,t'} \qquad \forall\ v \in V_p,\ \forall\ p \in P,\ t = 1 \cdots T \tag{3.8}$$

$$\sum_{t'=1}^{t} (d_{t'}^k - \sum_{s \in R_{t'}} \alpha_s^k \cdot B_{s,t'}) \leq G_t^k \qquad \forall\ k \in K,\ t = 1 \cdots T \tag{3.9}$$

$$\sum_{t=1}^{T} \sum_{s \in R_{t'}} \alpha_s^k \cdot B_{s,t} \leq \sum_{t=1}^{T} d_t^k \qquad \forall\ k \in K \tag{3.10}$$

$$B_{s,t}, \quad G_t^k \geq 0 \qquad \forall\ k \in K, \forall\ s \in R_t,\ t = 1 \cdots T \tag{3.11}$$

Problem $\mathcal{MP}$ is equivalent to $\mathcal{LP}$ except that now minimization is over subsets $R^t \subseteq S$, $t = 1 \cdots T$ rather than over the entire set S. At each iteration of the procedure after solving $\mathcal{MP}$ to optimality, it is desirable to find new variables to be included into R_t, $t = 1 \cdots T$ that will improve the current solution. For minimization linear problems, non-basic variables with the negative reduced cost satisfy this criterion. If there are no non-basic variables with negative reduced cost then the current solution is optimal for $\mathcal{LP}$ and the procedure ends. The column generation method allows new entering variables to be located without explicit enumeration of all the columns. This is accomplished by formulating the search for new variables as the pricing problem.

In our proposed approach we solve T pricing problems, one for each period, so that solving each pricing problem yields a feasible set of the parameter variables

α^k and $\beta^{v,p}$ $\forall$ $k \in K$, $\forall$ $v \in V_p$, $\forall$ $p \in P$. Let $\Gamma_{k,v,v'}$ denote the AVM for customer k between component vendors v and v' of parts p and p' respectively, where $\Gamma_{k,v,v'} = 1$ if v and v' cannot be used together for customer k, 0 otherwise. The pricing problem for period t is stated as follows. We subsume the subscripts in s for notational convenience.

Problem $\mathcal{RP}_t$

$$\text{Minimize } Z_r = \sum_{k \in K} \left[\left(\sum_{t'=t}^{T} \pi_2(k,t') - \pi_3(k) \right) \cdot \alpha^k \right] - \sum_{p \in P} \sum_{v \in V_p} \sum_{k \in K_v} \left[\left(\sum_{t'=t}^{T} \pi_1(v,t') \right) \cdot r_k^p \cdot \alpha^k \cdot \beta^{v,p} \right] - \pi_0(t) \quad (3.12)$$

subject to:

$$\sum_{k \in K} \alpha^k = 1 \quad (3.13)$$

$$\sum_{v \in V_p^k} \beta^{v,p} = \alpha^k \qquad \forall\ p \in P, \forall\ k \in K \quad (3.14)$$

$$\beta^{v,p} + \beta^{v',p'} \leq 2 - \alpha^k \cdot \Gamma_{k,v,v'} \qquad \forall\ v \in V_p, \forall\ v' \in V_{p'}, \text{where } p,\ p' \in P, p \neq p', \forall\ k \in K \quad (3.15)$$

$$\alpha^k,\ \beta^{v,p} \in \{0,1\} \qquad \forall\ k \in K, \forall\ v \in V_p, \forall\ p \in P \quad (3.16)$$

where $\pi_0(t)$, $\pi_1(v,t)$, $\pi_2(k,t)$ and $\pi_3(k)$ refers to dual variables associated with each constraint in (3.7), (3.8), (3.9) and (3.10) respectively. Note that $\pi_0(t), \pi_1(v,t)$ $\pi_2(k,t)$ and $\pi_3(k)$ are non-positive $\forall$ $k \in K$, $\forall$ $v \in V_p, \forall$ $p \in P, t = 1 \cdots T$.

Z_r in (3.12) is the reduced cost expression for $B_{s,t}$, $t = 1 \cdots T$ and $s \in S$. (3.13) allows s to be assigned to one and only one customer k. Constraints (3.14) imposes

the AVM for individual components and allows only one vendor per component. Constraints (3.15) imposes the AVM restrictions for the components. Constraints (3.16) restricts α^k, $\beta^{v,p}$ to be zero-one binary variables.

Consider the case when the variables α^k are fixed, i.e. $\alpha^{k'} = 1$ for some $k' \in K$, and $\alpha^k = 0\ \forall k \neq k'$. Observing that only the second term in (3.12) involves $\beta^{v,p}$, and since $\pi_1(v,t) \leq 0\ \forall v \in V_p, p \in P$, $\mathcal{RP}_t$ becomes the following problem:

$$\min_{\beta^v} Z_s = \sum_{p \in P} \sum_{v \in V_p^k} \left[\left(\sum_{t'=t}^{T} -\pi_1(v,t') \right) \cdot r_k^p \cdot \beta^{v,p} \right] \tag{3.17}$$

subject to (3.14) to (3.16).

The problem is hence to find a feasible build type to be assigned to a chosen customer k at the minimum cost. In a network representation, this is equivalent to finding the shortest directed path through a set of nodes representing the components. Figure 3.1 shows an example of the network representation in the hard-disk drive problem for an order type. Here there are three 'layers' of *component nodes*, one layer for each component, i.e. the HSA, disc and the PCBA. Each node in a layer represents a component vendor that is acceptable for building the customer's order. For example, the three nodes in the layer of the HSA nodes correspond to the HSA models. An arc links two component nodes in adjacent layers only if they do not violate the AVM requirements. For instance in Figure 3.1 , there is no arc linking the second HSA component node to the second disc component node, indicating that the combination of vendors is not allowed in the AVM.

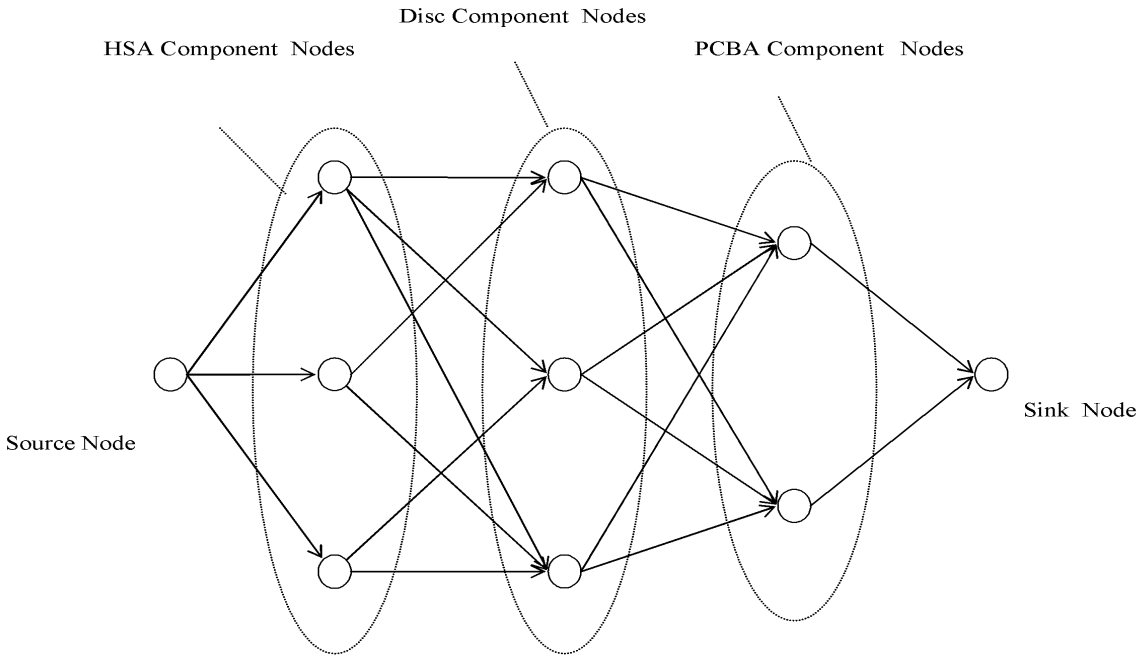

Figure 3.1: Shortest Path Network for Hard-Disk Drive Production Planning

In the network arrangement, any feasible walk starting from the source node to the sink node through the layers of component nodes constitute a feasible build type. The original problem of finding a feasible build type to be assigned to a chosen customer at minimum cost is thus equivalent to searching for the shortest path through the equivalent network of the problem. The 'length' of an arc feeding into a node associated with component model v is then the cost coefficient of $\beta^{v,p}$ in expression (3.17), $v \in V_p^k$.

In the following we define the components of the shortest path network. Let the network of concern be $\mathcal{G}(N, A)$, where N is the node set and A is the set of arcs in the network connecting the component nodes in the special 'layered' structure, $A \subseteq N \times N$. Let $x_{i,j}$ denote the flow leaving node i and entering node j, where

arc $(i, j) \in A$. Define the source node a, and the sink node b. Further, for each vendor $v \in V_p^k$ of each component $p \in P$, define the component node n_v. The structure of $\mathcal{G}$ is such that the nodes n_v, $\forall\ v \in V_p^k$ for each component p forms a layer of nodes, and the layers are arranged in a serial manner in the network. In order to obtain a complete build type, there must be flow from a to b through one and only one node per layer through the series of layers of component nodes. An arc $(n_v, n_{v'})$ (such that $v \in V_p^k$ and $v' \in V_{p'}^k$) exists only if the pair of components p and p' constitute adjacent layers in the network, and such that v and v' does not violate AVM restrictions for k.

The shortest path problem associated with customer k in period t is stated as follows.

Problem $\mathcal{SP}_t^k$

$$\text{Minimise } Z_s = \sum_{p \in P} \sum_{v \in V_p^k} \left[\left(r_k^p \cdot \sum_{t'=t}^{T} -\pi_1(v, t') \right) \cdot \sum_{i:(i,n_v) \in A} x_{i,n_v} \right] \tag{3.18}$$

subject to:

$$\sum_{j:(a,j) \in A} x_{a,j} = 1 \tag{3.19}$$

$$\sum_{i:(i,b) \in A} x_{i,b} = 1 \tag{3.20}$$

$$\sum_{j:(n_v,j) \in A} x_{n_v,j} = \sum_{i:(i,n_v) \in A} x_{i,n_v} \qquad \forall\ v \in V_p^k, \forall\ p \in P \tag{3.21}$$

$$x_{i,j} \in \{0, 1\} \qquad \forall (i, j) \in A \tag{3.22}$$

Constraints (3.19) to (3.21) are network flow balance equations for the source node a, sink node b and each of the components nodes respectively. Constraints (3.22) imposes $x_{i,j}$ to be discrete, but due to the network structure of the problem, total unimodularity guarantees $x_{i,j}$ to be integral in the optimal solution.

Our procedure to solve $\mathcal{RP}_t$ searches through the set of customers. Customer k in period t is 'qualified' as a candidate in the final solution only if the value of $Z_{r'} = \sum_{t'=t}^{T} \pi_2(k,t') - \pi_0(t) - \pi_3(k)$ from (3.12) is negative, since we are only interested in negative reduced-costs (Z_r) and the rest of the terms in (3.12) are non-negative. For each qualified order-type we search for a corresponding build-type which minimizes the value of Z_s in (3.17). This is accomplished by solving the shortest path problem $\mathcal{SP}_t^k$. The resulting pack-type is a valid candidate for the final solution if the sum of the above two terms are negative. Hence there are two factors which decide whether a pack-type $< k, \theta >$ is a valid candidate. The first corresponds to the value of $Z_{r'}$, which indicates the promise of improving the current schedule by packing for customer k. The second factor corresponds to the value of Z_s, which indicates the trade-off to be met if we pack for customer k using build-type θ at the expense of other customers' orders competing for the same components. The decision of whether a pack-type should be considered as a candidate for production is hence determined by the best possible overall promise in schedule improvement. The pack-type with the highest overall promise is then introduced into the production schedule. We summarize the solution procedure for $\mathcal{RP}_t$ below.

Procedure *Price*

1. Initialize the current best solution value $Z_r^* = 0$.

2. Select a $k \in K$ as a candidate. If $\sum_{t'=t}^{T} \pi_2(k,t') - \pi_0(t) - \pi_3(k) \geq 0$ then eliminate the candidature of k and proceed to Step 3. Otherwise go to Step 5.

3. If there are still unselected k from K, go to Step 2. Otherwise proceed to Step 4.

4. If $Z_r^* = 0$ then no entering non-basic variables $B_{s,t}$ $\forall$ $s \in S$ can be located. Otherwise, the current solution to $\mathcal{RP}_t$ is used to form the new column to enter the set R_t. Let the optimal solution be $(k^*, \mathbf{x}^*)$, where $\mathbf{x}$ denotes the vector of $x_{i,j}$. Then $\alpha^k = 1$ if $k = k^*$, 0 otherwise. Also, $\beta^{v,p} = 1$ if $\sum_{i:(i,v)\in A} x_{i,v}^* = 1$, $i \in N$, 0 otherwise. Procedure *Price* terminates.

5. Solve the following shortest path problem $\mathcal{SP}_t^k$ associated with the candidate k and obtain the minimum cost Z_s.

6. Compute $Z_r(k) = Z_s + \sum_{t'=t}^{T} \pi_2(k,t') - \pi_0(t) - \pi_3(k)$. If $Z_r(k) < Z_r^*$ then update $Z_r(k) = Z_r^*$. Go to Step 3.

Once the sets R_t is updated, the master problem $\mathcal{MP}$ is then re-solved and the steps of solving $\mathcal{RP}$ are repeated. The procedure for solving $\mathcal{LP}$ is summarized as follows.

Procedure CG

1. Initialize the column sets R_t, $t = 1, \cdots, T$.

2. Solve the restricted master problem $\mathcal{MP}$ to optimality.

3. Using the dual solution from $\mathcal{MP}$, define the corresponding pricing problems $\mathcal{RP}_t$, $t = 1, \cdots, T$. Solve $\mathcal{RP}_t$ using procedure $Price$.

4. If the solutions to $\mathcal{RP}_t$ gives no non-basic variables with negative reduced costs, then the current solution of $\mathcal{MP}$ is optimal in $\mathcal{LP}$, and the procedure ends. Otherwise proceed to Step 5.

5. Update the column sets R_t, $t = 1, \cdots, T$ with the new columns generated. Go back to Step 2.

Although the above solution approach is developed based on the characteristic AVM of the hard-disk drive problem, we can also accommodate in our work all other types of AVM that have an equivalent network representation with structure as defined in $\mathcal{G}(N, A)$. This implies that for a customer order, there must be no conflicts between the AVM of different component pairs, hence enabling a shortest path representation of the problem.

3.3 Computational Results

In our computational experiments we tested procedure CG using three problem sets. The sizes of the problems used, i.e. number of customer demands, the number

of component vendors, and the number of planning periods, are comparable to the actual problems faced by the production planners of the hard-disk manufacturer each week. The problems were also designed based on the HDD product structure. There are three critical components: the HSA, the disc and the PCBA. Each problem set consists of eight instances with randomly generated demand, capacity and AVM restrictions.

The solution algorithms were all coded in C++. The CPLEX 7.0 LP and network solver libraries were invoked to solve the decomposed sub-problems in the column generation scheme. For comparison purposes the problem instances were also solved entirely using the CPLEX LP solver. The computations were performed on a Pentium IV, 2.4 GHz PC with 512 MB RAM. Tables 3.1, 3.2 and 3.3 highlight the results of our computational study. $|K|$ denote the total number of customers, and $|V|$ denote the number of vendors for each component.

Table 3.1: Problem $\mathcal{LP}$ Set 1: $|K| = 100$ $|V| = 10$ $T = 7$

			Price1			Price2			GLP
Instance	Cols	Rows	CPU/s	Itn	Cols(E)	CPU/s	Itn	Cols(E)	CPU/s
1	567700	913	2.2	19	2235	2.7	11	4468	157.2
2	448700	913	2.8	15	2441	2.4	7	4630	318.8
3	567700	913	2.5	9	2197	2.4	7	4840	182.7
4	343700	913	2.5	18	2401	2.6	12	4875	169.6
5	700700	913	5.6	23	3385	6.3	15	7515	170.7
6	448700	913	3.4	17	2415	2.5	8	4482	293.5
7	252700	913	3.2	16	2475	2.6	10	5002	196.5
8	567700	913	3	24	2489	3.9	14	6199	198.7

Table 3.2: Problem $\mathcal{LP}$ Set 2: $|K| = 200$ $|V| = 10$ $T = 7$

			Price1			Price2			GLP
Instance	Cols	Rows	CPU/s	Itn	Cols(E)	CPU/s	Itn	Cols(E)	CPU/s
1	1401400	1613	11.7	11	4580	12	7	8614	2010
2	1401400	1613	19	10	4419	15.2	7	9191	2150
3	1135400	1613	33.2	15	5130	24.4	9	10268	2985
4	897400	1613	10.8	12	4531	11.8	8	9253	2150
5	1135400	1613	24.8	11	4598	17.2	7	9439	3115
6	897400	1613	16.3	10	4201	10.5	5	7214	3216
7	1135400	1613	18.3	10	4193	13.9	6	8783	2755
8	897400	1613	18	10	4596	12.9	6	7989	2514

Table 3.3: Problem $\mathcal{LP}$ Set 3: $|K| = 200$ $|V| = 20$ $T = 7$

			Price1			Price2			GLP
Instance	Cols	Rows	CPU/s	Itn	Cols(E)	CPU/s	Itn	Cols(E)	CPU/s
1	11201400	1823	41.5	24	6992	42.7	14	18538	NA
2	9073400	1823	48.5	18	6193	59.8	12	16434	NA
3	7169400	1823	30	19	5800	36.7	9	13626	NA
4	2801400	1823	50.8	18	5800	51.2	9	13474	NA
5	1793400	1823	23.8	19	5744	30.4	10	12392	4319.5
6	1009400	1823	49.2	16	5793	48.8	8	12097	1891.4
7	1793400	1823	69.8	21	6579	60.8	7	13799	6747.7
8	1793400	1823	19.9	13	4793	22.1	5	9255	8919.4

Cols and *Rows* in the tables denote the total number of variables and constraints in each problem instance respectively. *CPU* is the computation time in seconds used to solve the instance to optimality. *GLP* corresponds to using the CPLEX general-purpose LP solver to solve the entire problem instance. For procedure *CG* two pricing schemes were implemented. In *Price 1*, only the most promising column found among all sub-problems in $t = 1 \cdots T$ for each k is added to the master problem during each pricing iteration. In *Price 2* the most promising column found from each sub-problem is added. One feasible column per customer is randomly generated in the initialization phase of the algorithm. *Itns* refer to the number of pricing iterations during the solution of problem $\mathcal{LP}$. *Cols(E)* refers to the total number of columns in the master problem at the end of the solution process. *NA* in Table 3.3 indicates that there is insufficient memory space for CPLEX to load the problem.

The computational efficiency of the column generation schemes are evident from the results we obtain. Procedure *CG* uses an average of only 1.6%, 0.65% and 1.1% of the CPU times required by *GLP* for the problems in Tables 3.1, 3.2 and 3.3 respectively. This thus demonstrates the substantial savings in computation time that can be achieved by adopting the column generation solution scheme for our problems. In an actual implementation, our solution algorithm can be embedded in a 'production planning module' in the APS. By incorporating the solution algorithm into the APS, the direct contribution seen by the user is a reduction of the planning cycle, which is of great importance.

The results presented also verify that the column generation algorithm uses only a modest subset of the total number of columns to solve the problems. The total number of variables at the end of the column generation procedure is on average about 0.3% and 0.6% of the total number of variables in the full problem for scheme 1 and 2 respectively for all three problem sets. This amounts to saying that in practice only modest amounts of computational resources like memory or storage space are required for running the module of the APS that solves this production problem. There is even no requirement to store in the database the entire set of possible build-types, since build-types are only generated when the planning module is executed. Hence essentially only the AVM of each customer is required to be stored in the database.

Since *Price 2* generates a larger column pool than *Price 1*, it thus requires more computational effort in solving the larger master problems. On the other

hand it uses less pricing iterations than *Price 1*. This trade-off is observed to be advantageous for most of our test problems in Tables 3.1 and 3.2, with *Price 2* converging faster to the optimal solution. For our largest problems in Table 3.3, the performance of both schemes are almost on par. This indicates that *Price 1* may be more efficient for very large problem sizes as it becomes increasingly difficult to solve the larger master problems in *Price 2*.

3.4 A Multicommodity Network Representation

We now attempt to give more insight into the structure of the build-pack scheduling problem. The pricing problem in the column generation method presented in the previous section has revealed that there are some network structures in the problem, and these that can be exploited for solution efficiency. In this section we will explicitly model the network components of the problem in a new formulation. In particular, we identify our problem as being similar to a class of *multi-commodity network flow* (MCNF) problems. The motivation of this is so that we may then exploit the abundance of existing techniques dedicated to solve MCNF problems.

MCNF problems arise when several items (commodities) share arcs in a capacitated network. They have been studied extensively because of their numerous applications and their intriguing network structure exhibited. Ahuja[2] and Kennington[79] provide comprehensive surveys of the MCNF problem formulations and solution approaches.

In this section we develop a MCNF representation of our problem, where the

objective is to find the minimum cost paths to ship commodities from the origin(s) to the destination(s) nodes through the given capacitated network. Let the network of concern be $G = (N, A)$, where N is the node set consisting of a supply node, a number of demand nodes, *manpower nodes* and *component vendor nodes*. $A \subset N \times N$ is the set of arcs connecting N in a special 'layer' structure to form the required network. The MCNF equivalent of a customer demand k is a commodity to be transported from the supply node to some demand node(s). Figure 3.2 shows an example of the multicommodity network in the hard-disk drive production planning context.

Let o be the supply node. The outflow of each commodity k at o is equal to the total demand for k over the entire planning horizon, i.e. $\sum_{t=1}^{T} d_t^k$.

Let q_t as the demand nodes, $t = 1 \cdots T$. The net requirements of each commodity k at q_t is the demand for k due in t, i.e. d_t^k. We also define a *shortages node* q_{T+1} with zero net requirements. q_{T+1} is not a demand node by definition, and is only used for transfer requirements from o into the demand nodes when shortages occur.

Let h_t, $t = 1 \cdots T$ denote the *manpower nodes*. Commodity k flows from o to h_t when production of k in period t happens. The capacity at node h_t is equivalent to the total manpower or labor resource available in the period t, i.e. c_t.

Let n_v^t denote a *component vendor node*, where $v \in V_p$, $p \in P$ and $t = 1 \cdots T$. The system of the component vendor nodes is structured in the form of layers. Each layer is composed of all nodes for each p. Each commodity must flow through

at least one node per layer to constitute a complete build type. If some level of a commodity k flows through n_v^t, then this level of k is produced using vendor v for component p in period t. The nodal capacity of n_v^t is the net available supply of v_p in period t. Note that these capacities are dynamic since they depend on the flow and capacities on previous periods.

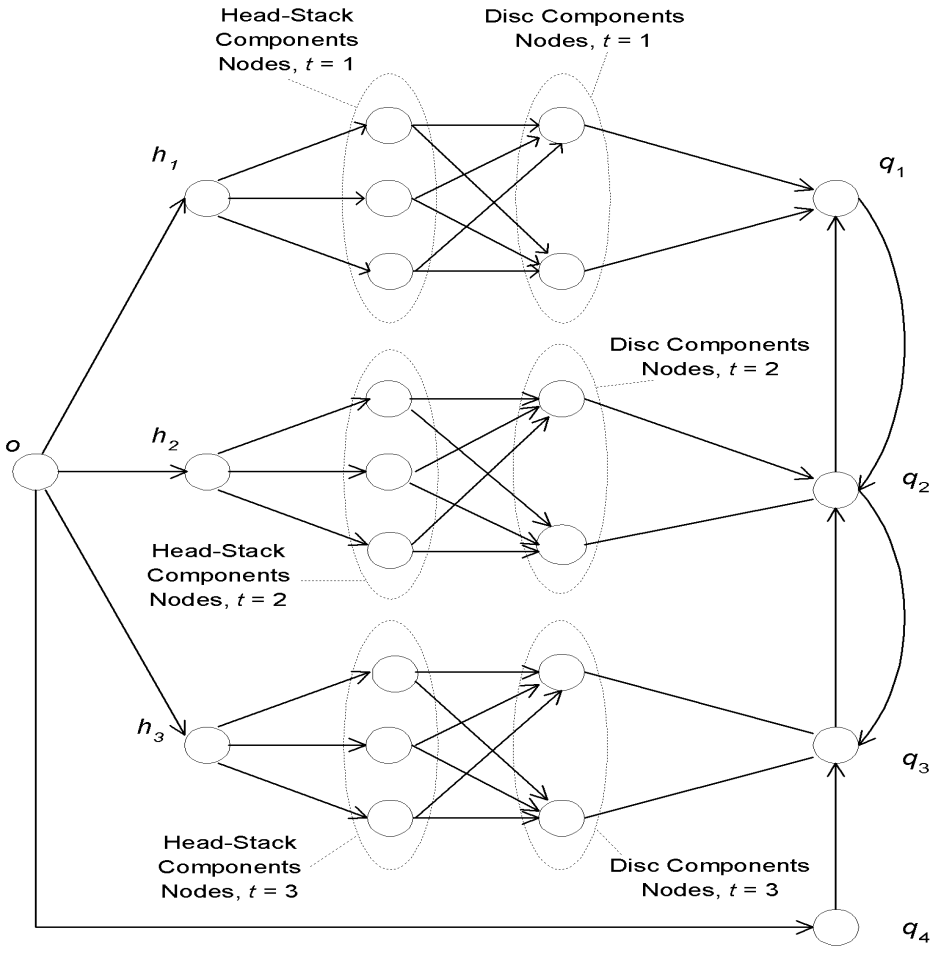

Figure 3.2: Multicommodity Network for Hard-Disk Drive Production Planning

The cost of shipping on arc (q_{t+1}, q_t) per unit of k is the tardiness cost g_t^k. Note that the arc cost is uni-directional on the arc. There is no cost in shipping from q_t to q_{t+1} if we do not consider holding costs.

The MCNF problem is to find a set of feasible flow paths and levels to ship the commodities from o to q_t, $t = 1 \cdots T+1$ so as to Minimize the total arc costs, subjected to the limited node capacities. Define F_t^k as the set of all feasible paths from the origin node o to node q_t, $t = 1 \cdots T$. Similarly F^k is the set of all paths from origin o to all demand nodes q_t, $t = 1 \cdots T$, i.e. $F^k = F_1^k \cup F_2^k \cdots \cup F_T^k$. The path-flow variable X_f, for all paths $f \in F_t^k$, is then equivalent to a scheduled production level of some build type assigned to a customer k in period t. Denote also the path-indicator parameters $\delta_{i,j}^f$ and δ_i^f, so that $\delta_{i,j}^f = 1$ if path f contains arc (i, j), $\delta_{i,j}^f = 0$ otherwise. Similarly $\delta_i^f = 1$ if path f contains node i, $\delta_i^f = 0$ otherwise. In this manner the AVM restrictions can be explictly modeled in the definition of F_t^k, i.e. if customer k does not allow $v \in V_p$ and $v' \in V_{p'}$ to be used together for some pair of components p and p', then $\delta^f_{n_v^t, n_{v'}^t} = 0$ for all the paths $f \in F_t^k$. The equivalent *path-flow* formulation can then be stated as:

Problem $\mathcal{MCF}$:

$$\text{Minimize} Z = \sum_{k \in K} \sum_{t=1}^{T} \sum_{f \in F_t^k} g_t^k \cdot \delta^f_{q_{t+1}, q_t} \cdot X_f \tag{3.23}$$

subject to:

$$\sum_{f \in F_t^k} X_f = d_t^k \qquad \forall\ k \in K,\ \ t = 1 \cdots T \tag{3.24}$$

$$\sum_{k \in K} \sum_{f \in F^k} \delta_{h_t}^f \cdot X_f \leq c_t \qquad t = 1 \cdots T \tag{3.25}$$

$$\sum_{t'=1}^{t} \left(\sum_{k \in K_v} r_k^p \cdot \left(\sum_{f \in F^k} \delta_{n_v^{t'}}^f \cdot X_f \right) \right) \leq \sum_{t'=1}^{t} m_{v,t'} \qquad \forall\ v \in V_p,\ \ p \in P,\ \ t = 1 \cdots T \tag{3.26}$$

$$X_f \in \Re^+ \qquad \forall f \in F^k,\ \forall\ k \in K \tag{3.27}$$

The objective function in (3.23) minimizes the total costs of shipping all the commodities from the origin to destination nodes. Costs are only incurred when commodities ship on arcs (q_{t+1}, q_t) , $t = 1 \cdots T$. Constraints (3.24) are the flow requirements conditions at each demand node q_t. Constraints (3.25) and (3.26) are the manpower capacity the components supply limitations respectively. Constraints (3.27) are uni-directional conditions on the path flows.

Since a MCNF representation exists for the build-pack problem, we can apply techniques used to solve MCNF problems for the build-pack problems. Readers familiar with MCNF models would realize that the path-flow formulation and the path-generation method to solve it is in fact very similar to the column generation formulation and solution approach of $\mathcal{LP}$ in the previous sections. We refer readers to Ahuja[2] for a detailed description of the application of the path-generation and other solution approaches for MCNF problems.

3.5 Concluding Remarks

The multi-period build-pack scheduling problem aims to fulfill customer demands as closely as possible under supply constraints of components availability. Although at the aggregate level total supply can meet total demands, the presence of the AVM restrictions often results in misalignment between the demands at the detail level and the components availability. Substantial planning time is thus used to draft schedules that matches build-type production to customer order fulfillment as closely as possible. We identify this problem as an opportunity to apply optimization technologies to help reduce the planning time and achieve the optimal schedule. In this chapter we have first developed a LP formulation for the multi-period build-pack scheduling problem. The decision variables identify the assignment of the build-types to customer orders. In this manner the AVM restrictions can be modeled explicitly by proper definition of the variables. In the case when there are many components and vendors supplying each component, the possible combinations of build types and the assignments of customer orders can result in a very large number of decision variables. The method of column generation is applied to solve the problem using only a subset of all possible columns. The success of a column generation approach usually hinges on the structure of the pricing problem, i.e. whether the structure allows efficient solution. In our case, we model the pricing problem for each customer order as a shortest path problem. We remark that our network is appropriate for capturing the AVM restrictions considered in our problem. Since HDDs are highly modular by design (see Chapter 1),

we can assume that the interactions between interfaces are limited to pairs of components (e.g. the disc platter and HSA components). 'Higher order' interactions of the interfaces can be assumed to be negligible. The design of the AVM scheme that is offered to the customers is also limited to component-pair restrictions.

Our computational experiments demonstrated that the column generation algorithm is much more efficient compared to solving the problem via general-purpose LP solvers. This has a direct consequence of helping to reducing the planning cycle time and meeting the customer demands in an optimal manner. Only modest amounts of computational resources are required since the entire set of build-type and customer order assignments need not be stored.

Finally, we presented an alternative approach to model the same problem using a MCNF-type formulation. This was developed by combining the network structure of the pricing problem with network components to model flow balance conditions in the multiperiod setting. Doing so opens up the opportunity to exploit the abundance of techniques available in the literature for solving general MCNF problems.

Chapter 4

A Multi-Stage Bender's Decomposition Solution Approach

In this chapter we present an alternative solution approach to the multi-period build-pack scheduling problem. The purpose of this is two-fold. First, from the previous chapter, we have seen that the problem contains inherent network structures, and this can be exploited for solution efficiency. The network aspect of the problem was made explicit by using a MCNF-type formulation at the end of Chapter 3(§3.4). The solution approach presented in this chapter further makes use of the network structure as being composed of 'stages' of networks. It is thus of interest to study the solution performance of a formulation and solution algorithm that is based on this stage-wise network structure. Secondly, the motivation of presenting this solution and modeling approach is practical one, based on the sensible question of whether, in a situation where there are very little AVM restric-

tions, it is possible to avoid a formulation such as $\mathcal{LP}$ which explicitly identifies the build-types. Recall that in the case when there are no AVM restrictions, the problem collapses into a very simple total tardiness planning problem that can be solved efficiently. The solution, which can be viewed as a master schedule of the production levels for each customer order, would then be sufficient, since it would not be necessary to explicitly identify the different build-types. It would hence be of value if this basic result can be made use of in some way when AVM restrictions are present. Some scheme is then required to disaggregate the master schedule solution into build schedules, and then to repair the solution if AVM restrictions are violated. The issue of interest would then be comparing the economy of the effort required to perform the repairing versus the column generation approach.

Our solution approach applies the *outer linearization technique* using *benders decomposition*[16] . A summary of the solution procedure is as follows. We first cast the problem as a multi-stage model, where the first stage is simply a relaxation of the original problem formed by dropping all the AVM requirements. The subsequent stages each consist of a transportation problem involving a pair of components. The solution of each transportation problem accounts for the AVM restrictions and yields a feasible component vendor assignment for the components involved. The transportation problems are solved in a sequential manner, from the first to the last component in the HDD. The solution for each stage sets up the transportation problem in the next stage. If the problem at any stage the becomes infeasible, some AVM restrictions are then violated. A *benders feasibility cut* is

then generated and augmented to the previous stage, which is solved again for a new component vendor assignment. The procedure terminates when all the stages are feasible. In §4.1 the multi-stage formulation is presented. The cut-generating procedure is then developed in §4.2. §4.3 discusses some methods of implementation to help reduce the problem size. Computational results are presented in §4.4 to highlight the differences between the performance of the new solution approach and the column generation approach presented in Chapter 3.

4.1 Multi-stage Formulation

We first present the multi-stage formulation to the Build-Pack problem. To begin we sequence the product components in the order $p^0, p^1, \cdots p^N$, where $N+1 = |P|$, $|P|$ being the cardinality of the set P, and p^0 is the first component constituting the shortest path network presented in §3.2 (or the MCNF network in §3.4), p^1 the second component and so on. This sequence is fixed throughout the solution procedure and determines the order in which the transportation problems are solved.

In the formulation we define the variables $X_{k,t}$ as the production level for customer k in period t. $G_{k,t}$ indicates the backlog level for k in period t. $Y_v^{k,t}$ is the level of component j, vendor v ($v \in V_{p^j}$), that is allocated to the production of k in t. For all components p^j, where $j = 1 \cdots N$, $Y_v^{k,t}$ is defined as

$$Y_v^{k,t} = \frac{r_{p^j}^k}{r_{p^{j-1}}^k} \cdot \sum_{v' \in V_{p^{j-1}}} Y_{v',v}^{k,t} \qquad \forall v \in V_{p^j},\ \ j = 1 \cdots N \tag{4.1}$$

where $Y_{v',v}^{k,t}$ is the disaggregation of $Y_v^{k,t}$ which uses v' for the immediate predecessor component p^{j-1}. Note that (4.1) holds only for p^j for $j \geq 1$ since p^0 does not have a predecessor component by definition. The multi-period build-pack scheduling problem can then be re-written as:

Problem $\mathcal{B}$:

$$\text{Minimize } Z = \sum_{k \in K} \sum_{t=1}^{T} g_{k,t} \cdot G_{k,t} \tag{4.2}$$

subject to:

$$\sum_{k \in K} X_{k,t} \leq c_t \qquad t = 1 \cdots T \tag{4.3}$$

$$\sum_{t'=1}^{t} \sum_{k \in K} r_p^k \cdot X_{k,t'} \leq \sum_{t'=1}^{t} \sum_{v \in V_p} m_{v,t'} \qquad \forall p \in P,\ \ t = 1 \cdots T \tag{4.4}$$

$$G_{k,t} \geq \sum_{t'=1}^{t} d_{k,t} - \sum_{t'=1}^{t} X_{k,t'} \qquad \forall k \in K,\ \ t = 1 \cdots T \tag{4.5}$$

$$\sum_{v \in V_p^0} Y_v^{k,t} = r_{p^0}^k \cdot X_{k,t} \qquad \forall k \in K,\ \ t = 1 \cdots T \tag{4.6}$$

$$\sum_{t'=1}^{t} \sum_{k \in K} Y_v^{k,t'} \leq \sum_{t'=1}^{t} m_{v,t'} \qquad \forall v \in V_{p^0},\ \ t = 1 \cdots T \tag{4.7}$$

$$\sum_{v' \in V_{p^{j-1}}} Y_{v',v}^{k,t} = \frac{r_{p^{j-1}}^k}{r_{p^j}^k} \cdot Y_v^{k,t} \qquad \forall k \in K,\ \ \forall v \in V_{p^j},\ \ j = 1 \cdots N,\ \ t = 1 \cdots T \tag{4.8}$$

$$\sum_{v' \in V_{p^j}} Y_{v,v'}^{k,t} = \frac{r_{p^j}^k}{r_{p^{j-1}}^k} \cdot Y_v^{k,t} \qquad \forall k \in K,\ \forall v \in V_{p^{j-1}},\ j = 1 \cdots N,$$
$$t = 1 \cdots T \tag{4.9}$$

$$\sum_{v' \in V_{p^{j-1}}} \sum_{k \in K} \sum_{t'=1}^{t} Y_{v',v}^{k,t'} \leq \sum_{t'=1}^{t} m_{v,t'} \qquad \forall v \in V_{p^j},\ j = 1 \cdots N,$$
$$t = 1 \cdots T \tag{4.10}$$

$$X_{k,t} \in \Re^+, \quad Y_v^{k,t} \in \Re^+, \quad Y_{v,v'}^{k,t} \in \Re^+ \tag{4.11}$$

The objective function (4.2) minimizes the total penalty for tardiness of the orders. Constraints (4.3) imposes total manpower availability for each period, and (4.4) are the components availability aggregated over all vendors for each component and period. Constraints (4.5) are 'backlog' equations. Constraints (4.6) and (4.7) are the flow-balance and availability conditions for the first component p^0 respectively. Constraints (4.8) follows directly from the definition of $Y_v^{k,t}$ in expression (4.1). Constraints (4.9) together with (4.10) form the flow balance and availability conditions for the rest of the components p^j, $j = 1 \cdots N$.

4.2 Solution Procedure

Our solution procedure decomposes $\mathcal{B}$ into a master problem and a series of sub-problems. Observe that for a fixed set of production levels $X_{k,t}$, (4.6) and (4.7) can be replaced by a set of transportation feasibility conditions, with each $X_{k,t}$ forming a *demand point* and each component supply $m_{v,t}$ $\forall v \in V_{p^0}$ forming a *supply point*. Similarly, for a fixed set of production levels $Y_v^{k,t}$ $\forall v \in V_{p^{j-1}}$, (4.9) and (4.10) can be

replaced by transportation feasibility conditions with each $Y_v^{k,t}$ forming a demand point and $m_{v,t}$ $\forall v \in V_{p^j}$ forming the supply point, $j = 1 \cdots N$. In the following we first formalize the definitions of the decomposed problems.

Define the master problem $\mathcal{RB} : \{Z = \text{Minimize } (4.2), s.t. (4.3) - (4.5)\}$, which is just the basic total-tardiness problem. Note that when there are no AVM restrictions , the solution to $\mathcal{RB}$ forms an optimal production schedule.

The transportation problem associated with (4.6)and (4.7) is used to test for possible AVM violations for the first component p^0. In the following we define the time-expanded transportation 'route' variable $Y_v^{k,t,\tau}$, $\tau = 1 \cdots t$ where τ is interpreted as the arrival period of the supply of v being used. Note that $Y_v^{k,t} = \sum_{\tau=1}^{t} Y_v^{k,t,\tau}$. The Phase-One transportation problem can then be written as:

Problem $\mathcal{TP}_0$:

$$\text{Minimize } Z = \sum_{t=1}^{T} \sum_{\tau=1}^{t} \sum_{k \in K} \sum_{v \in V_{p^0}} \Gamma_{k,v} \cdot Y_v^{k,t,\tau} \tag{4.12}$$

subject to:

$$\sum_{\tau=1}^{t} \sum_{v \in V_{p^0}^k} Y_v^{k,t,\tau} = r_{p^0}^k \cdot X_{k,t} \qquad \forall\ k \in K,\ t = 1 \cdots T \tag{4.13}$$

$$\sum_{\tau=1}^{T} \sum_{v \in V_{p^0}} Y_v^{\tilde{k},\tau} = \sum_{\tau=1}^{T} \sum_{v \in V_{p^0}} m_{v,\tau} - \sum_{t=1}^{T} \sum_{k \in K} r_{p^0}^k \cdot X_{k,t} \tag{4.14}$$

$$\sum_{t=\tau}^{T} \sum_{k \in K} \sum_{v \in V_{p^0}^k} Y_v^{k,t,\tau} = m_{v,\tau} \qquad \forall\ v \in V_{p^0},\ \tau = 1 \cdots T \tag{4.15}$$

$$Y_v^{k,t,\tau} \in S_0 \qquad \forall\ k \in K,\ v \in V_{p^0},\ \tau = 1 \cdots t,\ t = 1 \cdots T \tag{4.16}$$

The objective function (4.12) minimizes the sum of all the shipment levels on inadmissible routes, or model assignment levels $Y_v^{k,t,\tau}$ which are 'illegal', i.e. in conflict with the AVM requirements. Constraints (4.13) and (4.15) follows directly from (4.6) and(4.7), which are simply the demand and supply equations of the transportation problem. Constraint (4.14) captures the supply that is in excess of the demand in the formulation, and is commonly termed as a 'dummy' demand point in unbalanced transportation problems. $Y_v^{\tilde{k},\tau}$ is used to denote the assignment of the supply from v and τ to the dummy demand $\tilde{k}$. Note that (4.4) ensures that there will be no unbalance in the other direction, i.e. demand in excess of supply. $X_{k,t}$ in (4.13) is fixed at the solution levels obtained from $\mathcal{RB}$. (4.16) requires $Y_v^{k,t,\tau}$ to belong in the set S_0, where we define $S_0 = \{Y_v^{k,t,\tau} \geq 0, \text{s.t. } I_0\}$, I_0 being the set of cut constraints in $\mathcal{TP}_0$. We will further discuss cut constraints I_0 later in this section. Note that I_0 is initialised as a null set in the first iteration of the solution procedure.

The transportation problem associated with (4.9) and (4.10) for each component p^j where $j \geq 1$ is used to test for possible AVM violations between component p^j and its immediate predeccessor component p^{j-1}. Define the transportation route variable $Y_{v,v'}^{k,t,\tau}$, where τ is as defined previously, and $Y_{v,v'}^{k,t} = \sum_{\tau=1}^{t} Y_{v,v'}^{k,t,\tau}$. Suppose we have a fixed set of vendor assignment levels $Y_v^{k,t}$, $\forall v \in V_{p^{j-1}}$, $\forall k \in K$ and $t = 1 \cdots T$ for component p^{j-1}. We can then define the Phase-One transportation problem associated component p^j as:

Problem $\mathcal{TP}_j$:

$$\text{Minimize} \quad Z = \sum_{t=1}^{T} \sum_{\tau=1}^{t} \sum_{k \in K} \sum_{v \in V_{p^{j-1}}} \sum_{v' \in V_{p^j}} \Gamma_{k,v,v'} \cdot Y_{v,v'}^{k,t,\tau} \tag{4.17}$$

subject to:

$$\sum_{\tau=1}^{t} \sum_{v' \in V_{p^j}} Y_{v,v'}^{k,t,\tau} = \frac{r_{p^j}^k}{r_{p^{j-1}}^k} \cdot Y_v^{k,t} \qquad \forall\ k \in K,\ \forall\ v \in V_{p^{j-1}},\ t = 1 \cdots T \tag{4.18}$$

$$\sum_{t=\tau}^{T} \sum_{k \in K} \sum_{v \in V_{p^{j-1}}} Y_{v,v'}^{k,t,\tau} = m_{v',\tau} \qquad \forall\ v' \in V_{p^j},\ \tau = 1 \cdots T \tag{4.19}$$

$$\sum_{\tau=1}^{T} \sum_{v \in V_{p^{j-1}}} Y_{v'}^{\tilde{k},\tau} = \sum_{\tau=1}^{T} \sum_{v' \in V_{p^j}} m_{v',\tau} - \sum_{t=1}^{T} \sum_{k \in K} \frac{r_{p^j}^k}{r_{p^{j-1}}^k} \cdot Y_v^{k,t} \tag{4.20}$$

$$Y_{v,v'}^{k,t,\tau} \in S_j \qquad \forall\ k \in K,\ \forall v \in V_{p^{j-1}},\ \forall v' \in V_{p^j},$$

$$\tau = 1 \cdots t,\ t = 1 \cdots T \tag{4.21}$$

As in $\mathcal{TP}_0$, the objective function (4.17) of $\mathcal{TP}_j$ minimizes the sum of all the shipment levels on inadmissible routes, and (4.18) and (4.19) follows from (4.9) and (4.10) respectively. Constraint (4.20) as in (4.14) is the demand equation for the dummy demand point used to capture any supplies in excess of the demand. The set S_j in (4.21) is defined as: $S_j : \{Y_{v,v'}^{k,t,\tau} \geq 0, \text{ s.t. } I_j\}$, where I_j is the set of cut constraints in $\mathcal{TP}_j$.

In the initial iteration of the solution procedure, we first solve $\mathcal{RB}$ for a set of $X_{k,t}$. If there are no AVM restrictions, a complete build and pack schedule can then be generated. However, if the AVM is present, a feasible pack schedule may not exist for the current solution of $X_{k,t}$. In our solution scheme we check for

possible AVM violations components-wise, starting with p^0. Given $X_{k,t}$, a feasible component vendor assignment exists for p^0 only if there are solutions to (4.6) and(4.7) with $X_{k,t}$ held at the given levels. To check this we can solve $\mathcal{TP}_0$, fixing the demand quantities on the right-hand side of (4.13). If $Z = 0$ in (4.12), then a feasible assignment exist for the first component p^0, and we proceed to check the assignment for the next component. Otherwise, a benders feasibility cut can be generated using the dual solution of $\mathcal{TP}_0$. Benders feasibility cuts are based on locating extreme rays in the convex polyhedra of the dual bender's sub-problem. The cut is written as:

$$\sum_{k\in K}\sum_{t=1}^{T}\mu_{k,t}\cdot r^{k}_{p^0}\cdot X_{k,t}+\sum_{v\in V_{p^0}}\sum_{t=1}^{T}\nu_{v,t}\cdot m_{v,t}$$
$$+\mu_{\tilde{k}}\cdot\left(\sum_{\tau=1}^{T}\sum_{v\in V_{p^0}}m_{v,\tau}-\sum_{t=1}^{T}\sum_{k\in K}r^{k}_{p^0}\cdot X_{k,t}\right)+\sum_{i\in I_0}b_i\cdot\pi_i\leq 0 \tag{4.22}$$

where $\mu_{k,t}$, $\nu_{v,t}$ and $\mu_{\tilde{k}}$ denote the optimal dual multipliers associated with (4.13), (4.15) and (4.14) respectively. π_i and b_i refer to the dual multiplier and right-hand side value of cut constraint $i \in I_0$. (4.22) is augmented in $\mathcal{RB}$, which is then re-solved for a new set of production levels $X_{k,t}$. $\mathcal{TP}_0$ is solved again, and if new violations are found, the corresponding cuts are then generated. The procedure is repeated iteratively until $Z = 0$ in (4.12).

Once a feasible assignment exist for p^0, we proceed to check for possible AVM violations for the next component. Given $Y^{k,t}_{v}$ $\forall v \in V_{p^{j-1}}$, where p^{j-1} is the immediate predeccesor component of p^j by definition ($j \geq 1$), a feasible assignment

exists for component p^j only if there are solutions to (4.9) and (4.10) with $Y_v^{k,t}$ $\forall v \in V_{p^{j-1}}$ fixed at the given levels. For $j = 1$, $Y_v^{k,t}$ $\forall v \in V_{p^j}$ is obtained directly from the solution of $\mathcal{TP}_0$. For $j > 1$ this is obtained from the solution of $\mathcal{TP}_{j-1}$ and then applying (4.8). $\mathcal{TP}_j$ is then solved. As before, $Z = 0$ in (4.17) indicates that a feasible assignment exists for component p^j, and we proceed to check the next component p^{j+1}. Otherwise, a feasibility cut is generated, and this can be written as:

$$\sum_{k \in K} \sum_{v \in V_{p^{j-1}}} \sum_{t=1}^{T} \mu_{k,v,t} \cdot \frac{r^k_{p^j}}{r^k_{p^{j-1}}} \cdot Y_v^{k,t} + \sum_{v \in V_{p^j}} \sum_{t=1}^{T} \nu_{v,t} \cdot m_{v,t}$$
$$+\mu_{\tilde{k}} \cdot \left(\sum_{\tau=1}^{T} \sum_{v' \in V_{p^j}} m_{v',\tau} - \sum_{t=1}^{T} \sum_{k \in K} \frac{r^k_{p^j}}{r^k_{p^{j-1}}} \cdot Y_v^{k,t} \right) + \sum_{i \in I_j} b_i \cdot \pi_i \leq 0 \qquad (4.23)$$

where here $\mu_{k,v,t}$, $\nu_{v,t}$, and $\mu_{\tilde{k}}$ denote the optimal dual multipliers associated with (4.18), (4.19) and (4.20) respectively. As before b_i and π_i are the dual price and right-hand side value of cut constraint $i \in I_j$. Constraint (4.23) is augmented in I_{j-1} and $\mathcal{TP}_{j-1}$ is then re-solved. Note that $Y_v^{k,t}$ in (4.23) is first replaced back with the transportation route variables of $\mathcal{TP}_{j-1}$. If $\mathcal{TP}_{j-1}$ remains feasible a new set of model assignment values $Y_v^{k,t}$ $\forall v \in V_{p^{j-1}}$ is then generated. This is used to solve $\mathcal{TP}_j$ again, and if new violations are found, the corresponding feasibility cuts are then found. The procedure is repeated until $Z = 0$ in (4.17). If $\mathcal{TP}_{j-1}$ becomes infeasible, then a cut (4.23) will have to be generated from $\mathcal{TP}_{j-1}$ and passed back to the previous problem.

The solution scheme thus proceeds in a nested form, with $\mathcal{TP}_j$ acting as the

benders subproblem for predecessor components and a master problem for succesive components. The optimal solution is found at the end of the solution procedure when $Z = 0$ in (4.17) $\forall p^j$, where $j = 0 \cdots N$. The solution procedure is summarised below.

Multi-Stage Algorithm:

1. Solve $\mathcal{RB}$, yielding production levels $X_{k,t}$. Set the component index $j = 0$.

2. Fix $X_{k,t}$ and solve $\mathcal{TP}_0$. If $Z^* = 0$, increment $j : j = j + 1$ and proceed to Step 3. Otherwise generate cut (4.22), update $\mathcal{RB}$ and return to Step 1.

3. If $j = N + 1$ optimality is reached, and the procedure terminates. If $j = 0$ go to Step 1. Otherwise solve $\mathcal{TP}_j$ using the current $Y_v^{k,t}$ $\forall v \in V_{p^{j-1}}$.

4. If $Z^* = 0$, increment $j : j = j + 1$ and proceed to Step 3. Otherwise generate cut (4.23) and update I_{j-1}. Decrement $j : j = j - 1$ and go to Step 3.

4.3 Implementing $\mathcal{TP}_j$

As the number of customers and component suppliers increase, the size of the problems $\mathcal{TP}_j$ for $j = 1 \cdots N$ may become considerably large since the number of transportation 'routes' in $\mathcal{TP}_j$ is in the order of $|K| \times |V|^2 \times T^2$, where $|K|$ and $|V|$ denote the cardinality of the sets K and V respectively. This may render solution times to increase significantly even if $\mathcal{TP}_j$ is a pure network problem (i.e. I_j is empty). One way to ease the computational burden is to include in $\mathcal{TP}_j$ only the

demand points that has positive demands and hence only the routes serving these points. Solving this compact version of $\mathcal{TP}_j$ can help to reduce the problem size substantially. In the case when a cut is to be generated from $\mathcal{TP}_j$, we need to obtain the dual prices associated with all the demand points. This can be accomplished as follows. First note that the dual feasibility condition associated with each $Y_{v,v'}^{k,t,\tau}$ in $\mathcal{TP}_j$ can be written as:

$$\mu_{k,v',t} + \nu_{v,t} + \sum_{i \in I_j} a_i \cdot \pi_i \leq \Gamma_{k,v,v'} \tag{4.24}$$

where $\mu_{k,v',t}$, $\nu_{v,t}$, π_i and $i \in I_j$ are as previously defined. a_i denotes the coefficient of the route variable in cut constraint i, $i \in I_j$.

Since the omission of the zero demand points do not change the optimal solution, the solution obtained from the compact version of $\mathcal{TP}_j$ is also optimal in the complete $\mathcal{TP}_j$. It then follows from strong duality of linear programming that the corresponding dual solution $\nu_{v,t}^*$ and π_i^* is also feasible and optimal in the complete dual problem. Applying (4.24), the dual prices associated with the zero demand points can then be recovered by simply setting $\mu_{k,v',t} = \min_{v,t} \left(\Gamma_{k,v,v'} - \nu_{v,t}^* - \sum_{i=1}^{I} a_i \cdot \pi_i^* \right)$. In actual implementation, only the set of cut coefficients a_i needs to be stored. Terms in the cuts are included into the model only if the corresponding route variables are designated for non-zero demand points.

In implementing $\mathcal{TP}_j$ with a non-empty set I_j, we apply the Dantzig-Wolfe decomposition algorithm[10] to solve $\mathcal{TP}_j$. Here the *master problem* consists of the cut constraints $i \in I_j$, and the *sub-problem* is the pure transportation problem. The master problem determines an optimal convex combination of corner point

solutions from a subset of available transportation solutions. The dual multipliers associated with I_j are then used to price out a new master variable by solving the transportation problem, where the objective is to minimise the reduced cost of the master variable. Optimality in $\mathcal{TP}_j$ is achieved when no more negative reduced-cost variables can be priced out. The master problem is thus kept to a modest size, and the network structure of the sub-problem is preserved. We refer readers to Ho and Sundarraj[64] for a comprehensive description of implementing the decomposition algorithm.

4.4 Computational Results

In our computational experiments we implemented both the column generation (Chapter 3) and multi-stage decomposition (§4.2) algorithms using three industrial-strength problems sets. All problems were designed based on the hard-disk drive product structure. There are three components, the HSA, disc and the PCB, hence $|P| = 3$. Each problem set consists of ten instances with randomly generated demand and resource levels. Each instance is solved under different scenarios of AVM restrictions. We define the 'restriction level' β of an AVM between two components as the proportion of illegal assignments among all possible vendor assignments, i.e. for an order k and components p^j and p^{j+1},

$$\beta = \frac{\sum_{v \in V_{p^j}, v' \in V_{p^{j+1}}} \Gamma_{k,v,v'}}{|V_{p^j}| \times |V_{p^{j+1}}|}$$

For example, $\beta = 0.3$ indicates that 30% of the component vendor assignments

are not allowed in the AVM. In all our computations we set β to be the same for all orders and components in each scenario. Note that $1 > \beta \geq 0$, where a high β value indicates that there are many illegal component-vendor combinations. On the other hand $\beta = 0$ refers to the case where there are no AVM restrictions.

The solution algorithms were all coded in C++. The CPLEX 7.0 LP and network solver libraries were invoked to solve the decomposed sub-problems. All computations were performed on a Pentium IV 4.0 GHz PC with 512MB RAM and 18GB disk space. Tables 4.1 to 4.3 highlight the results of our computational study, where *Cols* and *Rows* indicate the total number of columns and rows in formulation $\mathcal{LP}$ (see §3.1) respectively. Note that the column count decreases with increasing β since the number of feasible pack-types decreases. Results obtained via three solution approaches are presented, i.e. procedures *GLP*, *CG* and *BD*. *GLP* refers to solving formulation $\mathcal{LP}$, with all feasible pack types enumerated directly using the CPLEX general-purpose LP solver. *CG* and *BD* refer to the column generation procedure and the benders decomposition procedure respectively. *CPU/s* indicates the mean computation time (average of the ten random instances) in seconds obtained at each β level. *Itns* indicate the mean number of pricing iterations used by procedure *CG*.

Table 4.1: Problem $\mathcal{B}$ Set 1: $|K| = 200$, $|V| = 10$, $T = 7$

			CG		BD		GLP
β	Cols	Rows	CPU/s	Itns	CPU/s	Cuts	CPU/s
0	1400000	1617	17.22	8	1.35	0	2614.7
0.1	1134000	1617	14.2	7	1.38	0	2171.0
0.2	896000	1617	13.4	6.7	1.38	0	1450.6
0.3	686000	1617	13.3	6.3	1.39	0	987.4
0.4	504000	1617	12.75	6.3	1.39	0	913.4
0.5	350000	1617	12.65	6	1.4	0	652.0
0.6	224000	1617	12.4	5.9	1.4	0	448.3
0.7	126000	1617	12.2	5.7	1.39	0	340.7
0.8	56000	1617	11.9	5.1	1.74	0.2	240.2
0.9	14000	1617	11	4.7	4.1	2.2	128.5
0.98	560	1617	10.2	4.5	11.7	10.2	18.2

Table 4.2: Problem $\mathcal{B}$ Set 2: $|K| = 100$, $|V| = 20$, $T = 7$

			CG		BD		GLP
β	Cols	Rows	CPU/s	Itns	CPU/s	Cuts	CPU/s
0	5600000	1127	15.25	49	0.98	0	NA
0.1	4536000	1127	13.05	48.2	0.98	0	NA
0.2	3584000	1127	12.78	47	0.98	0.1	NA
0.3	2744000	1127	12.35	46.4	0.98	0.3	NA
0.4	2016000	1127	11.75	43.9	1.45	0.7	7184.25
0.5	1400000	1127	11.11	42.9	1.62	1.2	2402.1
0.6	896000	1127	10.44	41.7	2.82	2.2	1298.5
0.7	504000	1127	10	39.1	3.48	2.8	889.5
0.8	224000	1127	9.45	39.6	7.4	5.8	402.7
0.9	56000	1127	8.65	37.9	10.55	8.5	229.7
0.95	14000	1127	7.45	36.2	15.2	14.6	115.4

Table 4.3: Problem $\mathcal{B}$ Set 3: $|K| = 200$, $|V| = 20$, $T = 7$

			CG		BD		GLP
β	Cols	Rows	CPU/s	Itns	CPU/s	Cuts	CPU/s
0	11200000	1827	NA	56.18	18.6	2.5	0
0.1	9072000	1827	46.85	13	2.5	0	NA
0.2	7168000	1827	45.54	12.3	2.6	0	NA
0.3	5488000	1827	42.66	11.2	2.5	0	NA
0.4	4032000	1827	42.94	11.1	2.5	0	NA
0.5	2800000	1827	38.1	10.5	2.5	0.1	NA
0.6	1792000	1827	38.16	10.1	3.2	0.3	9835
0.7	1008000	1827	38.2	10.1	5.4	1.1	4138
0.8	448000	1827	35.46	8.8	10.7	3.1	1060.8
0.9	112000	1827	34.36	8.3	24	7.1	352.2
0.95	28000	1827	33.7	8.2	100.8	17.6	283.5

In our implementation of *CG*, the master problem initially consists of one column per customer order. For the pricing scheme we solve $|K| \times T$ shortest path problems during each pricing iteration. One new column per shortest-path problem is then augmented into the master problem if it has a negative reduced-cost. A few other pricing schemes were also experimented but generally gave inferior performance for our problem instances. *Cuts* refer to the mean number of cuts generated by procedure *BD*.

We first remark that for all problems sets, both the decomposition procedures *CG* and *BD* outperform *GLP* in computation times even for our smallest problem instances. For large instances (over 2 million variables) CPLEX was unable to load the problem data (indicated by *NA* in Tables 4 and 5) due to limited storage resources, whereas procedures *CG* and *BD* solves the same problems using only

modest computation times. For procedure *CG*, we observe that the number of pricing iterations for problem set 2 (Table 4.2) is generally larger than in problem sets 1 and 3 (Tables 4.1 and 4.3). This may be because $|K|$ in problem set 2 is the lowest, and since each pricing iteration can add up to $|K| \times T$ new columns, less number of columns are being priced out per iteration. Note that the number of component vendors $|V|$ does not change this maximum number of new columns in our pricing algorithm.

We now compare the performance of *CG* and *BD* under different AVM restriction scenarios. Figures 4.1, 4.2 and 4.3 show scatter plots of CPU time against the β level using the results in Tables 4.1, 4.2 and 4.3 respectively. In all problem sets we observe that for lower β values, procedure *BD* outperforms *CG*, while for high β values procedure *CG* outperforms *BD*. Procedure *CG* is relatively insensitive to the β level (with slightly lower CPU times and fewer pricing iterations as β increases), whereas the computational effort required by *BD* increases significantly when β is high (in particular $\beta > 0.9$).

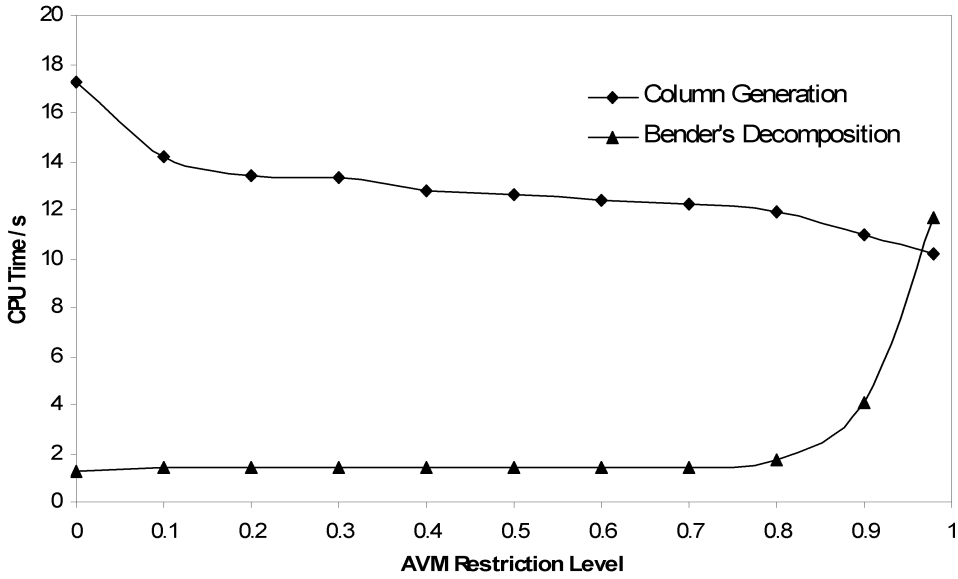

Figure 4.1: CPU Times vs AVM Restriction Level: Problem Set 1

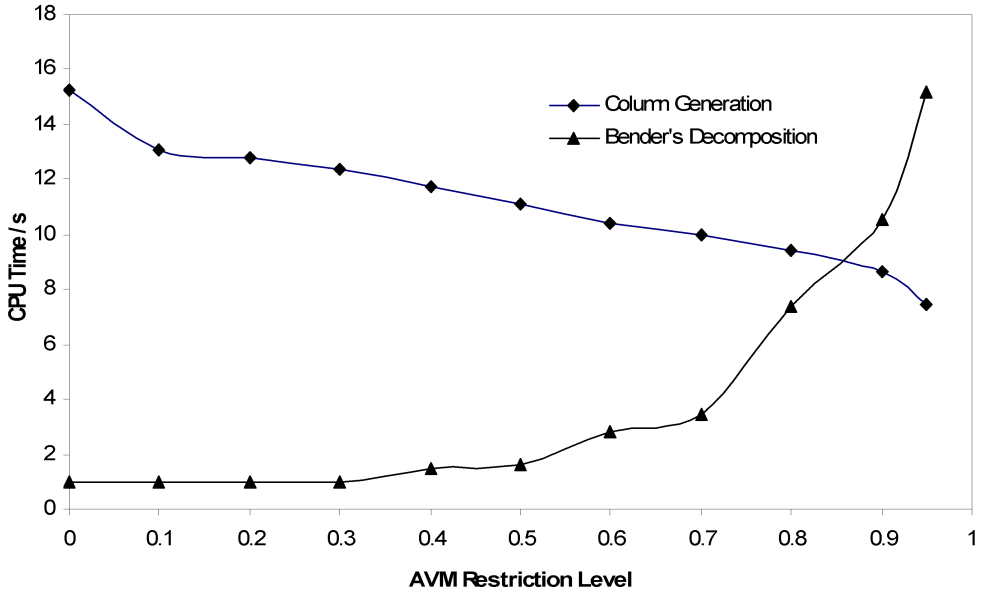

Figure 4.2: CPU Times vs AVM Restriction Level: Problem Set 2

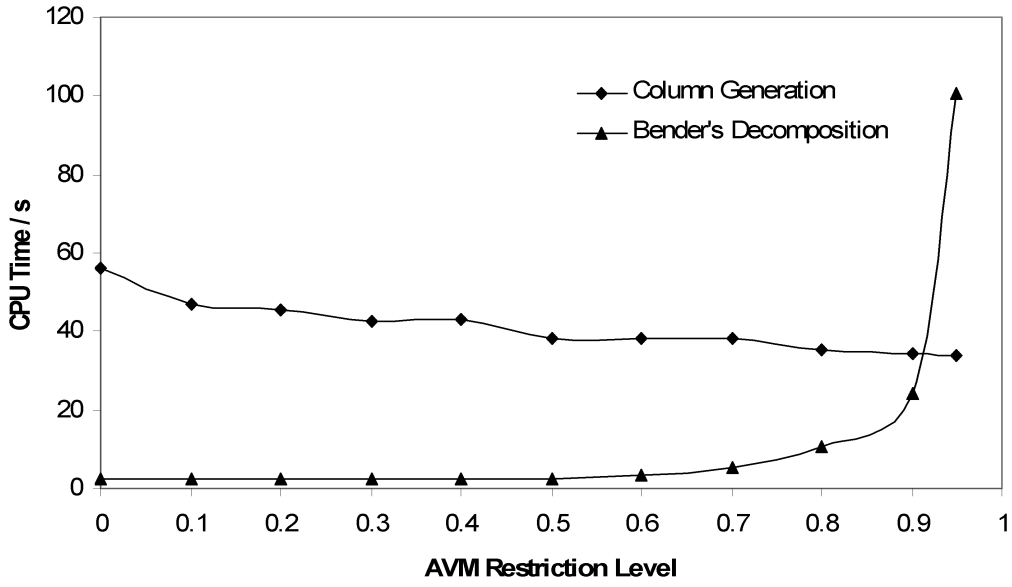

Figure 4.3: CPU Times vs AVM Restriction Level: Problem Set 3

The performance behavior exhibited by procedure *BD* is reasonably intuitive. When β is low, there are relatively few vendor assignments in the AVM that are disallowed. Hence for a given schedule generated in problem $\mathcal{RB}$ the chance of finding a feasible vendor assignment is conceivably higher. For our example problems, solving the sequence of pure transportation problems $\mathcal{TP}_j$ in the first iteration of the procedure proves to be sufficient for obtaining the complete solution when β is relatively low (in the range of 0 to 0.4 on average over all three problem sets). When β increases, more vendor assignments are illegal, and as a consequence an increasing number of feasibility cuts are generated before the solution converges to optimality. Despite this, *BD* still outperforms procedure *CG* consistently until $\beta > 0.9$ (average over all three problem sets) in our example problems.

We note that procedure *BD* is clearly a winner when the AVM restrictions are not too severe. This is in fact the case for the hard-disk drive example, where customers specify the HSA-disk combinations which are not allowed in their orders.

Most HSA-disk assignments are typically acceptable to the customers, and the disallowed assignments turn out to be quite sparse and few in the AVM but cannot be ignored during the course of planning. In formulations $\mathcal{LP}$ or $\mathcal{MCF}$ in Chapter 3, the number of feasible pack-types and flow-paths respectively are extremely large in such cases. Solving either formulations directly is clearly inefficient if not impossible. Procedure *CG*, though relatively robust, is not able to use this information to its advantage. We hence offer procedure *BD* as an alternative approach for such scenarios. On the other hand, in scenarios where the AVM restrictions are so tight that possibly only very few build-types are allowed for each order, it is conceivable that ennumerating all the possible flow-paths would possibly be the best solution approach.

4.5 Concluding Remarks

In this chapter we developed a different solution approach for the multi-period build-pack scheduling problem. This solution approaach uses a multi-stage benders decomposition method that solves the problem in a sequential manner. We have demonstrated in our computational experiments that our approach outperforms both the CPLEX general-purpose LP solver and the column generation technique in the case when the AVM restrictions are not too severe. We thus recommend the use of the multi-stage decomposition method in such scenarios. Possible future extensions of this can include some enhancements to our solution approach in order to accelerate convergence. For instance, the solution scheme in the multi-stage

decomposition generates a single cut at each stage of the problem. In anticipation of a possibly large number of infeasible assignments generated when the AVM restriction level is high, it may be profitable to attempt generating multiple cuts at each stage during a single iteration. One possibility is to make use the existing pool of transportation solutions generated from the Dantzig-Wolfe sub-problem to form demand allocations for the successor problem stage. New feasibility cuts can then be generated. Alternative decomposition schemes other than the sequential procedure we proposed can be considered too. For example, in the first stage independent transportation feasibility problems can be solved for the individual components in parallel. In the second stage the vendor assignments from the first stage are used to build the transportation problems for the component-pairs. Again, feasibility cuts can be generated from both stages when the AVM restrictions are violated, with the cuts from the second stage linking all the first stage problems together in general. Dantzig-Wolfe decomposition can again be used to solve the first stage problem with the cut constraints.

Chapter 5

The Build-Pack Scheduling Problem With Limited Set-ups

In this chapter we consider an extension of the build-pack scheduling problem when there are some special restrictions on work-cell (assembly lines) setups. In the h.d.d. assembly plant, there are a fixed number of work cells in each production period. The work cells are flexible in the sense that they can be set up to assemble any build type, but the current practice is such that these setups or *cell conversions* are only performed at the end of each period. Hence, a cell can only assemble one build type in each period. The motivation of such a practice is not in the technical difficulty of performing cell conversions as actual setup times are negligibly short, but rather for the convenience and ease of monitoring the yields of each batch of build type in production.

In §5.1 we provide a mixed integer program (MIP) formulation of this problem.

We also develop some simple LP rounding procedures in §5.2 to obtain good feasible solutions to the problem. Computational results are presented in §5.3.

5.1 Problem Formulation

In this problem we consider manpower capacity in the units of number of work cells available. The *cell-rate* f is the number units of products that can be produced per cell per period. Let n_t denote the total capacity available in period t in the unit of work cells. Let the activity levels C_t^θ denote the number of cells making build type θ in period t. The total tardiness production planning formulation $\mathcal{LP}$ presented in Chapter 3 is extended to accommodate the cell setup restrictions as follows:

Problem $\mathcal{IP}$

$$\text{Minimise } Z = \sum_{k\in K}\sum_{t=1}^{T} g_t^k \left[\sum_{t'=1}^{t}\left(d_{t'}^k - \sum_{s\in S^k} B_{s,t'}\right)\right]^+ \tag{5.1}$$

subject to:

$$\sum_{\theta\in\vartheta} C_t^\theta \le n_t \qquad \forall\ t = 1\cdots T \tag{5.2}$$

$$\sum_{s\in S_\theta} B_{s,t} \le f\cdot C_t^e \qquad \forall\ \theta\in\vartheta, t = 1\cdots T \tag{5.3}$$

$$\sum_{t'=1}^{t}\Big(\sum_{k\in K_v} r_k^p \sum_{s\in S^k} B_{s,t}\Big) \le \sum_{t'=1}^{t} m_{v,t'} \qquad \forall\ v\in V_p,\ \forall\ p\in P,\ \ t = 1\cdots T \tag{5.4}$$

$$\sum_{t=1}^{T}\sum_{s\in S^k} B_{s,t} \le \sum_{t=1}^{T} d_t^k \qquad \forall\ k\in K \tag{5.5}$$

$$B_{s,t}\in\Re^+,\ \ C_t^\theta\in\mathcal{Z}^+ \qquad s\in S,\ \ \theta\in\vartheta\ \ t = 1\cdots T \tag{5.6}$$

Constraints (5.2) require that the total number cells used is less than the available cells in each period. Constraints (5.3) state that the total units of products built using build type θ must be less than the total production capacity allocated to build type θ in each period t. Constraints (5.6) require C_t^θ to take on values belonging to the set of all positive integers $\mathcal{Z}^+$, and this together with (5.2) and (5.3) is sufficient to ensure that different build types are not built in the same cell in each period. Constraints (5.1), (5.4) and (5.5) are reproduced directly from (3.1), (3.3) and (3.4) in $\mathcal{P}$ for the sake of completeness.

$\mathcal{IP}$ is a linear mixed integer problem with general integer variables, and thus far we are not able to locate special structural properties in the problem that can be exploited for solution efficiency. We hence turn to study heuristics or approximate techniques that allow us to obtain good feasible solutions. In the next section we outline some simple rounding procedures that can be used for that purpose.

5.2 Rounding Procedures For Feasible Solutions in $\mathcal{IP}$

Rounding procedures for integer programs generally consists of two phases. First, the LP relaxation of the original discrete problem is solved. Second, if the solution does not satisfy the integer restrictions, then each fractional variable is either rounded up or down to its nearest integer. The resulting solution then forms an upper bound on the optimal solution. In general, LP rounding does not guaran-

tee optimal or near-optimal solutions. However, by proper design of the rounding scheme the upper bounds on the optimal solution that are obtained can be considerably tightened.

In the following we present four simple rounding procedures $R1$, $R2$, $R3$ and $R4$ for obtaining feasible solutions to $\mathcal{IP}$. $R1$ and $R2$ are based on straightforward rounding down of the LP relaxation, while $R3$ and $R4$ attempts to use some dual information myopically to influence the rounding decisions. The rounding procedures are outlined below. Before proceeding we define problem $\mathcal{IP}'$, which is the linear relaxation of $\mathcal{IBP}$ with a set of fixed $C_t^\theta \ \forall \theta \subset \hat{\vartheta}$, $s \in \hat{S}$, where $\hat{\vartheta}$ and $\hat{S}$ are the set of all build types and pack types respectively in the column pool of problem $\mathcal{MP}$ at the end of the column generation procedure described in §3.2.

Procedure R1

1. Solve $\mathcal{LP}$ using the column generation procedure in §3.2.

2. Check if the solution $\tilde{B}_{s,t}$ satisfies condition :

$$\sum_{\theta \in \hat{\vartheta}} \lceil \frac{\sum_{s \in S_\theta} \tilde{B}_{s,t}}{f} \rceil \leq n_t \qquad t = 1 \cdots T \tag{5.7}$$

   If it does, then the solution is optimal in $\mathcal{IP}$. The procedure hence terminates. Otherwise, proceed to Step 3.

3. Compute the number of work cells $\Delta(t)$ in the LP solution that exceed capacity limit for each period t, i.e.

$$\Delta(t) = \sum_{\theta \in \hat{\vartheta}} \lceil \frac{\sum_{s \in S_\theta} \tilde{B}_{s,t}}{f} \rceil - n_t \quad \forall \ t = 1, \cdots, \ T$$

4. For each period t, pick $\Delta(t)$ build types in production with the greatest slacks ($\lceil \frac{\sum_{s \in S_\theta} \tilde{B}_{s,t}}{f} \rceil - \frac{\sum_{s \in S_\theta} \tilde{B}_{s,t}}{f}$) and reduce one work cell of production of each type from the LP solution, i.e. fix $C_t^\theta = \lfloor \frac{\sum_{s \in S_\theta} \tilde{B}_{s,t}}{f} \rfloor$ if build type θ has been picked. Fix all other C_t^θ at $C_t^\theta = \lceil \frac{\sum_{s \in S_\theta} \tilde{B}_{s,t}}{f} \rceil$.

5. Solve $\mathcal{IP}'$. The resulting solution is feasible in $\mathcal{IP}$.

Procedure R2

1. Solve $\mathcal{LP}$ using the column generation procedure in §3.2.

2. Check if the LP solution $\tilde{B}_{s,t}$ satisfies condition (5.7). If it does, then the solution is feasible in $\mathcal{IBP}$ and the procedure terminates. Otherwise, proceed to Step 3.

3. For each period t that does not satisfy (5.7), select one build type $\theta' \in \hat{\vartheta}$ with the greatest slack. Fix $C_t^\theta = \lfloor \frac{\sum_{s \in S_{\theta'}} \tilde{B}_{s,t}}{f} \rfloor$ if $\theta = \theta'$, and $C_t^\theta = \lceil \frac{\sum_{s \in S_{\theta'}} \tilde{B}_{s,t}}{f} \rceil$ otherwise .

4. Re-solve $\mathcal{IP}'$. Go to Step 2.

Procedure R3

1. Solve $\mathcal{LP}$ using the column generation procedure in §3.2. Set the iteration $u = 1$.

2. Check if the LP solution $\tilde{B}_{s,t}$ satisfies condition (5.7). If it does, then the solution is feasible in $\mathcal{IBP}$ and the procedure terminates. Otherwise, proceed to Step 3.

3. Set the right-hand side of (5.3) as $\frac{\sum_{s \in S_{\theta'}} \tilde{B}_{\theta,t}}{f}$, and re-solve the LP relaxation of $\mathcal{IP}'$. Obtain the dual variables $\pi_{\theta,t}$ associated with (5.3).

4. For each period t that does not satisfy (5.7), select build type $\theta' \in \hat{\vartheta}$ where

$$\theta' = arg\left(\max_{\theta \in \vartheta} \ \pi_{\theta,t} \cdot \left(\frac{\sum_{s \in S_{\theta'}} \tilde{B}_{s,t}}{f} - \lfloor \frac{\sum_{s \in S_{\theta'}} \tilde{B}_{s,t}}{f} \rfloor\right)\right)$$

.

5. Fix $C_t^{\theta} = \lfloor \frac{\sum_{s \in S_\theta} \tilde{B}_{s,t}}{f} \rfloor$ if $\theta = \theta'$, $C_t^{\theta} = \lceil \frac{\sum_{s \in S_\theta} \tilde{B}_{s,t}}{f} \rceil$ otherwise. Re-solve $\mathcal{IP}'$. Go to Step 2.

Procedure R4

1. Begin with the feasible solution to $\mathcal{IBP}$ from *R3*. denote $\pi_{\theta,t}$ as the dual variable associated with each constraint in (5.4).

2. Increase $C_t^{\theta'}$ by one work cell, where $\theta' = arg\left(\min_{\theta \in \vartheta} \ \pi_{\theta,t}\right)$ in the current solution. Re-solve $\mathcal{IP}$.

3. Check condition (5.7) for $t = t'$. If (5.7) is satisfied go back to Step 2. Otherwise select the build-type θ'' with the greatest slack in (5.3) for $t = t'$. If $\theta'' = \theta'$ no improvement is being made to the solution and the procedure terminates. Otherwise proceed to Step 4.

4. Fix $C_{t'}^{\theta''} = \lfloor \frac{\sum_{s \in S_\theta} \tilde{B}_{s,t}}{f} \rfloor$. Re-solve $\mathcal{IP}'$ and go back to Step 2.

5.3 Computational Results

We use three problem sets to test the solution procedures. Each set consists of eight problem instances with the same number of order-types and component models. Demand, capacity and the AVM data though randomly generated in each instance, were also scaled in proportion to industrial parameter sizes. All problems were designed based on the hard-disk drive case. Each instance was solved using *R1-R4* and also solved using CPLEX 7.0 MIP Solver on a Pentium IV, 2.4 GHz PC with 512 MB RAM. All the algorithms were coded in C++. Tables 5.1, 5.2 and 5.3 highlight the results of our computational study.

Table 5.1: Problem $\mathcal{IP}$ Set 1: $|K| = 50$ $|V| = 5$ $T = 7$

			R1		R2		R3		R4		GMIP	
Instance	Cols	Rows	Ratio	CPU/s	Ratio	CPU/s	Ratio	CPU/s	Ratio	CPU/s	Ratio	CPU/s
1	44975	1337	0.997	0.5	0.997	0.6	0.998	2.8	0.999	6.2	1	848.33
2	44975	1337	0.646	0.7	0.786	0.7	0.738	4	0.785	11	0.807	10000
3	32095	1337	0.993	0.3	0.995	0.3	0.993	1.2	0.995	3.2	1	389.4
4	32095	1337	0.988	0.4	0.997	0.5	0.998	1.3	0.999	6.5	1	416.1
5	32095	1337	0.993	0.3	0.994	0.3	0.996	1.7	0.992	5.4	1	243.2
6	21455	1337	0.969	0.4	0.99	0.5	0.991	2.9	0.994	6.2	1	986.2
7	44975	1337	0.933	0.5	0.957	0.5	0.958	2.4	0.957	5.1	0.973	10000
8	44975	1337	0.984	0.4	0.986	0.4	0.986	1.5	0.987	3.2	0.99	10000

Table 5.2: Problem $\mathcal{IP}$ Set 2: $|K| = 100$ $|V| = 5$ $T = 7$

			R1		R2		R3		R4		GMIP	
Instance	Cols	Rows	Ratio	CPU/s	Ratio	CPU/s	Ratio	CPU/s	Ratio	CPU/s	Ratio	CPU/s
1	63315	1687	0.891	1.9	0.942	2.7	0.945	48.8	0.941	51.5	0.978	10000
2	89075	1687	0.984	2	0.989	2.3	0.992	17.1	0.992	28.4	0.972	10000
3	63315	1687	0.891	2.2	0.957	2.6	0.954	62.1	0.958	90.7	1	3968.1
4	63315	1687	0.95	1.1	0.98	1.2	0.981	19.7	0.991	36.5	1	4819.5
5	89075	1687	0.823	2.2	0.922	3	0.933	38.6	0.93	72.5	1	5317.7
6	89075	1687	0.977	1.5	0.99	1.7	0.991	17.5	0.997	31.6	1	4288.5
7	63315	1687	0.926	1.8	0.957	2.1	0.958	25.8	0.957	48.6	0.97	10000
8	89075	1687	0.968	1.7	0.985	2	0.987	15.8	0.994	25.1	1	9197.5

Table 5.3: Problem $\mathcal{IP}$ Set 3: $|K| = 100$ $|V| = 10$ $T = 7$

			R1		R2		R3		R4		GMIP	
Instance	Cols	Rows	Ratio	CPU/s	Ratio	CPU/s	Ratio	CPU/s	Ratio	CPU/s	Ratio	CPU/s
1	259700	7917	0.924	2.8	0.969	3.3	0.969	45.5	0.967	203.2	0.886	10000
2	259700	7917	0.98	3.3	0.985	3.3	0.987	61	0.99	233.6	0.94	10000
3	259700	7917	0.929	3.6	0.967	4.5	0.967	59.2	0.97	220.4	0.794	10000
4	259700	7917	0.94	3.7	0.969	5.1	0.972	43.3	0.969	275.9	0.868	10000
5	259700	7917	0.262	5.9	0.427	9.5	0.382	151.9	0.422	450.1	0.181	10000
6	259700	7917	0.984	3	0.991	3.3	0.99	58.6	0.994	183.6	0.891	10000
7	119700	7917	0.961	3.7	0.983	4.8	0.981	83.1	0.984	400.1	0.982	10000
8	119700	7917	0.988	3.3	0.994	3.75	0.992	45.2	0.994	210.5	0.993	10000

Cols and *Rows* refer to the problem sizes. *CPU* refers to the time in seconds used by the solution algorithms. *GMIP* refers to solving the problem instance using the general purpose CPLEX MIP solver. We set a solution time limit of 10,000 seconds and terminate the computations if *GMIP* does not find the optimal solution within the time limit. The best feasible solution obtained is presented. *Ratio* refers to the ratio of a valid lower bound to the solution obtained by the various

procedures. For problems solved to optimality by *GMIP*, the optimal solution is used for the lower bound. Otherwise, we use the LP relaxation of $\mathcal{IBP}$ for the lower bound.

The computation results shows that even the simplest procedures *R1* and *R2* gives reasonably good performance, with the solutions obtained within at least 90% of the respective lower bounds, validating also the robustness of the heuristics, with the exception of instance 5 in problem set 3 (Table 5.3). In this case the LP lower bound was only 42.2% of the best upper bound obtained (using *R*4). However all the rounding heuristics still outdo *GMIP*, which could only obtain a *Ratio* of 18.1%. The inferior solution quality in this instance may be due to a case of a weak LP bound, which results in a large gap between the optimal integer solution and the LP solution.

Although *GMIP* produced tighter upper bounds for most problems in Tables 5.1 and 5.2 at the end of the imposed time limit, the computation times required by the rounding heuristics to obtain their bounds are reasonably small in comparison. For problems in Tables 5.1 and 5.2 solved to completion by *GMIP*, *R*4 takes an average of 1.5% of the solution time required by CPLEX to form upper bounds at 98% of the optimal solution. Further, we remark that for all the problems in Tables 5.1 to 5.3, when *GMIP* was run for an equal amount of CPU time used by *R*4, no integer solutions were found. The rounding heuristics outperform the CPLEX solutions on the largest problems (Table 5.3), obtaining tighter upper bounds at modest computation times (average of 2.7% of the 10,000s used by *GMIP*. In other

words $R4$ uses less than 2.7% of the computation time needed by CPLEX to reach that same bound. Generally it is observed that $R3$ and $R4$ gives marginally better bounds than $R1$ and $R2$ but at the expense of more computation time used for the dual price calculations.

5.4 Concluding Remarks

In this chapter we have considered an extension of the multi-period build-pack scheduling problem that was formulated and solved in Chapters 3 and 4. The problem scenario is such that, in each production period, the number of production set-ups are limited. This effectively constrains the maximum number of kinds of build-types that can be produced in each period. A MIP formulation is then developed, which is based on the column generation formulation $\mathcal{P}$ of Chapter 3. Rather than solving the MIP directly, we developed some heuristics that simply perform rounding operations on the LP relaxation to produce upper bounds. These upper bounds are demonstrated to be reasonably tight and requires modest amounts of computation times.

For the rounding heuristics, as a managerial guide we suggest that, for a quick evaluation of the schedule in the situation of frequent capacity re-planning, $R1$ or $R2$ may be used. For example, the planning cycle may consist of several exchanges or negotiations in the capacity schedule between the production planning and industrial engineering (IE). These negotiations may go back and forth for several rounds before both parties are satisfied. In each round of negotiation the IE pro-

poses an updated capacity schedule. The production manager has to then evaluate the impact of the capacity schedule on the quality of his production schedule, and decide whether to accept or reject the proposed capacity change. If the change is rejected, the IE then modifies the capacity schedule and the negotiation cycle is repeated. In this case it may not be possible to devote excessive time to compute the optimal production schedule for each round of the negotiation. $R1$ or $R2$ can then be used to re-generate the production schedules for a conservative evaluation during the negotiations in the order of seconds. Once the final capacity schedule is published, $R3$ or $R4$ can then be applied for an improved final solution.

Chapter 6

The Build-Pack Planning Problem With Stochastic Demands

We consider the *build-pack planning problem with stochastic demands* in this chapter. The motivation and the description of the problem scenario is given in §2.2.2. A summary of the problem is as follows. Given a fixed set of component supplies, we are interested in evaluating a production plan (or build plan) that achieves as low a total production costs as possible prior to full knowledge of the customer demands. The production plan determines the total levels of each build-type to produce over the entire time bucket. The total production costs consists of shortage costs for unfulfilled demand and holding costs for excess production on expectation. When customer demands are realised, the build-types are assigned to fulfill the demands, observing the AVM requirements of the customers.

The outline for this chapter is as follows. In §6.1 we presents a set-partitioning

type formulation for the problem. §6.2 develops the column generation algorithm to solve the linear relaxation of the model, and based on this, §6.3 then presents a branch-and-price solution framework designed for the problem. Finally, §6.4 present some computational results of our implementation.

6.1 The Partitioning Policy Formulation

We assume that the customer demands d_k are independently and normally distributed with mean μ_k and standard deviation σ_k for each customer k. The distribution parameters may be estimated either using past demand data, or from the demand forecasting module if one exists. The normality assumption is relatively common in production planning and inventory control models, and is reasonable for most high-volume manufacturing environments. We consider the treatment of arbitrary distributions in Chapter 7.

Two key characteristics of the build-pack planning problem: high build-type proliferation and an extremely large number of random outcomes, renders the use general stochastic programming techniques like the L-shaped method or scenario decomposition methods impractical. In this work we adopt a set partitioning formulation for the problem. A feasible solution arising from the set-partitioning formulation is termed here as a *partitioning policy*. Such a policy requires a set partition over all customers to be determined *a priori* to demand realization. When the demands are realised, this set partition is not changed. Although a partitioning policy cannot be guaranteed to be optimal among all policies, it is considerably

more tangible and easier to execute in actual operation. Essentially, the idea of a partitioning policy is to consider the demands in terms of *customer pools* (i.e. groups of customers), rather than the individual demands. Hence the build-plans consist of allocating build-types towards these customer pools. Note that the AVM restrictions require these build-types to observe the combined AVM of the customer pool. The realised demands are then fulfilled with these allocated build-types. A larger customer pool can take advantage of the so-called 'risk-pooling' effect[127] , which serves to decrease demand variability, but at the same time stands to incur some opportunity costs associated with over-restricting the set of build-types or resources usable due to the combined AVM restrictions in the customer pool. An optimal partitioning policy is one that forms customer pools that gives the best trade-off between the benefit of the risk-pooling effect, versus the opportunity loss in restricting the set of build-types usable due to the AVM requirements, thus achieving minimum total costs on expectation.

Let m_v be the components supply from vendor v. Let i denote the index of customer pool $K_i \subseteq K$, and I the set of indices of all customer pools. The decision variables are:

$$Y_i = \begin{cases} 1 & \text{if } K_i \text{ appears in the solution} \\ 0 & \text{otherwise.} \end{cases}$$

$Q_{i,\theta} \geq 0$: the total level of build-type θ allocated to customer pool K_i

The set-partitioning formulation is as follows:

Problem $\mathcal{IBP}'$

$$\text{Minimize } Z = \sum_{i \in I} C_i \left(\sum_{\theta \in \vartheta_i} Q_{i,\theta} \right) \cdot Y_i \tag{6.1}$$

subject to:

$$\sum_{i \in I} \alpha_i^k \cdot Y_i \geq 1 \qquad \forall\ k \in K \tag{6.2}$$

$$\sum_{\theta \in \vartheta_i} Q_{i,\theta} \leq M \cdot Y_i \qquad \forall\ i \in I \tag{6.3}$$

$$\sum_{i \in I} \sum_{\theta \in \vartheta_i} \beta_\theta^v \cdot Q_{i,\theta} \leq m_v \qquad \forall\ v \in V_p\ , \forall\ p \in P \tag{6.4}$$

$$Q_{i,\theta} \geq 0\ ,\ Y_i \in \{0,1\} \qquad \forall\ i \in I\ \forall\ \theta \in \vartheta \tag{6.5}$$

where

$$\alpha_i^k = \begin{cases} 1 & \text{if } K_i \text{ contains order } k \\ 0 & \text{otherwise.} \end{cases}$$

$$\beta_\theta^v = \begin{cases} 1 & \text{if build plan } \theta \text{ uses material } v \\ 0 & \text{otherwise.} \end{cases}$$

$C_i(\cdot)$ in (6.1) is the production cost function associated with customer pool K_i. It consists of penalty costs g per unit short of the total demand of K_i, and holding costs h per unit production in excess of the total demand of K_i. Since the demands of K_i is random, C_i is an expectation, and can be regarded as parametric function of Q_i. Denoting $\mu(i) = \sum_{k \in K_i} \mu_k$ and $\sigma(i) = \sqrt{\sum_{k \in K_i} \sigma_k^2}$ for the aggregated demand $\sum_{k \in K_i} d_k$, the expected cost associated with K_i is:

$$C_i(Q_i) = h \cdot (Q_i - \mu(i)) + \sigma(i) \cdot (g + h) \cdot G\left(\frac{Q_i - \mu(i)}{\sigma(i)} \right) \tag{6.6}$$

where for notational convenience we denote $Q_i = \sum_{\theta \in \vartheta_i} Q_{i,\theta}$, ϑ_i being the set of all build-types that complies with the AVM of K_i. The function $G(\cdot)$ denotes the standard normal 'loss' function, where $G(\kappa) = \int_{\kappa}^{\infty} (z - \kappa) \cdot \phi(z) dz$ for $\kappa, z \in \Re$, and $\phi(\cdot)$ is the standard normal density function.

$C_i(\cdot)$ is also known as the news-vendor cost, and the optimum solution Q_i^* is given by: $Q_i^* = \mu(i) + \kappa^* \cdot \sigma(i)$, where κ^* is so that $\Phi(\kappa^*) = \frac{g}{g+h}$, and $\Phi(\cdot)$ is the standard normal distribution function. The minimum costs C_i^* is given by:

$$C_i^* = (g + h) \cdot \sigma(i) \cdot \phi(\kappa^*) \tag{6.7}$$

Constraints (6.2) ensure that each customer k is covered only once in the optimal solution. Constraints (6.3) ensure that a build-type allocation occurs only if the corresponding customer pool is activated. Constraints (6.4) are the limitations on the components availability.

Since $\mathcal{IBP}'$ is a minimization program and $C_i(\cdot)$ is known to be convex in Q_i for each K_i, we can create piece-wise linear approximations of C_i using a set of discretized levels as in usual linear programming techniques. Note that the convexity of $C_i(\cdot)$ holds for any demand distributions in general. In the following we develop the piece-wise linear approximation of IBP'. Define parameters $Q_{i,j,\theta}$ to be the j^{th} level of build for pool K_i using θ, $j = 1 \cdots J_i$, where Q_{i,θ,J_i} is some maximum build-level for K_i. Define the rational *indicator* variables $X_{i,\theta,j}$ and s_i, where $1 \geq X_{i,\theta,j} \geq 0$, $1 \geq s_i \geq 0$. $X_{i,\theta,j}$ indicates a build-level of $X_{i,\theta,j} \cdot Q_{i,\theta,j}$. s_i is viewed as the fraction of demand unfulfilled for K_i. In particular $s_i = 1$ indicates the case when $Y_i = 1$ but $Q_i = 0$ in $\mathcal{IBP}'$. In general, any total build-

level $0 \leq Q_i \leq Q_{i,\theta,J_i}$ can be indicated by a feasible solution of the convexity condition: $s_i + \sum_{j=1}^{J_i} \sum_{\theta \in \vartheta_i} X_{i,\theta,j} = 1$. Using this convexity condition in conjunction with the function convexity of $C_i(\cdot)$ in Q_i, a proper piece-wise linear approximation for $C_i(\cdot)$ can then be formed as follows:

$$\text{Minimize } Z = \sum_{j=1}^{J_i} \sum_{\theta \in \vartheta_i} C_i\left(Q_{i,\theta,j}\right) \cdot X_{i,\theta,j} + C_i(0) \cdot s_i \tag{6.8}$$

subject to:

$$\sum_{j=1}^{J_i} \sum_{\theta \in \vartheta_i} X_{i,\theta,j} + s_i = 1 \tag{6.9}$$

Note that in (6.8) $C_i(0) = g \cdot \sum_{k \in K_i} \mu(k)$. The complete piece-wise linear approximation of $\mathcal{IBP}'$ can then be written as:

Problem $\mathcal{IBP}$

$$\text{Minimize } Z = \sum_{i \in I} \left(\sum_{j=1}^{J_i} \sum_{\theta \in \vartheta_i} C_i\left(Q_{i,\theta,j}\right) \cdot X_{i,\theta,j} + \sum_{k \in K_i} g \cdot \mu(k) \cdot s_i \right) \tag{6.10}$$

subject to:

$$\sum_{i \in I} \alpha_i^k \cdot Y_i \geq 1 \qquad \forall\ k \in K \tag{6.11}$$

$$\sum_{j=1}^{J_i} \sum_{\theta \in \vartheta_i} X_{i,\theta,j} + s_i = Y_i \qquad \forall\ i \in I \tag{6.12}$$

$$\sum_{i \in I} \sum_{\theta \in \vartheta_i} \beta_\theta^v \cdot \sum_{j=1}^{J_i} Q_{i,\theta,j} \cdot X_{i,\theta,j} \leq m_v \qquad \forall\ v \in V_p\ ,\ \forall\ p \in P \tag{6.13}$$

$$Y_i \in \{0,1\},\ 1 \geq s_i \geq 0,\ 1 \geq X_{i,\theta,j} \geq 0 \qquad \forall\ i \in I\ ,\ \forall\ \theta \in \vartheta\ ,\ j = 1 \cdots J_i \tag{6.14}$$

The objective function(6.10) is the sum of the piece-wise approximations (6.8) over all $i \in I$. Constraints (6.11) is reproduced from (6.2) directly. Constraints

(6.12) follows from (6.9) to ensure that the cost associated with each customer pool K_i is a convex combination of s_i and $X_{i,\theta,j}$ $\forall \theta \in \vartheta_i$, $j = 1 \cdots J_i$. (6.12) like in (6.3) also ensures that a build-type is allocated to a customer pool only if the customer pool is activated (i.e. $Y_i = 1$). The components availability constraints (6.13) follow from (6.4). For the rest of this work we will focus our attention on solving the formulation IBP.

The linear relaxation of IBP is obtained by substituting out the binaries Y_i from the formulation using (6.12). Defining $X'_k = \sum_{i \in I} \alpha_i^k \cdot s_i$, the following formulation results:

Problem $\mathcal{BP}$:

$$\text{Minimize } Z = \sum_{i \in I} \sum_{\theta \in \vartheta_i} \sum_{j=1}^{J_i} C_i\left(Q_{i,\theta,j}\right) \cdot X_{i,\theta,j} + \sum_{k \in K} g \cdot \mu_k \cdot X'_k \tag{6.15}$$

subject to:

$$\sum_{i \in I} \alpha_i^k \cdot \sum_{\theta \in \vartheta_i} \sum_{j=1}^{J_i} X_{i,\theta,j} + X'_k \geq 1 \qquad \forall\ k \in K \tag{6.16}$$

$$\sum_{i \in I} \sum_{\theta \in \vartheta_i} \beta_\theta^v \sum_{j=1}^{J_i} Q_{i,\theta,j} \cdot X_{i,\theta,j} \leq m_v \qquad \forall\ v \in V_p\ ,\ \forall\ p \in P \tag{6.17}$$

$$1 \geq X_{i,\theta,j} \geq 0\ ,\ 1 \geq X'_k \geq 0 \qquad \forall k \in K\ ,\ \forall\ i \in I\ ,\ \forall\ \theta \in \vartheta\ ,\ j = 1 \cdots J_i \tag{6.18}$$

The objective function (6.15) and constraints (6.16) are obtained from (6.10) and (6.11) respectively by substituting out Y_i. X'_k $\forall k \in K$ can be interpreted as artificial variables introduced to ensure feasibility in BP, or as a build plan to serve k with zero build level, and hence has cost $g \cdot \mu_k$. Note that in a proper set-partition, (6.16) together with (6.15) still forms the appropriate piece-wise linear

approximation of total expected cost function. Constraints (6.17) is reproduced from (6.12) directly for the sake of completeness.

6.2 Solving Problem $\mathcal{BP}$

In this section we consider solving the linear problem $\mathcal{BP}$. Note that in the formulation of $\mathcal{BP}$, if the AVM restrictions are not present, then optimal pool K_i^* is such that $\alpha_{i*}^k = 1 \; \forall \; k \in K$, i.e. all the demands are pooled together. Furthermore, since there is no restrictions on the set of build-types to use, all the build-types become essentially indistinguishable, and hence the optimal build-level can be determined easily. In general however, all build-types are unique, and $\mathcal{BP}$ has a total column count in the order of $2^{|K|} \times |V|^{|P|}$. This renders solution by general purpose LP-solvers impractical if not impossible for any realistic problem instances. As in the deterministic build-pack scheduling problem we propose a column generation approach to address this problem.

Define $\mathcal{RBP}$ as a *restricted* version of $\mathcal{BP}$, where $\mathcal{RBP}$ contains only a subset of all feasible columns in $\mathcal{BP}$. The solution procedure is outlined as such. First $\mathcal{RBP}$ is solved to optimality using any general-purpose linear solver. Next, we check for further improvements that can be made to the current solution by considering new customer pools or build-types not in the current solution. This corresponds to scanning for non-basic columns with negative reduced costs in BP. If there are no such columns then the current solution is optimal for $\mathcal{BP}$ and the procedure ends. Otherwise, these columns are augmented to I' and $\mathcal{RBP}$ is re-solved.

Column generation allows new entering variables to be located without explicit enumeration of all the columns. This is accomplished by formulating the search for new variables as a *pricing problem*. In our pricing scheme, we search for a subset K_i, a corresponding feasible build-type θ, and the associated maximum build-level Q_{i,θ,J_i} that minimizes the reduced cost of a non-active X_{i,θ,J_i}. The columns $X_{i,\theta,j}$ for $j = 1 \cdots J_i$ are then augmented in $\mathcal{RBP}$. The pricing problem is stated as:

Problem $\mathcal{SBP}$

$$\text{Minimize } Z_{i,\theta,J_i} = C_i(Q_{i,\theta,J_i}) - \sum_{k \in K} \gamma^k \cdot \alpha_i^k - \sum_{p \in P} \sum_{v \in V_p} \pi^v \cdot \beta_\theta^v \cdot Q_{i,\theta,J_i} \tag{6.19}$$

subjected to:

$$\sum_{v \in V_p} \beta_\theta^v = 1 \qquad \forall\ p \in P \tag{6.20}$$

$$\alpha_i^k + \beta_\theta^v + \beta_\theta^{v'} \leq 3 - \Gamma_{k,v,v'} \qquad \forall\ k\ ,\ \forall\ v \in V_p\ ,\ \forall\ v' \in V_{p'}\ ,$$

$$\text{where } p,\ p' \in P, p \neq p' \tag{6.21}$$

$$\alpha_i^k\ ,\ \beta_\theta^v \in \{0,1\}\ ,\ Q_{i,\theta,J_i} \geq 0 \qquad \forall\ k \in K\ ,\ \forall\ v \in V_p\ ,\ \forall\ p \in P \tag{6.22}$$

(6.19) is the reduced cost expression of $X_{i,\theta,J}$. γ^k and π^v are the dual prices associated with (6.16) and (6.17)respectively. Note that $\gamma^k \geq 0$ and $\pi^v \leq 0$. (6.20) restricts one supplier v per component p. (6.21) enforces the AVM for the set of orders K_i.

Problem $\mathcal{SBP}$ is a binary integer problem and is non-linear in the objective function. The rest of this section develops the main components of the algorithm for solving $\mathcal{SBP}$. §6.2.1 presents procedure *Path* for solving $\mathcal{SBP}$ in the case when the customer pool K_i fixed. §6.2.2 develops procedure *Group* for solving $\mathcal{SBP}$ when

the build-type θ is fixed. In both cases we show that the respective problems can be solved efficiently. Using the results of these two special cases, a specialized branch-and-bound procedure is then developed in §6.2.3 to solve $\mathcal{SBP}$ for the minimum reduced cost.

6.2.1 Solving $\mathcal{SBP}$ When Customer Pool K_i is Fixed

In this case a subset K_i is given, and we need to minimize (6.19) over β^v and Q_i, (we temporally suppress the subscripts θ and j for clarity of notation). Substituting the expression for $C_i(Q_i)$ from (6.6) we can re-write (6.19) as:

$$Z_i = (h - \sum_{p\in P}\sum_{v\in V_p} \pi^v \cdot \beta^v) \cdot (Q_i - \mu(i)) + \sigma_i \cdot (g+h) \cdot G\left(\frac{Q_i - \mu(i)}{\sigma(i)}\right) - \left(\sum_{k\in K_i} \gamma^k + \mu(i) \cdot \sum_{p\in P}\sum_{v\in V_p} \pi^v \cdot \beta^v\right)$$

where Z_i denotes the objective function in (6.19) for fixed K_i. Denoting $\overline{\pi} = -\sum_{p\in P}\sum_{v\in V_p} \pi^v \cdot \beta^v$, $h' = h + \overline{\pi}$ and $g' = g - \overline{\pi}$, the above becomes:

$$Z_i = h' \cdot (Q_i - \mu(i)) + \sigma_i \cdot (g' + h') \cdot G\left(\frac{Q_i - \mu(i)}{\sigma(i)}\right) - \left(\sum_{k\in K_i} \gamma^k - \mu(i) \cdot \overline{\pi}\right)$$

The first two terms in the above corresponds exactly to the news-vendor problem with parameters g' and h'. Noting that (6.20)-(6.22) are independent of Q_i, and substituting the expression for minimum costs of the news-vendor problem (6.7) in the above, we have:

$$Z_i(\overline{\pi}) = (g' + h') \cdot \sigma(i) \cdot \phi(\kappa^*) - \left(\sum_{k\in K_i} \gamma^k - \mu(i) \cdot \overline{\pi}\right)$$

where here Z_i is expressed explicitly as a function of $\overline{\pi}$. As was previously defined, κ^* is so that $\Phi(\kappa^*) = \dfrac{g'}{h'+g'} = \dfrac{g-\overline{\pi}}{h+g}$.

We will now go on show that, for nonnegative build-levels, i.e. $Q_i \geq 0$, the pricing problem when K_i is fixed is equivalent to a shortest path problem. To establish this, we first state the following lemma:

Lemma 1: *$Z_i(\overline{\pi})$ is non-decreasing in $\overline{\pi}$ whenever $Q_i \geq 0$.*

Proof. Recall that from the news-vendor problem, our optimal build-level is given by $Q_i^* = \mu(i) + \kappa^* \cdot \sigma(i)$. Since we are concerned only with non-negative build-levels, it is necessary that $\kappa^* \geq -\dfrac{\mu(i)}{\sigma(i)}$.

Taking first derivatives on $Z_i(\overline{\pi})$, we have:

$$\begin{aligned}
\tfrac{\partial Z_i}{\partial \overline{\pi}} &= \sigma(i) \cdot (g+h) \cdot \tfrac{\partial \kappa^*}{\partial \overline{\pi}} \cdot \tfrac{\partial \phi(\cdot)}{\partial \kappa^*} + \mu(i) \\
&= \sigma(i) \cdot (g+h) \cdot \left(\tfrac{-1}{(g+h)\cdot\phi(\kappa^*)} \right) \cdot (-\kappa^* \cdot \phi(\kappa^*)) + \mu(i) \\
&= \sigma(i) \cdot \kappa^* + \mu(i) \geq 0
\end{aligned}$$

The last inequality follows directly from our definition of κ^*. ∎

Lemma 1 implies that to minimize Z_i, we only need to select the minimum $\overline{\pi}$. To accomplish this we simply solve the following problem:

Problem $\mathcal{SPP}$:

$$\min_{\beta^v} \quad \sum_{p \in P} \sum_{v \in V_p} (-\pi^v) \cdot \beta^v$$

subject to: (6.20)-(6.22).

$\mathcal{SPP}$ requires us to select exactly one supplier per component without violating the AVM constraints of the orders in K_i in a manner such that minimum costs is incurred. This is in fact equivalent to the shortest path network problem $\mathcal{SP}$ in §3.2 (see Figure 3.1 for the example of the network representation in the HDD context), where $\mathcal{G}(N, A)$ denotes the network graph, N being the node set and A the set of arcs in the network connecting the nodes. N consists of source node s, sink node t, and *component nodes* n_v corresponding to the supplier v. The only difference in this case is that an arc linking two component nodes exists only if it does not violate the AVM restrictions for the entire group K_i, rather than for a single customer. More formally, defining any full order $\{p_1, p_2, \cdots, p_{|P|}\}$ on the set of components P, the arc $(v, v') \in A$ only if $v \in V^k_{p^i}$ and $v' \in V^k_{p^{i+1}}$ $(i = 1 \cdots |P|)$, and v and v' does not violate the AVM for K_i. For instance, there is no arc linking the second HSA component node to the second disc component node in Figure 3.1, indicating that the combination of vendors is not allowed in the AVM of the entire group K_i.

In the network arrangement, any feasible walk starting from the source node to the sink node constitute a feasible build type. The problem of finding a minimum cost feasible build type for orders K_i is thus equivalent to searching for the shortest path through the equivalent network of the problem, where the 'length' of an arc feeding into a component node associated with vendor v is equal to $-\pi^v$. The procedure for minimizing Z_i for a given subset K_i is as follows.

Procedure *Path*

1. Establish the network $\mathcal{G}(N, A)$ for the customer set K_i by deleting arcs (v, v') which violate the AVM of K_i.

2. Solve the shortest path problem ($\mathcal{SPP}$) for the optimal path (build-type) $(\beta^V)^*$. Set the minimum path length $\overline{\pi}^* = min(g, \sum_{p \in P} \sum_{v \in V_p} (-\pi^v) \cdot (\beta^v)^*)$.

3. Compute the associated minimal Z_i using the news-vendor cost, i.e.

$$Z_i^* = (g + h) \cdot \sigma(i) \cdot \phi(\kappa^*) - \left(\sum_{k \in K_i} \gamma^k - \mu(i) \cdot \overline{\pi}^* \right)$$

where κ^* is so that $\Phi(\kappa^*) = \dfrac{g'}{h' + g'} = \dfrac{g' - \overline{\pi}^*}{h + g}$.

In Step 2, the minimum path length π^* is set to be at most g, since $\Phi(\kappa^*) \geq 0$. Note that because Lemma 1 is necessary but not sufficient for Q_i^* to be non-negative, it is possible that procedure Path may propose negative reduced cost columns but with $Q_i^* \leq 0$. Such columns are simply discarded from consideration.

6.2.2 Solving the Pricing Problem When Build-type θ is Fixed

This corresponds to the problem of picking a subset K_i and the associated Q_i to minimize the cost for a particular build-type θ, (i.e. β_v and hence $\overline{\pi}$ is fixed). For some fixed $\overline{\pi}$, the minimum cost associated with a group K_i is given by the optimal news-vendor cost:

$$Z_\theta(K_i) = (g' + h') \cdot \phi(\kappa^*) \cdot \sqrt{\sum_{k \in K_i} \sigma_k^2} - \left(\sum_{k \in K_i} \gamma^k - \overline{\pi} \cdot \sum_{k \in K_i} \mu_k \right)$$

where here $Z_\theta(K_i)$ here denotes $Z_{i,\theta}$ in (6.19) for fixed θ and varying with K_i. Note that $\phi(\kappa^*)$ is independent of K_i. $Z_\theta(K_i)$ can be shown to be *submodular* in K_i. Given a ground set $\mathcal{N}$, and subsets $\mathcal{A}$ and $\mathcal{B}$ where $\mathcal{A}$, $\mathcal{B} \subseteq \mathcal{N}$, a function $\mathcal{F} : 2^N \rightarrow \Re$ is submodular if $\mathcal{F}(\mathcal{A} \cup \mathcal{B}) \leq \mathcal{F}(\mathcal{A}) + \mathcal{F}(\mathcal{B}) - \mathcal{F}(\mathcal{A} \cap \mathcal{B}) \; \forall \; \mathcal{A}, \mathcal{B} \subseteq \mathcal{N}$. Submodularity can be shown to hold for $Z_\theta(K_i)$ as a result of the concavity of the square-root term. Submodular function minimization (SFM) problems seek to find the minimizing subset A^*, so that $\mathcal{F}(\mathcal{A}^*) \leq \mathcal{F}(\mathcal{A}) \forall \mathcal{A} \subseteq \mathcal{N}$. SFM problems are known to be solvable in polynomial time, and several new *combinatorial* algorithms have recently been developed (Iwata, Fleishcer and Fujishige[71] and Schrijver[122]) to solve general SFM. Here we apply an optimal ranking algorithm proposed by Shen, Coullard and Mark[124] to solve SFM of the above form arising from a joint location-inventory model. In the following we state the self-explaining solution procedure, where $K^*(\overline{\pi})$ denotes the minimizing set for a given $\overline{\pi}$.

Procedure *Group*

1. Denote $a_k = \gamma^k - \overline{\pi} \cdot \mu_k$, and $b_k = f(\kappa^*) \cdot \sigma_k^2 \;\; \forall k \in K$. If $a_k \leq 0$ for some $k \in K$, the customer k is discarded from the rest of the algorithm. Update K as the remaining set.

2. Rank the customers $k \in K$ in the sequence $\{k_1, k_2 \cdots k_{|K|}\}$, so that

$$-\frac{a_1}{b_1} \leq -\frac{a_2}{b_2} \cdots - \frac{a_{|K|}}{b_{|K|}} \leq 0$$

Denote $K(n)$ as the set of customers $\{k_1, \cdots k_n\}$. Initialize the index $n = 1$, and $min = 0$.

3. If $n > |K|$ go to Step 4. Otherwise compute $Z_i(K(n))$. If $Z_i(K(n)) \leq min$ update $min = Z_i(K(n))$. Increase $n = n + 1$, and repeat Step 3.

4. The minimizing set $K^*(\overline{\pi})$ is the set $K(n)$. Terminate the algorithm.

The minimizing set $K^*(\overline{\pi})$ is also called a *lower ideal* of the ranking order in Step 1. We refer readers to Shen, Coullard and Mark[124] for a proof of the algorithm. We now go on to establish some useful results for our problem.

Lemma 2: *For any $\overline{\pi}' > \overline{\pi}$, the minimizing set $K^*(\overline{\pi}')$ always forms a subset of $K^*(\overline{\pi})$, i.e. $K^*(\overline{\pi}') \subseteq K^*(\overline{\pi})$.*

Proof. For some fixed $\overline{\pi}$ and non-empty subset $K' \subseteq \{K \backslash K^*(\overline{\pi})\}$, the below inequality holds by optimality of $K^*(\overline{\pi})$:

$$\frac{\sum_{k \in K'} \left(\gamma^k - \overline{\pi} \cdot \mu_k\right)}{(g+h) \cdot \phi(\kappa^*) \cdot \sqrt{\sum_{k \in K'} \sigma_k^2}} \leq \frac{\sqrt{\sum_{k \in K^*(\overline{\pi}) \cup K'} \sigma_k^2} - \sqrt{\sum_{k \in K^*(\overline{\pi})} \sigma_k^2}}{\sqrt{\sum_{k \in K'} \sigma_k^2}} \tag{6.23}$$

Now we assume that for some $\overline{\pi}' > \overline{\pi}$, the new minimizing set $K^*(\overline{\pi}')$ is not a subset of $K^*(\overline{\pi}')$. Thus $K^*(\overline{\pi}')$ can be described as $K^*(\overline{\pi}') = K" \cup K'$, for some $K" \subseteq K^*(\overline{\pi})$. By optimality of $K^*(\overline{\pi}')$ the below inequality holds:

$$\frac{\sum_{k \in K'} \left(\gamma^k - \overline{\pi}' \cdot \mu_k\right)}{(g+h) \cdot \phi(\kappa^*) \cdot \sqrt{\sum_{k \in K'} \sigma_k^2}} \geq \frac{\sqrt{\sum_{k \in K" \cup K'} \sigma_k^2} - \sqrt{\sum_{k \in K"} \sigma_k^2}}{\sqrt{\sum_{k \in K'} \sigma_k^2}}$$

where $(\kappa')^*$ is so that $\Phi((\kappa')^*) = \dfrac{g - \overline{\pi}'}{g+h}$. By the concavity of the square-root function the following holds for non-empty K':

$$\sqrt{\sum_{k\in K"\cup K'}\sigma_k^2}-\sqrt{\sum_{k\in K"}\sigma_k^2}>\sqrt{\sum_{k\in K^*(\overline{\pi})\cup K'}\sigma_k^2}-\sqrt{\sum_{k\in K^*(\overline{\pi})}\sigma_k^2} \tag{6.24}$$

Thus it follows that

$$\frac{\sum_{k\in K'}\left(\gamma^k-\overline{\pi}'\cdot\mu_k\right)}{(g+h)\cdot\phi((\kappa')^*)\cdot\sqrt{\sum_{k\in K'}\sigma_k^2}}>\frac{\sum_{k\in K'}\left(\gamma^k-\overline{\pi}\cdot\mu_k\right)}{(g+h)\cdot\phi(\kappa^*)\cdot\sqrt{\sum_{k\in K'}\sigma_k^2}}$$

Since $\overline{\pi}'>\overline{\pi}$, the above can only be true if the numerator term increases faster than the denominator term in $\overline{\pi}$, i.e.

$$\frac{\partial}{\partial\overline{\pi}}\left(\sum_{k\in K'}\left(\gamma^k-\overline{\pi}\cdot\mu_k\right)\right)>\frac{\partial}{\partial\overline{\pi}}\left((g+h)\cdot\phi(\kappa^*)\cdot\sqrt{\sum_{k\in K'}\sigma_k^2}\right)$$

$$\Rightarrow-\sum_{k\in K'}\mu_k>(g+h)\cdot\sqrt{\sum_{k\in K'}\sigma_k^2}\cdot\frac{\partial\kappa^*}{\partial\overline{\pi}}\cdot\frac{\partial\phi(\cdot)}{\partial\kappa^*}$$

$$\Rightarrow-\sum_{k\in K'}\mu_k>(g+h)\cdot\sqrt{\sum_{k\in K'}\sigma_k^2}\cdot\left(\frac{-1}{(g+h)\cdot\phi(\kappa^*)}\right)\cdot(-\kappa^*\cdot\phi(\kappa^*))$$

$$\Rightarrow 0>\sum_{k\in K'}\mu_k+\kappa^*\cdot\sqrt{\sum_{k\in K'}\sigma_k^2}=Q^*(K')$$

The last equality follows from the definition of the news-vendor cost model for the subset K', where $Q^*(K')$ is the associated optimum build-level. Since $Q^*(K')\geq 0\ \forall\ K'\subseteq K$ by feasibility, this last inequality is contradictory for non-empty sets K'. Hence the assumption that $K^*(\overline{\pi}')$ is not a subset of $K^*(\overline{\pi})$ cannot hold, and thus $K^*(\overline{\pi}')\subseteq K^*(\overline{\pi})$. ■

Lemma 3: *For any given subset $\hat{K} \subseteq K$, the minimizing set $\hat{K}^*(\overline{\pi})$ over $\hat{K}$ always forms a subset of $K^*(\overline{\pi})$.*

Proof. As before, (6.23) holds by optimality of $K^*(\overline{\pi})$, and this re-written as:

$$\frac{\sum_{k\in K'}\left(\gamma^k - \overline{\pi}\cdot\mu_k\right)}{\phi(\kappa^*)\cdot(g+h)} \leq \sqrt{\sum_{k\in K^*(\overline{\pi})\cup K'}\sigma_k^2} - \sqrt{\sum_{k\in K^*(\overline{\pi})}\sigma_k^2}$$

for some non-empty $K' \subseteq \{\hat{K}\backslash K^*(\overline{\pi})\}$. Now we assume that $\hat{K}^*(\overline{\pi})$ is not fully contained in $K^*(\overline{\pi})$. As before we can then describe $\hat{K}^*(\overline{\pi})$ as $\hat{K}^*(\overline{\pi}) = K' \cup K$", where K" is some subset so that $K" \subseteq \{K^*(\overline{\pi}) \cap \hat{K}\}$. By optimality of $\hat{K}^*(\overline{\pi})$ the following holds:

$$\frac{\sum_{k\in K'}\left(\gamma^k - \overline{\pi}\cdot\mu_k\right)}{\phi(\kappa^*)\cdot(g+h)} \geq \sqrt{\sum_{k\in K"\cup K'}\sigma_k^2} - \sqrt{\sum_{k\in K"}\sigma_k^2}$$

Using (6.24) again, it is obvious that the last two inequalities are contradicting in the strict sense. Hence, the assumption that $\hat{K}^*(\overline{\pi}) \not\subseteq K^*(\overline{\pi})$ cannot be true, thus $\hat{K}^*(\overline{\pi}) \subseteq K^*(\overline{\pi})$. ■

Lemmas 2 and 3 imply that if K' is the set of orders not in the minimizing set $K^*(\overline{\pi})$, then K' can be simply discarded from consideration whenever $\overline{\pi}' > \overline{\pi}$, or when we solve the problem over some subset $\hat{K} \subseteq K$. Conversely, if K" belongs to the minimizing subset for some $\overline{\pi}' > \overline{\pi}$ or over some $\hat{K} \subseteq K$, then $K" \subseteq K^*(\overline{\pi})$. These observations are useful in reducing the search space when we apply procedure *Group* to solve the related problems.

6.2.3 Solving for the Minimum Reduced Cost

We now consider solving $\mathcal{SBP}$ for the minimum reduced cost Z_{i^*,θ^*}, where K_{i^*} and θ^* denote the optimal customer pool and build-type respectively. To accomplish this we will make use of the results in §6.2.1 and §6.2.2 embedded in a specialized branch-and-bound procedure. Define the problem at the *root node* of the branch-and bound tree as the relaxation of $\mathcal{SBP}$ by dropping constraints (6.21), i.e. the AVM restrictions. At the beginning of the solution procedure this root node problem is first solved. When AVM restrictions are absent, any build-type θ assigned to customer pool K_i is feasible. It is clear then from §6.2.1 that it suffices to select the minimum cost build-type θ^0 with cost $\overline{\pi}^0$, or equivalently the shortest path length on the network of problem $\mathcal{SPP}$. Solving for the corresponding subset K^0 can then be accomplished by simply applying procedure *Group* over the *candidate set* K, where the candidate set of a node is the set of all customers (variables) in the node problem.

Once K^0 is obtained, it is checked to see if any AVM restrictions are violated. if θ^0 is feasible for K^0, the optimal solution is found and the procedure terminates. Suppose (6.21) is now violated for the pair of suppliers (v, v'), where $v \in p$ and $v' \in p'$ for some pair of components p and p'. *Branching* then take place to create two new subproblems (or *descendent nodes*) based on arc (v, v'). The first subproblem enforces (v, v') to be active. This is equivalent to requiring (v, v') to be used in the shortest path network. All other arcs feeding nodes n_v $\forall v \in p'$ are thus eliminated. Customers k that are incompatible with (v, v') are also eliminated

from the candidate set of the node, so that an AVM violation on (v, v') will not occur again. The second subproblem removes arc (v, v') from the network, so that again violations on (v, v') will never occur. Customers that are incompatible with all other remaining paths (build-types) are also eliminated.

At each descendent node, the problem is again solved by relaxing the remaining AVM restrictions, using procedure *Group* over its ground set of customers. The solution Z' at the node then forms a lower bound to all its descendent nodes. If the solution does not violate any AVM restrictions, the node is *fathomed*, and its solution forms an upper bound on the optimal solution of $\mathcal{SBP}$. The best upper bound obtained in the procedure is updated as the *incumbent* solution Z^*_{ub}. *Pruning* of the branch-and-bound tree takes place by fathoming nodes with lower bounds greater than the incumbent. Branching takes place at a node if the node cannot be fathomed. Upper bounds can also be generated using the current solution by applying some *repairing heuristics*. The procedure terminates when there are no more unfathomed nodes.

For our problems a *depth-first search* strategy is used. In implementation depth-first search requires less memory since only the nodes on the current path are stored. This contrasts with breadth-first search approaches,, where all of the tree that has so far been generated must be stored. The solution procedure is summarized below.

Procedure *Branch*

1. Initialize $Z^*_{u.b.} = 0$. Set the root node as the current node and proceed to Step 2.

2. Solve the problem at the current node using procedure *Group* to obtain Z'. The solution then falls in one of the two situations below:

   a. The solution does not violate AVM restrictions. Z' is an upper bound to the optimal solution of $\mathcal{SBP}$. Furthermore, if $Z' \leq Z^*_{u.b.}$ update the incumbent $Z^*_{u.b.} = Z'$. Prune all nodes in the tree with solutions greater than $Z^*_{u.b.}$. The node is fathomed. Proceed to Step 3.

   b. The solution violates some AVM restrictions. If $Z' < Z^*_{u.b.}$, proceed to Step 4. Otherwise if $Z' \geq Z^*_{u.b.}$, the node is fathomed and proceed to Step 3.

3. If the list unfathomed nodes is empty the procedure ends. Otherwise select the next unfathomed node and proceed to Step 2.

4. Select an arc (v, v') which is violated in the current solution. Create two descendent nodes: the first node must contain arc (v, v'), and the second must not contain (v, v'). Update the candidate sets in each subproblem accordingly. Store these two nodes into the list of unfathomed nodes. Proceed to Step 3.

In Step 2.b of the above procedure, we can apply a simple repairing heuristic on the current solution to obtain a feasible upper bound. This is accomplished by taking the solution K' and applying procedure *Path* to obtain a feasible build-type θ' for the customer pool. We then apply procedure *Group* by fixing θ' to obtain the optimal customer pool on this path.

In Step 4, the descendent nodes derive their candidate sets from their parent node. This derived candidate set is simply the customer pool K' in the solution of the parent node, since by applying Lemmas 2 and 3 it is known that $k \notin K'$ will not appear in the optimal solutions of the more constrained descendent nodes. The candidate sets associated with each node thus becomes smaller as we proceed deeper in the tree. This serves to reduce the solution space in procedure *Group* when solving each node.

6.3 Solving Problem $\mathcal{IBP}$

In this section we consider the solution of the mixed integer problem $\mathcal{IBP}$ by extending the column generation approach for $\mathcal{BP}$. Since $\mathcal{BP}$ is a linear relaxation of problem $\mathcal{IBP}$, it is unlikely that the optimal solution of $\mathcal{BP}$ will yield a proper set-partition in general. In fact, one can expect that each customer will appear in different customer pools in the relaxed solution, which is clearly infeasible. To repair this the integer restrictions in $\mathcal{IBP}$ can be enforced on the LP column generation solution, and a feasible solution can be obtained. However, it is well-known (see Vance[136]) that since only a small fraction of all the columns is available to the MIP, this approach may not produce a solution that is close to being the problem in which all possible customer pool-build type assignments are considered.

The branch-and-price method[13] can implicitly consider all possible customer pool-build-type assignments. Branch-and-price has been widely and successfully applied to large-scale problems like generalized assignment[121], crew scheduling[136]

and vehicle routing [40]. The essential difference between a branch-and-bound and branch-and-price method is that in the latter, the linear relaxation at each node in the tree is solved using the column generation method, hence allowing new columns to be added into the problem. However, devising a branch-and-price procedure presents some inherent difficulties (Applegren[4]). First, variable fixing in conventional branch-and-bound methods may destroy the structure of the pricing problem. Secondly, solving the LPs to optimality at each node may not be efficient. Different rules thus applies for managing the branch-and-price tree and is often problem-specific.

The rest of this section presents the essential ingredients for designing the branch-and-price solution method. §6.3.1 develops the special branching rules required for the problem, and §6.3.2 discusses various tools that are important in increasing the efficiency of the branch-and-price algorithm. These tools include: (a) a pricing problem heuristic used to obtain good (though not necessarily optimal) solutions of $\mathcal{SBP}$ quickly, (b) the computation of an initial upper bound, which can be used to prune inferior sections of the branch-and-price tree and (c) the computation of lower bounds at each branch-and-price node to counter the so-called 'tailing-off' effect, which will be discussed later.

6.3.1 The Branch-and-Price Scheme

In order to obtain a proper set-partition, we need a branching scheme that excludes the current fractional solution, validly partitions the solution space of the problem,

and does not complicate the pricing problem too much. The third requirement renders standard branching rules based on variable fixing ineffective, because fixing a variable to 0 corresponds to forbidding a certain solution to the pricing problem. Deeper down in the search tree this implies that a set of solutions to the pricing problem must be excluded, which in general is very complicated if not impossible.

We adopt a branching scheme proposed by Ryan and Foster[118] for pure partitioning problems, which is based on the following proposition:

Proposition 1. *If A is a $0-1$ matrix, and a basic solution $Ay = 1$ is fractional, i.e., at least one of the components of y is fractional, then there exists two rows r and s of the master problem such that*

$$0 < \sum_{i=1}^{|I|} A_{r,i} \cdot A_{s,i} \cdot y(i) < 1.$$

Proposition 1 implies a branching scheme based on identifying pairs of rows or elements (r, s) in which the above inequality holds. If no branching pair can be identified, then the solution to the master problem must be integral. Otherwise, branching is done by dividing the solution space into one set in which r and s appear together, and into another set in which they must appear separately. In our problem A is the constraint sub-matrix formed by the first $|K|$ rows in $\mathcal{IBP}$ i.e. (6.11), and y the vector of all columns Y_i, $i = 1 \cdots |I|$.

We apply the branching scheme to formulation $\mathcal{BP}$ rather than to $\mathcal{IBP}$ directly. Although we do not require $X_{i,\theta,j}$ or even $\sum_{\theta,j} X_{i,\theta,j}$ to be integral, this branching rule is appropriate since we are only concerned with obtaining proper set partitions in the solution. Thus in each iteration of the solution procedure, we scan for pairs

of customers k' and k'' that appear together in some customer pool and separately in another pool in the solution of $\mathcal{BP}$. Branching is then based on the pair (k', k''), so that the left branch requires all legitimate customer pools to contain k' and k'' together, and the right branch requires its solution to be such that k' and k'' never appear together in the same customer pool. Clearly a proper set-partition is formed when no such customer pairs can be identified.

We now consider the solution of the sub-problems on both branches. For the branch that enforces k' and k'' together in the solution, this can be easily achieved by considering the two customer demands together as a single demand, effectively reducing $|K|$ by one. The pricing problem SBP and the solution approach in §6.2.3 remains essentially unchanged.

On the other branch, the pricing problem becomes considerably more complicated. The generated columns must now not only respect the AVM restrictions, but also in general a list of customer pool restrictions imposed on the node. We account for the restrictions by embedding them in the branch and bound procedure described in §6.2.3. This implies that in Step 2 of procedure *Branch* the solution arising from procedure *Group* may violate either the AVM restrictions, the customer pool restrictions or both. Branching can then be done based on either class of violations. Suppose k' and k'' appears together in the solution. Branching based on this pair of elements would then require k' (or k'') to appear in the solution on one branch. The candidate set of customers on this branch is then updated by eliminating all other customers which appears in the list of customer pool restric-

tions for the current master problem. On the other branch, k' (or k'', if k'' was chosen to remain in the first branch) is simply eliminated from its candidate set of customers.

6.3.2 LP Solution, Termination and Bounds

The computationally most intensive component of a branch-and-price algorithm is the solution of the linear programs, which includes the solution of many pricing problems. Thus, it is imperative that the solution of these linear programs are efficient in order for the branch-and-price algorithm to be efficient. Further, the use of strong upper and lower bounds can help to effectively prune branches in the branch-and-price tree, thus reducing the search space and effort considerably. Below we look at some measures related to implementing the branch-and-price method effectively.

(a) Pricing Problem Heuristic

In each iteration of the column generation procedure, the pricing problem is solved to generate new columns with negative reduced costs. Because any columns that satisfy this criteria will suffice, and since procedure *Branch* is computationally expensive because of the tree search, we simply apply a greedy construction heuristic to generate columns as long as such columns can be found. Procedure *Branch* is called only when the heuristic fails to locate any columns. The heuristic we implement constructs customer pools K_i in a greedy manner. The construction

procedure is as follows.

Heuristic *Construct*

1. Initialize the *current pool* $K_i = \{\}$, i.e. an empty set. Initialize the *remaining pool* K' to be the full candidate pool of customers of the node.

2. Apply procedure *Path* to solve for the minimum reduced cost of each pool $\{K_i \cup k\}, \forall k \in K'$. Let $\{K_i \cup \hat{k}\}$ yield the lowest reduced cost z with shortest path solution β_{v^*} among all pools $\{K_i \cup k\}$, $\hat{k}$, $k \in K'$. If $z \leq 0$, proceed to Step 3. Otherwise proceed to step 4.

3. Generate entering columns where: $\alpha_k = 1 \ \forall k \in \{K_i \cup \hat{k}\}$, and $\beta_{v^*} = 1$.

4. Update $K_i := \{K_i \cup \hat{k}\}$, and $K' := \{K' \backslash \hat{k}\}$. If $|K'| > 0$ go to Step 2. Otherwise the procedure terminates.

The *Construct* heuristic calls procedure *Path* at most $|K|!$ times, and generates up to $|K|$ columns with negative reduced costs.

(b) Upper Bound Computation

A strong initial upper bound can be used to prune inferior sections of the branch-and-price tree, especially at the initial stages of the search. We implemented an iterative heuristic search procedure to construct feasible solutions requiring modest computation times. The heuristic essentially constructs customer pools and uses these to form feasible solutions according to some AVM requirements. Define the level of combined AVM restrictions $\eta(K_i)$ of pool K_i as the percentage of infeasible

build-types for the pool among all build-types. The *tolerance* level t, $0 \leq t \leq 1$ is simply defined as the maximum allowable level of $\eta(K_i)$. The heuristic *Search* is stated as follows, where the output of the algorithm is Z_{best}, the best upper bound obtained.

Heuristic *Search*

1. Set *tolerance* $t = t_{min}$ and $Z_{best} = \infty$. Initialize also the set of customer pools of a feasible solution $Sol. := \{\}$. Set a desired N, the number of candidates evaluated per iteration.

2. If $t \leq t_{max}$, Initialize the counter $n = 0$ and go to Step 3. Otherwise the procedure terminates.

3. Initialize pool $K' := \{\}$, and the remaining set $\overline{K} := K$.

4. Randomly select $k \in \overline{K}$. If the AVM restriction level $\eta(k \cup K') < t$ then update $K' := K' \cup k$, $\overline{K} := \{\overline{K} \backslash k\}$ and repeat step 4. Otherwise go to Step 5.

5. Update the solution set $Sol. := \{Sol. \cup K'\}$. If $\overline{K} \neq \{\}$ reset $K' = \{\}$ and go to Step 4. Otherwise go to Step 6.

6. Compute $Z(Sol.)$, the optimal solution associated with set $Sol.$. If $Z(Sol.) \leq Z_{best}$ update $Z_{best} = Z(Sol.)$. Update $n := n + 1$. If $n < N$ go to Step 3. Otherwise go to Step 6.

7. Update $t = t+\Delta$, where Δ is some fixed step-size increment. Reset $Sol. := \{\}$. Go to Step 2.

In the above procedure, t_{min} and t_{max} are some pre-determined minimum and maximum tolerance levels of any customer pool K_i. In each iteration the heuristic confines the customer pools constituting the solution $Sol.$ to be of a certain η level. The solution $Z(Sol.)$ for the set $Sol.$ is computed by solving a restricted version of $\mathcal{BP}$, whereby only the columns associated with the customer pools in $Sol.$ are allowed in the solution. Heuristic *Search* then evaluates such solutions over the range of $t_{min} \leq \eta \leq t_{max}$, and returns the best upper bound obtained.

(c) Lower Bound Computation

Another potential cause of excessive computational effort in the solution the linear program is the well-known 'tailing-off effect' of the column generation process, i.e. a large number of pricing iterations is required to prove LP optimality. In implementation, the most common response is then usually some 'cut-off' rule of the form "stop if the change in the objective function does not exceed $x\%$ over p consecutive iterations"[51] . However, as Farley[42] noted, such a rule suffers from the danger of stopping the algorithm at a *stall point*. Further, in the branch-and-price scheme, this may cause the node to be prematurely pruned, thus incurring the risk of losing some potentially good or even optimal solutions.

Recall that at each node of the branch-and-price tree, the linear relaxation forms a lower bound to the solution of all its descendent nodes. To control the size of the

tree, it is best to work with strong lower bounds. However, the method will work with any bound. The tradeoff is thus between the computational effort required to obtain strong bounds and evaluating a smaller search tree. Barnhart[13] noted that a way to exploit this tradeoff is by choosing to prematurely terminate the column generation process and work with lower bounds on the final LP value, rather than solving the LP to optimality. For our problem we use a slight modification of the method proposed by Lasdon[87] to compute the lower bound. Multiplying (6.16) by the multipliers λ_k and (6.17) by the multipliers π_v, and subtracting from the objective function of $\mathcal{BP}$ (6.15) yields:

$$Z - \sum_{p\in P}\sum_{v\in V_p} \pi_v \cdot m_v - \sum_{k\in K} \lambda_k = \sum_{i,\theta,j} x_{i,\theta,j} \cdot \left(C_{i,\theta,j} - \sum_{k\in K} \alpha_i^k \cdot \lambda_k - \sum_{p\in P}\sum_{v\in V_p} \beta_\theta^v \pi_v \right)$$

The term in the parentheses on the right of the above is simply the reduced cost of $x_{i,\theta,j}$. Replacing this term with the minimum reduced cost $Z^*_{r.c.}$ (with respect to the current multiplier set) it is clear that:

$$Z - \sum_{p\in P}\sum_{v\in V_p} \pi_v \cdot m_v - \sum_{k\in K} \lambda_k \geq \sum_{i,\theta,j} x_{i,\theta,j} \cdot (Z^*_{r.c.})$$

$$Z - \sum_{p\in P}\sum_{v\in V_p} \pi_v \cdot m_v - \sum_{k\in K} \lambda_k \geq Z^*_{r.c.} \cdot \sum_{i,\theta,j} x_{i,\theta,j}$$

Further, the cardinality constraint: $\sum_{i,\theta,j} x_{i,\theta,j} \leq |K|$ holds in the optimum solution of BP by considering the aggregation of all the constraints in (6.16). We can then establish that:

$$Z \geq \sum_{p\in P}\sum_{v\in V_p} \pi_v \cdot m_v + \sum_{k\in K} \lambda_k + Z^*_{r.c.} \cdot |K| = Z_L$$

where Z_L is our required bound. Since the above holds for all values of Z in $\mathcal{BP}$, it holds for the optimum solution Z^*, and thus $\mathcal{Z}_\mathcal{L}$ is a valid lower bound.

6.4 Computational Results

For our computational experiments we generated twelve random problem instances from the HDD scenario. The HDD consists of three critical components i.e. $|P| = 3$, the HSA, disc and the PCB sub-assemblies. Table 6.1 summarizes the problem parameters in each of the instances, where $|K|$, $|V|$ indicate the number of customers and vendors per component respectively. The column labeled η denotes the total number of AVM restrictions (infeasible build-types) per customer as a percentage of the total number of build-types. *Cols* and *Rows* are the number of columns and rows in the formulation $\mathcal{BP}$ respectively.

Table 6.1: Problem Instances For Hard-Disk Drive Build-Planning Problem

instance	$\|K\|$	$\|V\|$	$\eta(\%)$	Cols	Rows
1	18	10	2.73	262144000	48
2	15	15	1.25	110592000	60
3	35	5	6.67	4.29497×10^{12}	50
4	35	10	1.82	3.43597×10^{13}	65
5	15	10	6.36	32768000	45
6	15	10	2.73	32768000	45
7	30	5	6.67	1.34218×10^{11}	45
8	30	10	1.82	1.07374×10^{12}	60
9	20	10	1.82	1048576000	50
10	40	8	5.56	5.6295×10^{14}	64
11	40	5	6.67	1.37439×10^{14}	55
12	50	10	2.73	1.1259^{18}	80

The solution algorithms in our implementation were all coded using C++. The CPLEX 8.0 LP and network callable libraries were invoked to solve the decomposed sub-problems in the column generation scheme. The computations were performed on a Pentium IV, 2.4 GHz PC with 512 MB RAM. Table 6.2 presents the computational results obtained. The columns *No-pooling*, *All-pooling* and *Search* tabulates upper bounds obtained by three different heuristics that were implemented. All upper bounds obtained are presented as ratios to the optimal solution obtained. In instances where the optimal solution is not found, the ratio is taken with respect to the tightest lower bound obtained in the branch-and-price tree. In the *No-pooling* heuristic, the only valid customer pools are the individual customer demands $k \in K$. Thus the build-plans allocates resources to individual customers prior to demand realization, and no risk-pooling opportunities are exploited. On the other hand, in the *All-pooling* heuristic, all the customer demands are always grouped together as a single customer pool. Heuristic *Search* follows from §6.3.2. The columns *B&P (3600s)* and *B&P (7200s)* display the ratios after 3600 and 7200 seconds of executing the branch-and-price code respectively. A value of '1' indicates that the optimum solution is achieved before the stipulated computation period. Similarly, the column *B&P* either displays that the optimum solution is achieved (i.e. value '1') or indicates the ratio of the upper bound to the tightest lower bound found, taken at 10,000s.

Table 6.2: Computational Results For Hard-Disk Drive Build-Planning Problem

instance	No-pooling	All-pooling	Search	B&P(3600s)	B&P(7200s)	B&P
1	1.15	11.65	1.09	1	1	1
2	4.04	771.41	1.13	1	1	1
3	1.10	7.17	1.09	1	1	1
4	253.07	4078.76	65.38	18.46	11.92	1
5	3.68	799.36	1.44	1	1	1
6	1.09	58.53	1.03	1	1	1
7	1.29	44.22	1.22	1	1	1
8	5.07	722.81	2.26	1.03	1	1
9	5.13	894.17	1.27	1	1	1
10	8.05	831.26	1.73	1.15	1.15	1.13
11	31.90	6006.13	13.38	2.66	2.09	1.09
12	13.06	1722.9	1.53	1.16	1.12	1.12

Table 6.3: CPU time (s) For Hard-Disk Drive Build-Planning Problem

instance	Col. Gen.(LP)	GLP	Search	B&P	Nodes Explored
1	3	NA	30	43	2
2	9	NA	36	580	142
3	2	NA	198	207	22
4	112	NA	591	9665	937
5	13	NA	102	3335	1004
6	1	NA	33	36	5
7	4	NA	111	126	20
8	53	NA	183	5999	506
9	60	NA	26	100	2
10	226	NA	33	10000	6400
11	48	NA	378	10000	4913
12	227	NA	914	10000	6394

Table 6.3 displays the CPU time (in seconds) requirements of the solution procedures. The columns headed *Col.Gen.(LP)* and *GLP* refer to the solution

times of the linear problem $\mathcal{BP}$ using the column generation method and using the CPLEX general-purpose solver directly. The columns *Search* and *B&P* refer to the computation times for problem $\mathcal{IBP}$ using the heuristic *Search* and the branch-and-price method respectively. *Nodes Explored* indicate the number of branch-and-price nodes that were evaluated during the search process.

Note that in Table 6.3, for all twelve of our problems, CPLEX was unable to solve even the linear problems $\mathcal{BP}$ given our computational resources. This was due to the large column count of the models, and as a result CPLEX was unable to load the entire problem into memory. On the other hand, the column generation algorithm was able to solve all the instances using only meagre amounts of computation times. For problem $\mathcal{IBP}$, nine out of the twelve problems were solved to completion by the branch-and-price method, using reasonably small amounts of CPU times. The remaining three problems were terminated at 10,000s, and yielded upper bounds which were reasonably tight compared to the lower bounds obtained.

The computational results in Table 6.2 indicates that the *All-pooling* heuristic generally produce inferior quality solutions (averaging 1,300 times the optimal costs). On the other hand the *No-pooling* heuristic performed consistently much better than the *All-pooling* heuristic (averaging 27 times the optimal costs). This verifies that the risk-pooling benefits diminishes rapidly in the presence of AVM restrictions. Although the AVM restriction levels η used for each customer is relatively low ($\eta \leq 7\%$), the combined restriction level of a customer pool tends to reduce the number of feasible build-types for the pool significantly. Lastly the

Search heuristic is shown to produce even better solution quality of about 7.7 times of the optimal costs on average, at the expense of relatively small amounts of CPU times. This indicates that although the AVM restrictions are present, there are still significant opportunities for improvement by exploiting the risk-pooling benefits.

6.5 Concluding Remarks

We have studied in this chapter the build-pack planning problem with stochastic demands. In this problem, the customer demand that occurs in the future is uncertain, and the objective is to develop a build plan that hedges against the future uncertainty. We assume that only the distribution the demands is known. A partitioning policy approach is adopted for the solution, where the customer demands are grouped as customer pools, and the mix of build-types is determined for each group prior to demand realization. When the demand is realized the grouping is not changed. The problem is then to find a good partition that results in the lowest production costs on expectation. For a given partition, the remaining problem of determining the build-type levels is then a news-vendor type problem with supply constraints. A MIP formulation is developed for this problem, and the column generation method is used to solve the LP relaxation of the formulation, where each decision variable determines a grouping and a build-type assignment. The pricing problem was solved using a specialized kind of branch-and-bound algorithm.

To obtain the optimal partition we need to solve the original MIP, as the LP solution in general does not yield proper set partitions. To obtain a feasible solu-

tion we can execute a normal branch-and-bound algorithm using the current set of columns available, but to achieve the optimal solution we need to be able to generate new columns at each branch-and-bound node. Thus we adopt the approach of branch-and-price, and in our scheme we branch on pairs of customers rather than on the original columns. This serves to preserve the column generation scheme, and only a slight modification is required in the pricing problem.

Our computational results demonstrated that grouping all the customers together in the build-planning phase can result in quite inferior solutions, because the effect of combining the AVM restrictions together may severely restrict the set of feasible build-types. On the other hand, ignoring the advantages of risk-pooling by planning for each customer individually also results in inferior solutions, although our computations demonstrate that they are generally better than the former case. The branch-and-price method can achieve much better quality solutions, if not optimal. The computational efficiency of the branch-and-price method depends on the implementation of many other tools that can help limit the search of the branch-and-price tree. In our implementation we used some tools to compute lower and upper solution bounds. Other tools may be incorporated to further improve the solution efficiency.

Chapter 7

Extensions to the Stochastic Model

In this chapter we consider two realistic extensions to the build-pack planning problem with stochastic demands. In particular we will show how the solution framework developed in Chapter 6 can be modified in these cases. In §7.1, we consider the problem when there are *homogenous lot requirements*, which may be an actual customer requirement on the batches of finished goods. To adapt the branch-and-price framework for this problem, a different set of branching rules is required to preserve the structure of the pricing problem. In §7.2, the normality assumptions of the customer demand distributions are relaxed, and the demands are allowed to be non-homogenous and following arbitrary discrete distributions. In this case, we present the method to evaluate the expected costs associated with the customer pools during the pricing problem solution.

7.1 Homogenous Lot Requirements

7.1.1 Problem Scenario

In this case, the demand of each customer can only be fulfilled using a single build-type. This requirement is commonly found in practice, where very frequently customers prefer each lot or a few lots of their finished products to be of the same make. The primary purpose of such a requirement is usually to improve lot traceability and is often part of a quality management program. In our case, the disk-drives assigned to fulfill a particular order must all have the same combination of component suppliers in the assembly.

7.1.2 Adapting the Branch-and-Price Solution Framework

There is a simple way to adapt the branch-and-price solution approach to account for the homogenous lot requirement. First note that the linear relaxation BP and its solution method remains essentially unchanged. In the solution of BP, if each customer demand is served using only at most one build-type θ, then the optimal solution is found. Otherwise, the branch-and-price procedure is applied. In this case however, a different branching scheme from that in §6.3 is required. We adopt a scheme analogous to those used in branch-and-price methods for integer multi-commodity network flow problems[14] . Here we scan for a customer order k which uses two vendors v and v' for the same component p, where v, $v' \in V_p$ in the relaxed solution. Branching then creates two new nodes based on the pair (k, v). The left

branch enforces k to use only v for component p, while the right branch disallows k to use v for component p. As a consequence of this branching scheme, it is interesting to note that on either branch, the only changes made to the new master problems are the AVMs of customer k. Procedure *Branch*(see §6.2.3) for solving the pricing problem thus remains essentially unchanged. The branch-and-price procedure terminates when no more such (k, v) pairs can be found.

We now go on to show that the branching scheme described is appropriate for solving $\mathcal{IBP}$. In particular, we show that proper set-partitions are formed when each customer is served by only one build-type. Suppose after solving the linear relaxation at some node of the branch-and-price tree, no (k, v) pairs can be identified, i.e. all customers are partitioned properly by a set of build-types. Denote the set $K_{\theta'}$ as the set of all customers that appears in the solution of the LP solution at this node that uses build-type θ'. In general there can be several columns with different customer pools K_i, where $K_i \subseteq K_{\theta'}$ appearing in the LP solution that are all assigned the build-type θ'. It can then be verified that in the LP solution, among all columns with K_i assigned to θ', only those with $K_i = K_{\theta'}$ will be active. This follows from the fact that the *marginal expected cost* of $K_{\theta'}$ is the least (most negative) among all subsets $K_i \subseteq K_{\theta'}$. Taking first derivatives with respect to Q on the news-vendor expected costs expression (6.6), the marginal expected cost $\dfrac{\partial C_i(\cdot)}{\partial Q}$ associated with K_i is:

$$\begin{aligned}\frac{\partial C_i(\cdot)}{\partial Q} &= \frac{\partial}{\partial Q}\left(h\cdot(Q-\mu(i))+\sigma(i)\cdot(g+h)\cdot G\left(\frac{Q-\mu(i)}{\sigma(i)}\right)\right)\\ &= h+(g+h)\cdot(\Phi(z)-1)\end{aligned}$$

where z denotes the standard normal random variable, $z=\dfrac{Q-\mu(i)}{\sigma(i)}$. The above result is well-established in the analysis of style goods (see Silver, Pyke and Peterson[126]). Since z is decreasing in both $\mu(i)$ and $\sigma(i)$, and $\Phi(z)$ diminishes for decreasing z, it follows that the minimum marginal costs at any given Q is associated with maximum $\mu(i)$ and $\sigma(i)$. Thus, the minimum marginal cost at any given Q is achieved by $K_{\theta'}$, since $\mu_{\theta'}=\sum_{i\in K_{\theta'}}\mu(i)\geq\mu(i)\ \forall K_i\subseteq K_{\theta'}$. Similarly $\sigma^2_{\theta'}=\sum_{i\in K_{\theta'}}\sigma_i^2\geq\sigma_i^2\ \forall K_i\subseteq K_{\theta'}$. Since we can consider all $k\in K_{\theta'}$ to be competing for a single resource θ', it then follows that a greedy procedure that always allocates resource θ' to customer pool $K_{\theta'}$ among all $K_i\subseteq K_{\theta'}$ is optimal.

7.2 Demands Following Arbitrary Distributions

In the basic model (Chapter 6) the customer demands were assumed to approximate normal probability distributions. In this section we consider demands following arbitrary discrete distributions. Using discrete distributions to consider stochasticity is also popularly known in the mathematical programming community as *scenario-based modeling*, where the uncertainty is modeled as a set of outcomes each with an associated probability of occurrence. Such a modeling approach allows the problem to be formulated as a large-scale linear program, which can in turn be solved using decomposition algorithms such as the L-shaped method. In essence, the L-shaped

method partitions the problem into two stages, where the first stage decisions are made prior to demand realization, and the second stage decisions are known as 'recourse' actions. The method is based on generating cuts in the first-stage problem to approximate the *expected recourse function*, which is known to be piece-wise linear convex in the first-stage decisions. Such an approach, however, is known to be computationally efficient only when the number of possible outcomes (the sample space) are of limited size. Furthermore, the application of such a modeling approach presents an additional difficulty in our build-pack problem. This stems from the fact that our first-stage decision are the build-plans, i.e. determining how much of which build-types θ to produce. Recall that a key characteristic of our problem is the high-proliferation of the build-types. It follows then that our first-stage problem necessarily contains a large number of columns in θ, $\forall \theta \in \vartheta$. To address this, column generation may be applied to solve the first stage without explicit enumeration of all the variables. However, the cuts generated in the second stage problem and augmented in the first stage are in θ, and this destroys the structure of the pricing problem.

In the following we show how our solution framework for a partitioning policy as developed in Chapter 6 can be modified to accommodate discrete demand distributions. Furthermore, we assume that customer demands are non-homogenous, i.e. each customer demand k is associated with a shortage penalty cost g_k. We first note that formulation BP is still valid in this case, since as mentioned in §6.1 piece-wise linear convexity of expected cost function $C_i(\cdot)$ holds for any discrete

distributions. The challenge is then in evaluating $C_i(\cdot)$ for any customer pool K_i when solving the pricing problem.

7.2.1 Computing the Expected Cost Function $C_i(\cdot)$

We consider how to evaluate the expected cost function $C_i(Q_i)$ for a given the customer pool K_i. First we re-cast the expected cost function as $C_i(Q_i) = \sum_{k \in K_i} g_k \cdot \mu_k - S_i(Q_i)$, where $S_i(\cdot)$ is defined as the *expected savings function*.

Let ω be an outcome or scenario, where $\omega \in \Omega$, Ω being the set of all possible outcomes, and ξ is the random vector of customer demands. $S_i(Q_i, \xi(\omega))$ then denotes the maximum savings associated with a build level of Q_i when the demand realization is $\xi(\omega)$. In the following we introduce the idea of a *recourse network*[110] which is useful for describing our problem. The recourse network in our case is a single-level tree, where a single incoming arc feeds a source node, and the source node emanates $|K_i| + 1$ outgoing arcs. The first $|K_i|$ outgoing arcs represents each customer demand k, with random arc capacities d_k and constant weights g_k. The $(|K_i| + 1)^{th}$ outgoing arc represents excess inventory, with weight $-h$ and unbounded from above. Q_i is pushed through the incoming arc and distributed into the outgoing arcs. The left of Figure 7.1 shows an example of the recourse network for three customer demands, where the pair of values $\{g_k, (d_k)\}$ indicates the weight and random capacity of each arc respectively. In the recourse tree we assume the demands (the out-going arcs) are indexed so that $g_1 \geq g_2 \geq \cdots \geq g_{|K_i|}$. The computation of $S_i(Q_i, \xi(\omega))$ is then equivalent to a deterministic problem of

maximizing the weighted flow on the recourse tree, where the random arc capacities are replaced with the realisations $d_k(\omega)$. A greedy algorithm allocating as much of Q_i as possible in sequence of rank to $d_1(\omega)$, $d_2(\omega)$ etc solves this deterministic problem.

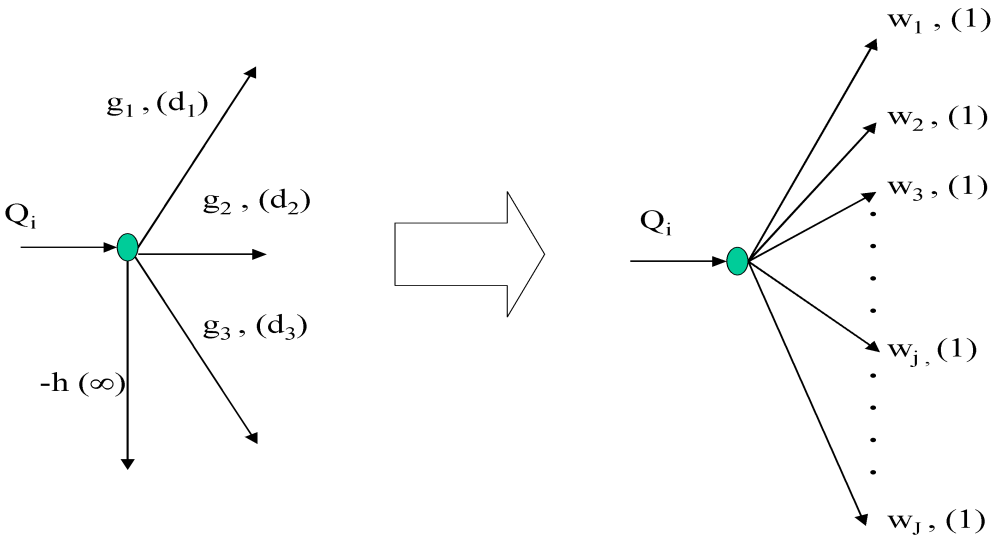

Figure 7.1: The Recourse Network and its Deterministic Equivalent Representation For Three Customers

Recall that in our problem we are interested in computing $C_i(Q_i^*)$ (or $S_i(Q_i^*)$), i.e. the minimum expected cost (maximum expected savings) for the optimal build-level Q_i^* over all demand realisations. We thus need some method to compute the expected costs $C_i(\cdot)$ as Q_i varies. Let the expected saving over all realizations be $S_i(Q_i) = E_\xi S_i(Q_i, \xi(\omega))$ for some Q_i. It turns out that the greedy algorithm for computing the maximum total weighted flow in the deterministic network flow problem can be easily adapted for the stochastic version of the problem. Powell[110]

suggests a procedure for computing the expected savings parametrically as a function of Q_i using *incremental flows* through the recourse network. A major step in this procedure is to compute the probability that a particular unit of flow entering the network uses a particular path. For customer pool K_i, let

$f_i(j,n)$ = Probability that the j^{th} unit of flow entering the source node takes path n

$s_i(j)$ = Expected marginal saving for the j^{th} unit of flow entering the source node

Then the expected recourse function is obtained by:

$$S_i(Q_i) = E_\omega S_i(Q_i, \xi(\omega)) = \sum_{j=1}^{Q_i} s_i(j) = \sum_{j=1}^{Q_i} \sum_{n=1}^{N_i} \tilde{g}_{i,n} \cdot f_i(j,n)$$

where $\tilde{g}_{i,n}$ denotes the weight on path n for the recourse tree associated with K_i. Thus $\tilde{g}_{i,k} = g_k$ for $k = 1 \cdots |K_i|$, $\tilde{g}_{i,|K_i+1|} = -h$, and $N_i = |K_i| + 1$ is the total number of paths. $f_i(j,n)$ is computed using the following equation:

$$f_i(j,n) = P\{c_{i,n} \geq j \cap c_{i,n-1} < j\} = P\{c_{i,n} \geq j\} - P\{c_{i,n-1} \geq j\} \tag{7.1}$$

where $c_{i,n}$ denotes the total capacity of the first n ranked paths. The condition in (7.1) says that the j^{th} unit of flow is on path n if and only if the first $n-1$ ranked paths have total capacity of less than j and the first n ranked paths have a total capacity of at least j.

Thus, given the customer pool K_i, the functional form of $S_i(Q_i)$ can be found by repeatedly applying (7.1), increasing a unit of flow of Q_i each time until $S_i(Q_i)$ reaches its maximum value. Since S_i is piece-wise linear concave in Q_i for discrete

distributions, we can represent $S_i(Q_i)$ as the maximum weighted flow solution through a set of deterministic arcs a_j, $j = 1 \cdots J(i)$ with unit capacity each and weights $w_j = s_i(j)$, in a single-level tree structure with Q_i as the input, and $J(i)$ such that $s_i(J(i)) \geq 0$ and $s_i(J(i)+1) < 0$. The right of Figure 7.1 illustrates the example of the deterministic equivalent representation corresponding to the three-customer example. The cumulative capacities of these arcs correspond to the 'breakpoints' in the piece-wise linear function and the weights on the arcs correspond to the marginal contribution to the expected savings function. The optimal input level Q_i^* corresponds to the maximum attainable expected savings, i.e. $S_i(Q_i^*)$.

7.2.2 Solving the Pricing Problem

We first consider solving the pricing problem $\mathcal{SBP}$ for a fixed customer pool K_i. Re-writing (6.19) in terms of $S_i(Q_i)$, the reduced cost expression becomes:

$$Z_i = \sum_{k \in K_i} g_k \cdot \mu_k - S_i(Q_i) - \sum_{k \in K} \gamma^k \cdot \alpha_i^k - \sum_{p \in P} \sum_{v \in V^p} \pi^v \cdot \beta^v \cdot Q_i$$

where Z_i as before denotes the reduced cost for the customer given K_i. Define x_j as the flow level on arc a_j in the deterministic single-level tree representation of $C_i(Q_i)$, $j = 1 \cdots J(i)$. From the preceding discussion, $S_i(Q_i)$ can be described as the sum of weighted flows in this tree network, i.e. $S_i(Q_i) = \sum_{j=1}^{J(i)} w_j \cdot x_j$, where the total flow $\sum_{j=1}^{J(i)} x_j = Q_i$. The pricing problem then becomes:

$$\min_{\beta_v, x_j, Q_i} Z_i = \sum_{k \in K_i} g_k \cdot \mu_k - \sum_{k \in K} \gamma^k \cdot \alpha_i^k - \sum_{j=1}^{J(i)} w_j \cdot x_j + \sum_{p \in P} \sum_{v \in V^p} (-\pi^v) \cdot \beta^v \cdot Q_i$$

subject to:

$$\sum_{j=1}^{J(i)} x_j = Q_i, \text{ and (6.20) to (6.22).}$$

Note that for a fixed K_i, the first two terms in the above expression for Z_i are unchanging. The third term decreases Z_i by w_j per unit flow on arc a_j in the deterministic single level tree. The last term penalizes the objective by increasing Z_i by $-\pi_v$ per unit flow on arc β_v in the shortest path network of problem $\mathcal{SPP}$. It follows then that the overall problem is equivalent to maximizing the *weighted flow* on the cascaded network as shown in Figure 7.2, with $\sum_{j=1}^{J(i)} x_j = Q_i$ acting as the flow-balance condition between the individual networks.

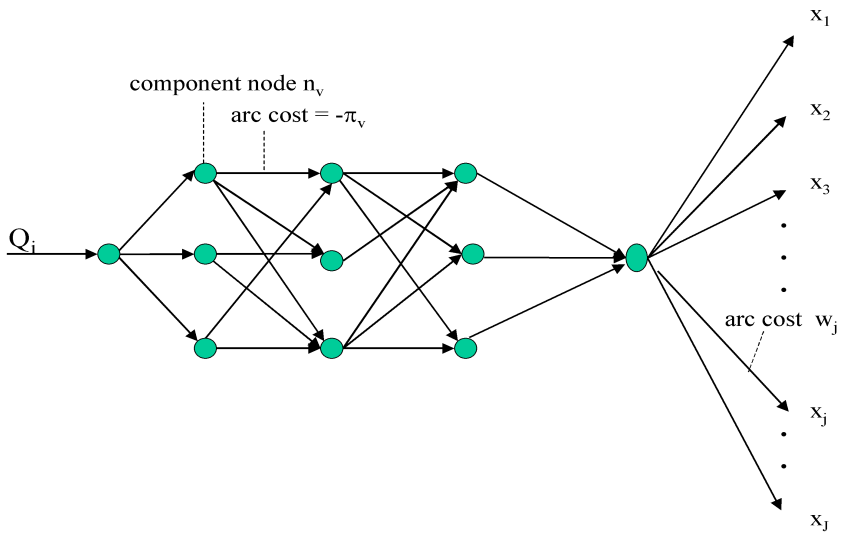

Figure 7.2: Cascaded Equivalent Network of Pricing Problem for Fixed K_i

The results established in §6.2.2 for solving $\mathcal{SBP}$ when build-type θ is fixed, however, does not readily extend in the case of discrete distributions. This is due to the fact that the expected costs function $Z_\theta(K_i)$ for a given θ (see §6.2.2) for arbitrary distributions in general may not exhibit submodularity. Solving the pricing problem to optimality is thus NP-hard in itself. We suggest devising heuristics or search procedures based on fixing K_i to generate entering columns in each iteration. Enumeration of all $K_i \in K$ is only performed to prove optimality when the heuristic fails to locate any negative reduced cost columns.

7.3 Concluding Remarks

In this chapter we considered two extensions of the build-pack planning problem with stochastic demands that allows the branch-and-price solution framework developed in Chapter 6 to be adapted. The first extension imposes a special requirement on the problem to allow only one build-type to be used for a particular customer's order. In the branching scheme presented in Chapter 6, the optimal solution produces a proper set partition of customer pools, but allows a mix of build-types to be used for each pool. In this scenario only one build-type can be used for each customer pool. We adopted a branching rule that exploits the network structure of the AVM restrictions. This rule is similar to those used in integer multi-commodity network flow problems, where sometimes each commodity is restricted to use only one path, i.e. shipments cannot be 'split'. In our case each path corresponds to a build-type, and the constituent arcs corresponds to the

product components of the build-type. The advantage of such a branching rule is that the structure of the pricing problem that was developed in Chapter 6 is virtually unchanged, and only the AVM information needs to be updated in each node.

In the second extension, we relaxed the distributional assumption of normality and considered a scenario-based framework which is more common in stochastic planning literature. In this case customer demands are independent and consists of a finite number of possible outcomes. The large set of all possible outcomes owing to the combinations renders the SLP model very cumbersome.

When the demands are arbitrarily distributed, the expected costs function generally cannot be given explicitly as in the case of the news-vendor problem with normally distributed demands. Typically the L-shaped method is then used to solve the problem by generating cuts to approximate the expected costs function, which is piece-wise linear in the decision variables. In our approach we compute these piece-wise linear functions parametrically for given customer pools. This is done by systematically varying the build-level decision variables as inputs and observing the expected costs as output. Because of its piece-wise linearity and convexity, the costs function can be modeled as a single-level tree of arcs, or a recourse network, with unit capacity bounds serving as 'breakpoints' in the piece-wise linear function.

In this manner, our branch-and-price solution framework can still be adopted. In the case when we solve the pricing problem for a given customer pool, the problem was shown to be a network flow problem by cascading the AVM network and

the recourse network. However, the pricing problem for the case of a given build-type cannot be solved easily, since the sub-modularity property is not necessarily preserved. Thus, procedure *Group* cannot guarantee the optimal solution in this case. Further research can be considered to identify the conditions in which sub-modularity holds, or development of other algorithms to solve the pricing problem efficiently.

Chapter 8

Conclusion and Future Research

Manufacturers competing in a thin-margin, commoditized market often offer their customers product flexibility options as a competitive advantage to defend their profit margins. These incentives however usually add complications to the basic production planning process. In this work we have considered the impact of one such complication known as the approved vendor matrix, based on the scenario of an actual manufacturer of hard disk drives. We define this class of problems the Build-Pack PPS problems, because there are two types of planning decisions i.e. what to build, and who to pack for. In particular we have studied two important problems: the multi-period build-pack scheduling problem, and the build-pack planning problem with stochastic demands.

For the multi-period build-pack scheduling problem, a mathematical model of the problem as an E/T production planning problem was developed. The model is then solved using the column generation method by exploiting special structural

properties in the pricing problem. Alternative formulations were also considered. We have shown that the same problem can be formulated as a multicommodity network flow (MCF) problem, hence enabling solution techniques for MCF problems to be applied to our problem.

An extension of the multi-period build-pack scheduling problem was then introduced, whereby the maximum number of work-cell set-ups in each production period are limited. We modified our E/T production planning formulation into a mixed-integer program to accommodate this class of restrictions. We then developed some simple linear programming rounding procedures to obtain good feasible solutions to the problem. Possible future research opportunities may include the devising of solution procedures for the Build-Pack problem with these set-up limitations. In our solution of the E/T planning model in Chapter 3, we have applied the column generation technique, and an extension to solve the build-pack problem with integer restrictions would be applying the branch-and-price method. This allows the generation of new columns deploying branch-and-bound simultaneously. However, the major challenge of the branch-and-price approach is in devising appropriate *branching rules*, so that the structure of the pricing problem would not be destroyed. This would require further investigation.

For the multi-period scheduling problem we have considered a fixed rather than rolling planning horizon in our model. This was adapted from the practice of the hard-disk drive manufacturer, which faces demand in weekly buckets. At the beginning of each work week a new schedule is generated to fulfill the week's demand.

As the work week starts, demand is actualized and new information arrives. Re-scheduling takes place at the end of each period to account for the changes, and the production plan for the rest of the week is updated. The planning horizon thus 'shrinks' as the week progresses, and using a rolling horizon model here does not seem plausible. In actual implementation the value of our work is to provide the initial schedule to guide the production planners at the beginning of the week. We offer rolling horizon considerations as a possible future extension for this work.

The next problem we studied is the build-pack planning problem with stochastic demands. We developed a realistic medium-term planning model which accounts for the AVM, and also the future uncertainty of the customer demands. Our modeling approach and solution scheme allows for a large number of random outcomes to be considered, which usually proves difficult for general-purpose stochastic linear programming methods to handle. The column generation method is used for solving the linear relaxation of our model. This is necessary due to the large number of columns in our model, arising from the large number of partitions combined with the high build-type proliferation of the problem. The branch-and-price solution procedure is then used to solve our model, and computational experiments shows that the method is capable of achieving good solutions in reasonable computation time.

Possible extensions for the stochastic problem include the study of the structural properties of the pricing problem when demands follow arbitrary distributions. As was mentioned in Chapter 7, the applicability of procedure to solve the pricing

problem hinges on the property that the problem is submodular when the build-type is fixed. This is shown to be valid for the normal distribution. It may then be of interest to consider the class of distributions that are valid for this property to hold true. On the other hand, we believe there is much room for study of more efficient heuristic methods (if not optimal methods) to solve the pricing problem under arbitrary distributional assumptions.

For the stochastic problems, we have employed the use of what is known as a 'two-stage' model. Such analysis assimilates production-planning of style-goods (with resource constraints), and is appropriate for medium term study of items facing rapidly shrinking product life-cycles. Another suitable application other than HDD may be the product-mix planning in a semiconductor foundry, since the wafer technology product-mix is known to change rapidly according to customer demands. However, if shorter-term production planning and scheduling is of interest, a multi-period model may be more appropriate, as the periodic availability of resources may present 'shifting-bottlenecks' that cannot be detected by the simpler two-stage model, which considers only total availability of the resources over the planning horizon. Adapting the partitioning-policy approach for a multi-period setting presents several challenges and issues, for example, whether customer pools should change from period to period, and if it does, how to assign end-of-period inventories to the new customer pools. We offer the multi-period extension of the partitioning-policy as an avenue of future research.

Bibliography

[1] P. Ahlstrom, R. Westbrook, "Implications of mass customization for operations management," *International Journal of Operations and Production Management* (v19, n3, 1999) pp 262-275.

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

[3] D. Alford, P. Sacket, and G. Nelder,"Mass-customisation: an automotive perspective," *International Journal of Production Economics*, (v65, n1, 2000) pp99-110.

[4] L.H. Appelgren, "A Column Generation Algorithm for a Ship Scheduling Problem," *Transportation Science* (v3, n1, 1969) pp53-68.

[5] S. Axsater, "Aggregation of product data for hierarchical production planning," *Operations Research* (v29, n4, 1981) pp744-756.

[6] S. Axsater, "On the feasibility of aggregate production plans," *Operations Research* (v34, n5, 1986) pp796-800.

[7] R.P. Bagozzi, *Principles of Marketing Management*. Science Research Associates, Chicago, IL, 1986.

[8] K.R. Baker, M.J. Magazine, and H.L.W. Nuttle, "The effect of commonality on safety stocks in a simple inventory model," *Management Science* (v32, n8, 1986) pp 982-988.

[9] K.R. Baker, "An experimental study of the effectiveness of rolling schedules in production planning," *Decision Science* (v8, n1, 1977) pp19-27.

[10] R. Balachandra, "Evaluating Modular Designs," *Decision Sciences Institute 2002 Proceedings*, San Diego: CA., Nov 22-26, 2002.

[11] R. Balachandra, "Modular Design and Mass Production," *Proceedings of the NEDSI Conference*, San Juan, PR, Mar 18-22, 2002.

[12] R. Balachandra, "Modular design and technological innovation: the case of hard disk drives," *Information Storage Industry Center. High-Technology Manufacturing*. Paper ISIC Report-2002-02.

[13] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance, "Branch-and Price: Column Generation For Solving Huge Integer Programs." Operations Research (v46, n3, 1998) pp316-329.

[14] C. Barnhart, C.A. Hane, and P.H. Vance, "Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems," *Operations Research* (v48, n2, 2000) pp318-326.

[15] E.M.L. Beale, J.J.H. Forrest, and C.J. Taylor, "Multi-time-period stochastic programming," in *Stochastic Programming*, M.A.H. Dempster (ed.), Academic Press, New York, 1980.

[16] J.F. Benders, "Partitioning Procedures for Solving Mixed Variables Programming Problems," *Numerische Mathematik* (v4, n2, 1962) pp238-252.

[17] W. Berry, T. Vollman, and C. Whybark, "Master production scheduling: principles and practice," *APICS*, Washington D.C., 1979.

[18] W. Berry, "Lot-sizing procedures for requirements planning systems: A framework for analysis," *Production and Inventory Management*, (v13, n2, 1972).

[19] J.R. Biggs, C.K. Hahn, and P.A. Pinto, "Performance of lot-sizing rules in an MRP system with different operating conditions," *Academic Management Review* (v5, n1, 1980) pp 89-96.

[20] P.J. Billington, J.O. McClain and L.J. Thomas, "Mathematical programming approaches to capacity-constrained MRP systems: review, formulation and problem reduction," *Management Science* (v29, n10, 1983) pp1126-1141.

[21] P.J. Billington, J.O. McClain and L.J. Thomas, "The interaction of lead-time determination, lot-size decisions and capacity planning in MRP systems," *Proceedings of the Eleventh Annual Meeting of the American Institute of Decision Sciences* (1979).

[22] G.T. Bishop, "On a problem of production scheduling," *Operations Research* (v5, n6, 1957) pp717-743.

[23] G.R. Bitran, and A.C. Hax, "On the design of hierarchical production planning systems," *Decisions Science* (v8, n1, 1977) pp28-55.

[24] G.R. Bitran, E.A. Haas, H. Matsuo "Production Planning of Style Goods with high set-up costs and forecast revisions," *Operations Research* (v34, n2, 1986) pp226-237.

[25] R.E. Bohn, "The Low-Profit Trap in Hard Disk Drives, and How to Get Out of It," *Insight* (March/April 2000) pp 6-9.

[26] E.H. Bowman, "Production scheduling by the transportation method of linear programming," *Operations Research* (v4, n1, 1956) pp100-103.

[27] M.D. Byrne, M.A. Bakir, "Production planning using a hybrid simulation-analytical approach," *International Journal of Production Economics*," (v59, n2, 1999), pp 305-311.

[28] J. Caie, and W. Maxwell, "Hierarchical machine load planning," in *Multi-level production-inventory systems: Theory and practice*, TIMS Studies in Management Sciences, L. Schwarz(ed.), North-Holland, Amsterdam, 1981.

[29] R.S. Chen, K. Y. Lu, S. C. Yu, H.W. Tzeng, and C. C. Chang, "A case study in the design of BTO/CTO shop floor control system," *Information and Management* (v41, n1, 2003) pp 25-37.

[30] S.C.K. Chu, "A Mathematical Programming Approach Towards Optimized Master Production Scheduling," *Production Economics* (v38, n2, 1995) pp 269-279.

[31] D. Collier, "A comparison of MRP lot-sizing methods considering capacity change costs," *Journal of Operations Management* (v1, n1, 1980) pp 23-29.

[32] D.A. Collier, "Aggregate safety stock levels and component part commonality," *Management Science* (v28, n11, 1982) pp 1296-1303.

[33] D.W. Cravens, and R.B. Woodruff *Marketing*, Addison-Wesley, Reading, MA, 1986.

[34] W. Crowston, M. Wagner, and J. Williams, "Economic lot-size determination in multi-stage assembly systems," *Management Science* (v19, n5, 1973) pp 517-528.

[35] M.A. Cusumano, "The limits of lean," *Sloan Management Review* (v35, n4, 1994) pp 27-32.

[36] G. Dantzig: *Linear Programming and Extensions* Princeton University Press (1963).

[37] G.B. Dantzig, and P. Wolfe, "Decomposition Principles for Linear Programs," *Operations Research* (v8, n1, 1960) pp101-111.

[38] S. Davis, "From future perfect: mass customizing," *Planning Review* (v17, n2, 1989) pp 16-21.

[39] R.F. Deckro, and J.E. Hebert "Goal programming approaches to solve linear decision rule based aggregate production planning models," *IIE Transactions* (v16, n4, 1984) pp 308-315.

[40] M.J. Desrochers and M. Solomon, "A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows," *Operations Research* (v40, n2, 1992) pp342-354.

[41] B.P. Dzielinski, and R.E. Gomory, "Optimal programming of lot sizes, inventory and labor allocation," *Management Science* (v11, n2, 1965) pp874-890.

[42] A.A. Farley, "A Note on Bounding a Class of Linear Programming Problems, Including Cutting Stock Problems," *Operations Research* (v38, n5, 1990) pp922-924.

[43] E. Feitzinger, H. Lee, "Mass customization at Hewlett-Packard: the power of postponement," *Harvard Business Review* (v75, n1, 1997) pp 116-121.

[44] H. Gabbay, "Multi-stage production planning," *Management Science* (v25, n11, 1979) pp 1138-1149.

[45] E.P. Garcia, and L.A. Swanson, "Scheduling production in an MRP enivronment when set ups are not significant." Paper in ORSA/TIMS Joint National Meeting (1989), Los Angeles.

[46] R. Garud, and A. Kumaraswamy, "Changing competitive dynamics in network industries: an exploration of Sun Microsystems' open systems strategy," *Strategic Management Journal*, (v14, n5, 1993) pp351-369.

[47] Y. Gerchak, and M. Henig "An inventory model with component commonality," *Operations Research Letters* (v5, n3, 1986) pp 157-160.

[48] M. Ghiassi, C. Spera, "Defining the Internet-based supply chain system for mass customized markets," *Computers and Industrial Engineering* (v45, n1, 2003) pp 17-41.

[49] J.B. Ghosh and C.E. Wells, "Scheduling to Minimize Weighted Earliness and Tardiness About a Common Due Date," *Computers and Operations Research* (v18, n6, 1991) pp465-475.

[50] J.B. Ghosh and C.E. Wells, "ON General Solution for a Class of Earliness/Tardiness Problems," *Computers and Operations Research* (v20, n2, , 1993) pp141-149.

[51] P.C. Gilmore and R.E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem," *Operations Research* (v9, n6, 1961) pp849-859.

[52] S.C. Graves, "Using Lagrangean Techniques to Solve Hierarchical Production Planning Problems," *Management Science* (v28, n3, 1982) pp260-275.

[53] H. Groenevelt, "The Just-in-Time System," *Handbooks in Operations Research and Management Science* (v4, Chap. 12, 1993) pp 629-670.

[54] N.G. Hall and M.E. Posner, "Earliness-tardiness Scheduling Problems: Weighted Deviation of Completion Times About a Common Due Date," *Operations Research* (v39, n5, ,1991) pp836-846.

[55] N.G. Hall, W. Kubiak and S.P. Sethi, "Earliness-tardiness Scheduling Problems: Deviations of Completion Times About a Restrictive Common Due Date," *Operations Research* (v39, n5,1991) pp847-856.

[56] R.W. Hall, "Graphical models for manpower planning," *International Journal of Production Research*, (v24, n5, 1986) pp1267-1282.

[57] Q. Hao, B.H. Soong, D.W. wanf, Z.H. Yang, "Earliness-tardiness production planning by JIT philosophy for job-lot manufacturing systems," *Production Planning and Control* (v9, n2, 1998)pp181-188.

[58] A.C. Hax and H.C. Meal, "Hierarchical Integration of Production Planning and Scheduling," *Studies in Management Sciences* (v1, Logistics, 1975, North-Holland, Amsterdam and American Elsevier) pp6-25.

[59] K. Heaghney, and T. Noden, "Enterprise profit optimization creates value through integrated decision-making," *Ascet* (v4, May 2002).

[60] R.S. Hiller, "Stochastic programming approximation methods, with applications to multi-stage production planning," Ph.D. disertaion, Operations Research Center, Massachusetts Institute of Technology.

[61] B. Hirsch, K.D. Thoben, and J. Hoheisel, "Requirements upon human competencies in globally distributed manufacturing," *Computers in Industry* (v36, n1, 1998) pp 49-54.

[62] H. Ho, and C. Lim "Spot the Early Bird," *China Logistics* (Oct 5, 2001).

[63] J.K. Ho, and W.A. McKenney, "Triangularity of the basis in linear programs for materials requirements planning," College of Business Administration, University of Tennessee (1987).

[64] K.H. Ho and R.P. Sundarraj, *DECOMP: An Implementation of Dantzig-Wolfe Decomposition for Linear Programming* (New York, Springer-Verlag, 1989).

[65] S.D. Hodges and P.G. Moore, "The product-mix problem under stochastic seasonal demand", *Management Science* (v17, n2, 1970) pp107-114.

[66] D. Hofman, "Achieving supply chain excellence," *Ascet* (v6, June 2004).

[67] A.J. Hoffman, and W. Jacobs, "Smooth patterns of production," *Management Science* (v1, n1, 1954) pp86-91.

[68] C.C. Holt, F. Modigliani., and H.A. Simon, "A linear decision rule for production and employment scheduling," *Management Science* (v2, n1, 1955) pp1-30.

[69] C.C. Holt, F. Modigliani., and J.F. Muth, "Derviation of a A linear decision rule for production and employment," *Management Science* (v2, n1, 1955) pp159-177.

[70] S.V. Hoover, and R.F. Perry, *Simulation: A Problem Solving Approach* (1989) Addison-Wesley, USA.

[71] S. Iwata, L. Fleischer, S. Fujishige, "A Combinatorial, strongly polynomial-time algorithm for minimizing submodular functions," *Proceedings of the* 32^{nd} *Annual ACM Symposium on Theory of Computing* (2000), pp97-106.

[72] S. Johnson, and G. Dantzig, "A A production smoothing problem," *Proceedings of the 2nd Symposium in Linear Programming*, pp151-176.

[73] A.P. Jones and R.M. Soland, "A branch and bound algorithm for multi-level fixed charge problems," *Management Science* (v16, n1, 1969) pp67-76.

[74] C.H. Jones, "Parametric production planning," *Management Science* (v13, n11, 1967) pp843-866.

[75] P. Kanchanasevee, G. Biswas, K. Kawamura, S. Tamura, "Contract-net based scheduling for holonic manufacturing systems," *Proceedings of the SPIE-The International Society for Optical Engineering* (n3203, 1999) pp 108-115.

[76] J.J. Kanet, 1986. "Towards a better understanding of lead times in MRP systems," *Journal of Operations Management* (v11, n3, 1986)pp305-315.

[77] E.P.C. Kao, "A multi-product dynamic lot-size model with individual and joint set-up costs," *Operations Research* (v27, n2, 1979) pp279-289.

[78] E.P.C. Kao and M. Queyranne, "Aggregation in a Two-Stage Stochastic Program for Manpower Planning in the Service Sector," Working Paper, Center for Health Management, University of Houston (1981).

[79] J.L. Kennington, "A Survey of Linear Cost Multicommodity Network Flows," *Operations Research* (v26, n2, 1978) pp209-236.

[80] D. Kira, M. Kusy and I. Rakita, , "A Stochastic Linear Programming Approach to Hierarchical Production Planning," *The Journal of the Operational Research Society* (v48, n2, 1997) pp207-211.

[81] S. Kotha, "From mass production to mass customization: the case of the National Industry Bicycle Company of Japan," *European Management Journal* (v14, n5, 1996) pp 442-450.

[82] S. Kotha, "Mass customization: Implementing the emerging paradigm for competitive advantage," *Strategic Management Journal* (v16, n1, 1995) pp 21-42.

[83] M. Lambrechet, and J. VanderEeken, "A capacity constrained single facility dynamic lot-size model," *European Journal of Operational Research* (v2, n2, 1978) pp132-136.

[84] R. Lau, "Mass customization: the next industrial revolution," *Industrial Management* (v37, n5, 1995) pp 18-19.

[85] G. Laurent, "A note on range programming: Introducing a satisfying ranging in an LP," *Management Science* (v22, n6, 1976) pp713-716.

[86] L.S. Lasdon and R.C. Terjung, "An Efficient Algorithm for Multi-Item Scheduling," *Operations Research* (v19, n4, 1971) pp946-970.

[87] L.S. Lasdon, *Optimization Theory for Large Systems* MacMillan, New York (1970).

[88] S.M. Lee, and L.J. Moore, "A practical approach to production scheduling," *Journal of Production and Inventory Management* (v15, n1, 1974) pp 79-92.

[89] Y. Li, D.W. Wang, and W.H. Ip ,"Earliness-tardiness production scheduling and planning, and solutions," *Production Planning and Control* (v9, n.3, 1988) 275-285.

[90] A.G. Lockett, and A.P. Muhlemann, "A problem of aggregate scheduling and application of goal programming," *International Journal of Production Research* (v16, n2, 1978) pp127-135.

[91] S. Love, "A facilities in series inventory model with nested schedules," *Management Science* (v18, n5, 1972) pp327-339.

[92] I. Lustig, "Optimization: Achieving Maximum ROI within the Supply Chain" *Ascet* (v1, 1999).

[93] A. S. Manne, "Programming of Economic Lot Sizes," *Management Science* (v4, n2, 1958) pp 1-22.

[94] B. H. Maskell, "Why MRP II Has Not Created World Class Manufacturing and Where Do We Go from Here?" *APICS-The Performance Advantage Magazine* (Sept. 1993)

[95] W. Maxwell and J. Muckstadt, "Coordination of production schedules with shipping schedules" in *Multi-level production-inventory systems: Theory and practice*, TIMS Studies in Management Sciences, L. Schwarz(ed.), North-Holland, Amsterdam, 1981.

[96] J.T. Meij, "Separable programming as a solution methodlogy for aggregate production planning," *International Journal of Production Research* (v18, n2, 1980) pp233-243.

[97] J.M. Mellichamp, and R.M. Love, "Production switching heuristics for the aggregate planning problem," *Management Science* (v24, n12, 1978) pp1242-1251.

[98] R. Metters, "Production planning with stochastic seasonal demand and capacitated production", *IIE Transactions* (v29, n11, 1997) pp1017-1029.

[99] J.V. Murphy and E.Sherman, "Supply Chain Planning Software Enables Revolutionary Change" *Global Logistics and Supply Chain Strategies* (April. 1998)

[100] R. Nellore, and R. Balachandra, "Factors Influencing Success in integrated Product Development Projects," *IEEE Transactions on Engineering Management* (v48, n2, 2001) pp164-174.

[101] New Business Systems Dept. (Maxtor Singapore), Production Planning and Control Dept. (Maxtor Singapore), and Industrial and Systems Engineering Dept. (National University of Singapore), "The Processes of Operational Production Planning in Maxtor Singapore," unpublished technical documentation, 2001.

[102] G.L. Nemhauser, and W.B. Widhelm, "A Modified Linear Program for Columnar Methods in Mathematical Programming," *Operations Research* (v19, n4, 1971) pp1051-1060.

[103] E.F.D. Newsom, "Multi-item Lot Size Scheduling by Heuristic," *Management Science* (v21. n10, 1975) pp1194-1203.

[104] P.J. O'Grady and M.D. Byrne, "A combined switching algorithm and linear decision rule approach to production planning," *International Journal of Production Research* (v24, 1986) pp285-296.

[105] M.D. Oliff, and E.E. Brunch, "Multi-product production scheduling at Owens-Corning Fiberglas," *Interfaces* (v15, n5, 1985) pp 25-34.

[106] J. Orlicky, "Materials requirements Planning," McGraw-Hill (1975), New York.

[107] M.G. Orrbeck, D.R. Schuette, and H.E. Thompson, "The effect of worker productivity on production smoothing," *Management Science* (v14, n6, 1968), pp332-342.

[108] R.J. Peters, K. Boskma and H.A.E. Kuper, "Stochastic Programming in production planning: a case with non-simple recourse," *Statistica Neerlandica 31* (v31, n1, 1977) pp113-126.

[109] J. Pine, B. Victor, and A. Boyton, "Making mass customization work," *Harvard Business Review* (v71, n5, 1993) pp 108-111.

[110] W.B. Powell and R.K.M. Cheung, "Network Recourse Decomposition Method for Dynamic Networks with Random Arc Capacities," *Networks* (v24, n7, 1994a)pp 161-175.

[111] M.E. Posner, and W. Szware, "A transportation type aggregate production model with backordering," *Management Science* (v29, n2, 1983), pp188-199.

[112] T.R. Rakes, L.S. Franz, and A.J. Wynne, "Aggregate production planning using chance-constrained goal programming," *International Journal of Production Research* (v22, n4 , 1984) pp 673-684.

[113] T.E. Ramsay Jr., "Integer programming approaches to capacitated concave cost production planning problems," unpublished Ph.D. thesis, georgia Institute of Technology, February 1980.

[114] J.B. ReVelle "Lean Manufacturing," in *Manufacturing handbook of best practices : an innovation, productivity, and quality focus* (Chap. 8, 2001), pp 203-226.

[115] L.P. Ritzman, L.J. Krajewski, W.L. Berry, S.H. Goodman, S.T. Hardy, and L.D. Vitt, eds. *Disaggregation problems in manufacturing and service organisations* Martinus Nijhoff, Boston, MA, 1979.

[116] R.T. Rockafella and R.J.B. Wets, "A Lagrangean finite generation Technique For solving Linear-quadratic Problems in Stochastic Programming," in A. Prekopa and R.J.B. Wets, *Stochastic Programming 1984* Mathematical Programming Study, North Holland (1985).

[117] D. Rogers, R. Plante, R. Wong, and J. Evans "Aggregation and disaggregation techniques and methodology in optimization," *Operations Research* (v39, n4, 1991) pp 553-582.

[118] D.M. Ryan and B.A. Foster, "An Integer Programming Approach to Scheduling," *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling* A. Wren (ed.), North-Holland, Amsterdam (1981) pp269-280.

[119] R. Sanchez, "Towards a Science of Strategic Product Design," paper presented at the Second International Product Development Management Conference on New Approaches to Development and Engineering, (May 30-31, 1994) Gotheburg, Sweden.

[120] R. Sanchez, and J.T. Mahoney "Modularity, flexibility and knowledge management in product and organization design," *Strategic Management Journal*, (v17, Winter Special Issue, 1996) pp 63-67.

[121] M.W.P. Savelsbergh, "A Branch-and-Price Algorithm for the Generalised Assignment Problem," *Operations Research* (v45, n6, 1997) pp 831-841.

[122] A. Schrijver, "A combinatorial algorithm minimizing submodular functions in strongly polynomial time," Preprint.

[123] A. Segerstedt, "A capacity-constrained multi-level inventory and production control problem," *International Journal of Production Economics* (v45, n3, 1996) pp449-461.

[124] Z. J. M. Shen, C. Coullard and M.S. Daskin, "A Joint Location-Inventory Model," *Transportation Science* (v37. n1, 2003) pp40-55.

[125] J. Shephard and L. Lapide, "Supply Chain Planning Optimization: Just the Facts," *Ascet* (v1, April 1999)

[126] E.A. Silver, D.F. Pyke and R. Peterson, *Inventory Management and Production Planning and Scheduling* John Wiley and Sons (1998).

[127] D. Simchi Levi, P. Kaminsky, E. Simchi-Levi, *Designing and Managing the Supply Chain: Concepts, Strategies, and Case Studies* McGrawHill (2000).

[128] K. Singhal, and V. Adlakha, "Cost and shortage trade-offs in aggregate production planning," *Decision Sciences* (v20, n1, 1989) pp 158-164.

[129] R.M.V. Slyke, J.B. Wets, "L-shaped Linear Programs With Applications to Optimal Control and Stochastic Programming", *SIAM Journal on Applied Mathematics* (v17, n4, 1969) pp638-663.

[130] R. Srinivasan, R. Jayaraman, J. Rappold, R. Roundy, and S. Tayur, "Procurement of common components in prescence of uncertainty," *IBM Technical Report* 1998.

[131] M.K. Starr, "Modular Production - A New Concept," *Harvard Business Review* (v43, n6, 1965) pp 131-142.

[132] J.M. Swaminathan and S.R. Tayur, "Managing Broader Product Lines Through Delayed Differentiation Using Vanilla Boxes," *Management Science* (v44, n12, 1998) pp161-172.

[133] J. Swaminathan, "Enabling customization using standard operations," *California Management Review* (v43, n3, 2001) pp 125-136.

[134] W.H. Taubert, "A search decision rule for the aggregate scheduling problem," *Management Science* (v14, n6, 1968) pp343-359.

[135] Valdero, "From planning to control: improving the high-tech supply chain," *Ascet* (v4, May 2002).

[136] P.H. Vance, "Crew Scheduling, Cutting Stock and Column Generation: Solving Huge Integer Programs," Ph.D Thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

[137] A. Veinott, "Minimum concave-cost solution of Leontief substitution models of multi-facility inventory systems," *Operations Research* (v17, n2, 1969) pp262-292.

[138] H. M. Wagner, and T.M. Whitin "Dynamic version of the economic lot size model," *Management Science* (v5, n1, 1958) pp89-96.

[139] W.E. Walker, "A heuristic adjacent extreme point algorithm for the fixed charge problem," *Management Science*, (v22, n5, 1976) pp587-596.

[140] D.W. Wang, "Earliness-Tardiness Production Planning Approaches for Manufacturing Systems," *Computers and Industrial Engineering* (v28, n3, 1995) pp425-436.

[141] D.W. Wang and W. WANG, "Earliness-tardiness production planning approaches with due-window for manufacturing systems," *Computers and Industrial Engineering*, (v34, n4, 1995) pp825-836.

[142] R.J.B. Wets, "Stochastic Programs With fixed Recourse: The Equivalent Deterministic Program," *SIAM Review* (v16, n3, 1974) pp309-339.

[143] A. Zahorik, J. Thomas, and W. W. Trigeiro, "Network programming models for production scheduling in multi-stage, multi-item capacitated systems," *Management Science* (v30, n3, 1984) pp 308-325.

[144] W. Zangwill, "A backlogging model and multi-echelon model of a dynamic economic lot-size production system - a network approach," *Management Science* (v15, n9, 1969) pp506-528.

[145] P.H. Zipkin, "Bounds on the effect of aggregating variables in linear programs," *Operations Research* (v28, n2, 1980) pp 403-418.