OPTIMAL LEVEL SCHEDULES FOR MIXED-MODEL, JUST-IN-TIME ASSEMBLY SYSTEMS

By

JULIAN SCOTT YEOMANS, B.ADMIN., B.SC.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Doctor of Philosophy

McMaster University

OPTIMAL LEVEL SCHEDULES FOR MIXED-MODEL JIT ASSEMBLY

DOCTOR OF PHILOSOPHY (1992) McMASTER UNIVERSITY

(Business Administration) Hamilton, Ontario

TITLE: Optimal Level Schedules for Mixed-Model, Just-In-Time Assembly Systems

AUTHOR: Julian Scott Yeomans, B.Admin. (University of Regina)
B.Sc. (University of Regina)

SUPERVISOR: Dr. George Steiner

NUMBER OF PAGES: xiii, 232

ABSTRACT

The usage problem which occurs in the scheduling of mixed-model assembly processes operating under Just-In-Time (JIT) methods is examined. A minimax objective function, which has not been considered previously for use with mixed-model JIT systems, is introduced to control these processes. A general integer programming model of the problem is developed with the goal being to determine optimal sequencing methods for various formulations of this model.

It is shown that the single-level, unweighted version of the model can be solved to optimum using an algorithm which is polynomial in the total product demand. A graph theoretic representation of the problem permits the calculation of bounds on the objective value. Of particular significance is the upper bound which demonstrates that a feasible sequence always exists for each copy of every product in which the actual level of production never deviates from the ideal level by more than 1 unit. Symmetries within the problem are shown to exist which substantially reduce the computational effort. Extensions to weighted single-level problems are made.

A dynamic programming (DP) algorithm is presented for optimizing the multi-level problem formulations. The time and space requirements of this DP are demonstrated. Tests of the DP's performance capabilities, characteristics and limitations are performed. As the growth of the DP's state space could severely restrict the problem size if all of the states are generated, a necessary screening method, employing simple heuristics, is used. The results of the testing indicate that the size of the problems which can be optimized is

constrained by the solution time. Conversely, the size of the problems that could be optimized never became constrained by the storage requirements since the simple heuristics acted as highly efficient screening mechanisms. The experimentation also uncovered certain inherent solution characteristics.

ACKNOWLEDGEMENTS

I would like like to express my deep gratitude to Dr. George Steiner for his supervision of this research. His time, effort, interest and thoroughness in this study exceeded by far what could be considered the regular 'call of duty' for a supervisor and has been greatly appreciated.

My thanks also go to the members of my supervisory committee, Dr. John Miltenburg and Dr. Tao Jiang, for their interest and involvement. Particular thanks must go to both Professor Miltenburg and Dr. Wieslaw Kubiak, of Memorial University, for permitting ready access to some of their unpublished research; not necessarily a common occurrence in the academic world.

I would like to thank my parents for their continuous support and encouragement over all of the years. They always wished that their son would become a doctor, but a Ph.D in Management Science...? I must also express my appreciation to my brother Mark, another aspiring doctor, for never letting the seriousness of research intefere with one's keen sense for the ridiculous.

Thanks go to Sebastian for digesting more preliminary draughts of this thesis than could be considered humanly possible. Finally I would like to thank my wife, Christine, for all of her love and support. I am especially grateful for her understanding (tolerance?) and for never questioning my sanity (at least openly) when ideas necessitated a transformation onto paper at the ungodly hour of three o'clock in the morning.

TABLE OF CONTENTS

CHAPTER 5 COMPUTATIONAL EFFICIENCIES FOR SINGLE LEVEL PROBLEMS

CHAPTER 6 MULTI-LEVEL OPTIMIZATION

LIST OF FIGURES

LIST OF TABLES

TWO LEVEL TESTING WITH D_1=1000

FOUR LEVEL TESTING WITH D_1=500

SAMPLE MULTI-LEVEL PROBLEM

CHAPTER 1 REVIEW OF JIT

1.1 Introduction to JIT

Notwithstanding the inherent controversies surrounding its definition, measurement and interpretation (Bailey & Hubert 1980; Caves et al. 1980) , productivity analysis is generally recognized as an effective tool for evaluating the past performance and for assessing the effectiveness (both positive and negative) of actions taken to improve efficiency (Eilon et al. 1976; Eilon 1962). It is widely viewed that efforts to achieve the highest possible productivity will, in themselves, reap economic benefits (e.g. see, Nelson 1981; Cosmetatos 1983). With the rapidly fluctuating economic landscape of the 1980's, many North American manufacturers have been obliged to adjust their approach to production in order to retain and regain their foothold in an increasingly more competitive global marketplace. The entrenched North American mindset of using the longest lead time for producing the largest lot size to obtain the lowest price would hold if one were basing price strictly in terms of manufacturing costs (e.g. see Chyr et al. 1990). However, where is the benefit in achieving these price reductions if the manufacturer is not producing exactly what the customers want at exactly the time when they want it? One innovative approach used in repetitive (& other) manufacturing industries to answer this question is the just-in-time (JIT) strategy (Hannah 1987).

Ways of measuring and evaluating the effectiveness of JIT systems are as plural and diverse (and also as controversial) as those for measuring the effectiveness in traditional manufacturing. In this study the focus will be on measuring the effectiveness of scheduling in a JIT system. This chapter will review the key features of JIT systems and will examine the applications and implementation of these concepts in North American companies. Comparisons to traditional manufacturing methods will be made and the need for the scheduling of these JIT systems will be reviewed. This study will examine the scheduling process in one particular type of JIT manufacturing environment; namely that of a mixed-model, JIT assembly process. This assembly process will be modelled in subsequent chapters and methods of optimization for various formulations of this model will be sought.

I.2 JIT Systems

The JIT approach was developed in post-Second World War Japan at the Toyota Motor Company by Taiichi Ohno, the former vice-president of manufacturing (Monden 1983). JIT emphasizes a continual process of removing waste and inefficiency from the production environment through high quality (Crosby 1984) and small lead times (Ohno 1988). The focus is one of solving production problems so that manufacturing operations become increasingly more efficient (Suri & Treville 1986). "Just-In-Time" refers to the actual production system whereby operations are activated just (and only) as they are needed. The JIT concept goes beyond the strict bounds of the production function and leads to more of a company-wide philosophy and way-of life. Companies using JIT treat production as an evolutionary operation (e.g. see Lambrecht & Decaluwe 1988); Bottlenecks to efficient production are

identified, focused upon and eliminated until new bottlenecks appear, thereby regenerating the improvement cycle.

The successful implementation of a JIT system necessitates a high degree of teamwork and cooperation by all employees of a company (Crawford 1990; Johnston et al. 1989; Oliver 1990). Fukuda (1983) describes how this team approach can be implemented. Wastes, such as scrap and rework, must be reduced and eliminated through the high quality of the product itself (Ebrahimpour 1985). Total Quality Control (TQC) is the systems approach to quality improvement within a company, in which all employees are responsible for the monitoring of product quality. TQC is a part of the JIT process (see Hendrick 1987) and its use within Japanese companies is discussed in Ishikawa (1985). The Japanese regard inventory as an "evil" (Wall Street Journal, April 7, 1982, p26; Huang et al. 1984) as it takes up space and ties up resources (Stevenson 1990, p624). Hall (1983) describes in detail how the JIT process, by producing only the necessary parts, in the necessary quantities, at the necessary times, results in very low levels of all types of inventory (i.e. raw materials, work-in-process and finished goods) which saves space (both in the warehouse and on the shop floor) and frees up resources which would normally be tied up in the idle inventory.

Monden (1981[a], 1981[b], 1981[c], 1981[d]), in a series of articles, outlined the key features of Toyota's JIT system. Most of these features are common to any JIT system (e.g. see Stevenson 1990, p626; Dilworth 1986, p354; Gaither 1987, p538) and will be summarized below.

I.2.1 Low Inventories

The most notable feature of JIT systems is the resultant low level of inventory. As indicated, all the types of inventory are reduced freeing up both space and resources. Production problems (i.e. poor quality, unreliable vendors, etc.) which might be hidden in the inventory of a 'traditional' manufacturing environment are exposed in the JIT system and may be corrected in the evolutionary approach taken in problem solving.

I.2.2 Stable and Level Production Rates

A JIT production system requires a uniform rate of production within the system (Monden, 1981[c]). Gaither (1987, p540) notes that Toyota, in its monthly production, tries to keep the same production schedule for every day of the month. Thus, '...if only a few of a particular [automobile] model were needed in a month, some would be assembled in each day of the month. If JIT is to work, stable and level production schedules are a must'.

Furthermore, the quantity of each part used in the assembly process per unit time should be maintained at as constant a rate as possible. There should be as little variability in the usage of each part by the process from one time period to the next (Miltenburg 1989). This is the most important goal of a JIT production system (Monden 1983) and Hall (1983) has referred to this as levelling or balancing the schedule.

I.2.3 Reduction of Lot Sizes

JIT systems require small lot sizes in the production process (South 1986). In order to produce these small lots (often of size 1) it is necessary that the changeover cost from one product to another (measured in time and other

resources) be negligible. These small lot sizes lead directly to reduced inventory levels throughout the factory. Fukuda (1983) describes the reduction of setup times for machines to the desired 'one-touch' [i.e. very rapid] setups (see also Spence & Porteus 1987).

To facilitate quick changeovers, JIT systems tend to arrange equipment to handle streams of parts and products with similar processing requirements. Monden (1983, p100) describes the 'U-turn format', for the arrangement of machines, where small groups of workers attend several machines arranged in the "U" pattern corresponding to the flow of parts within a particular machine group. This grouping of machines and the reduction of in-process inventory allows for smaller factories to be developed if JIT is employed. Workers are expected to be proficient in the operation of several machines and must be able to assist their fellow workers in maintaining the schedule should someone fall behind. This entails that workers be multi-functional and capable of ; handling their own setups, making minor adjustments to the machines in their charge, and being able to perform minor repairs. [This contrasts sharply to the strong opposition to the multi-tasking requirements by the manufacturing unions in North America (Inman & Mehra 1989).] Workers are responsible for checking the quality of their work and are expected to contribute to the problem solving process both for current problems and for those that may occur in the evolutionary process of the JIT system (Ishikawa 1985, p85 & p137).

I.2.4 Pull Systems

A facility operating under JIT uses what is known as a "pull" system (for a more complete description of push & pull systems see, Pyke & Cohen

1990; Detoni et al. 1988). In this system, work is moved from operation to operation only in response to demand from the next stage in the process. The control of this movement is the responsibility of the subsequent operation. Each workstation pulls the output from the preceding station only as it is needed. Output of the finished goods for the entire production facility is pulled by customer demand. Communication occurs backward through the system from station to station. Work moves "just in time" for the next operation and the flow of work is coordinated in such a way that the accumulation of excessive inventory between operations is avoided (Im & Schonberger 1988).

As production is not instantaneous some inventory must necessarily be present. By design, a workstation produces just enough output to meet the demand of the next station. Thus, either a workstation must communicate its need for the preceding stations output sufficiently in advance or (more commonly) there must be a small buffer stock between stations (So & Pinault 1988). When this buffer decreases to a prescribed level, a signal is sent to the preceding station to produce enough output to replenish the buffer supply. The size of this buffer is dependent upon the cycle time at the preceding work station. Production occurs only in response to the usage of the following station. Work is pulled only by the demand generated at the subsequent operation.

The communication of the demand can be achieved in a variety of ways. The most commonly used device is some variant of the kanban card system used at Toyota (the terms JIT production systems and kanban production systems are often interchangeable). When materials or work are required from the preceding station, a kanban card is sent authorizing the

moving or work for parts. No part or lot can be moved or worked on without the use of these cards. Monden(1981[b]) describes the use of kanban cards at Toyota to control their JIT process.

I.2.5 Quality

With a system which is geared to maintaining low inventory levels, a smooth flow of work and a pull system, work disruptions can cause significant problems (if not work stoppages) (Ebrahimpour 1985). The flow of parts will cease until a problem is resolved and if the problem is major, the entire production facility may be forced to cease operation (see also, Hendrick 1988). Hence the quality of the product is of utmost importance for the efficient functioning of a JIT system. As noted above, workers are responsible for ensuring the quality of their work. The design of the product must also be of the highest quality and this is the responsibility of upper management (Johnston 1989; Ishikawa 1985, p80). Furthermore, the materials that the workers use must be of high quality. To ensure this, a company operating under JIT has a limited number of highly trusted suppliers (Newman 1988; Fieten 1989). These suppliers must be able to deliver high quality goods in the quantities needed at exactly the time that they are needed (Hill & Vollmann 1986; Connell 1984). To ensure this, most companies using JIT, have long term contracts (usually 3 to 5 years) with a small number of suppliers (Burton 1988). Ishikawa (1985, p161) gives a strategy for the selection of this limited group of suppliers (see also Willis & Huston 1990). Limiting the number of suppliers and awarding them longterm contracts contrasts quite sharply to industry norms of North America, where shorter term contracts are usually put to tender and awarded to the company that submits the lowest bid.

The points above summarize the key features of any JIT system. To function successfully, a facility using the JIT approach must integrate all of these factors. The major benefits arising from the use of JIT (Gaither 1987, p546) are: Reduced inventory levels of raw materials, work-in-process and finished goods; increased productivity levels and increased utilization of the equipment; increased product quality and a reduction of scrap and rework; a reduction in lead times and a greater flexibility in changing the production mix; a smoother flow of production with shorter set-up times, multi-skilled workers and fewer disruptions due to quality problems; reduced space requirements due to an efficient plant layout and lower inventory levels; and, because the focus in JIT manufacturing is on solving production problems, the manufacturing operations become increasingly more streamlined and problem-free.

I.3 The Exporting of JIT from Japan

Justis (1981) reported that many Japanese production techniques had infiltrated North American manufacturing. Gilbert (1990) surveyed more than 100 U.S. companies and found that the majority of them are using JIT procedures in one form or another. Im and Lee (1989) discuss the implementation of just-in-time systems in U.S. manufacturing firms. Many large companies in the U.S. such as Sharp, Sanyo, Hewlett-Packard, Kawasaki, Matsushita, Sony, Black & Decker and IBM have adopted the philosophy and practices of Japanese manufacturers to varying degrees; all with U.S. workers (Gaither 1987, p550). Hence, JIT concepts have achieved at least an initial foothold in North American industry.

The influence of this adoption by certain major companies has necessitated changes in other companies with which they do business. In

response to its use of JIT, General Motors requested that its suppliers relocate closer to its plants and make more frequent deliveries (Automotive News, March 22, 1982, p3). However, the use of JIT has not been the sole domain of major U.S. industries. Allaby (1986) reports that the JIT approach has been successfully implemented at the Allen-Bradley facility in Canada. Laver (1991) describes how Camco (Canada's largest manufacturer of household appliances), using new JIT methods, is able to process a dealer's order, build the appliance and ship the finished product within three days; as opposed to its former 120-day manufacturing cycle. By adopting its kanban system, Camco has been able to reduce its massive inventory costs, simplify its product design (a new line of dishwashers in 1989 used 300 different components compared to 500 parts in the older version) and switch to a system of single sourcing (the number of outside suppliers has been reduced to 500 from 1500 in 1986). These examples have examined the use of JIT only in the larger manufacturing industries.

There are examples where JIT has been used for far more diverse applications than large, repetitive manufacturing. Finch (1986) and Inman & Mehra (1990) study how JIT techniques have been applied to smaller operations. Inman & Mehra, in surveying 100 companies, concluded that the benefits for the smaller companies using JIT are as great as the benefits accruing to the larger manufacturers. A number of these examples have been cited in the literature. Finch & Cox (1986) examine the application of JIT to a small bottling operation. Groenevelt & Karmarkar (1988) perform a case study of a batch flow shop producing a woven product under a JIT control system. Gravel & Price (1988) consider if any of the elements of JIT are applicable to

the job shop environment. Cheng (1988) studies the use and perception of JIT by the electronics manufacturers in Hong Kong. Carlson (1989) uses a case study to show how JIT concepts are applicable to warehousing and distribution operations. Wildemann (1988) describes how JIT is used within the West German manufacturing industries. Parnaby (1988) shows how JIT was implemented at a small company in the United Kingdom.

However, not all JIT techniques belong to the exclusive domain of manufacturing. Billesbach & Schneiderjans (1989) answer the question "Can JIT techniques be applied to non-manufacturing sectors?" with a definitive "yes". They examine JIT methods in an administrative, office setting. In a rather different vein, Smith et al. (1989) look at the medical supply system in Ecuador. In their paper, the replenishment policies and supply shortages of the country's health care system are examined, where the goal is to supply medical personnel throughout the country with the appropriate materials just as they are needed (i.e. just in time). Forbes et al. (1989) examine the managerial accounting practices needed for a medical equipment manufacturer whose supply function operates under a JIT philosophy.

These examples have dealt with the successful implementation of JIT. They have demonstrated how increasingly widespread the adoption of JIT has become into a diverse realm of applications (worldwide) and would tend to indicate that the inclusion of JIT in manufacturing (and to some extent non-manufacturing) throughout the world is increasingly more desirable. Koten (1982), however, reports that not all JIT concepts that were so successfully used in the Japanese auto plants are readily transferrable to U.S. auto manufacturers due largely to geography and demographics (see also Westbrook 1988).

Wilson (1985) advises a very slow transition to JIT, if at all, and says that companies would be better off adopting only specific parts of JIT such as reducing the cost of setups. Esparrago (1988) states that it is a lack of understanding of the JIT philosophy by the North American firms that has impeded its successful, universal implementation. Marshall (1977) points out that the success of JIT may be largely attributable to the cultural and social characteristics of the Japanese labour force. Ishikawa (1985, p29) disagrees (rather begrudgingly) with this not uncommon outlook and concludes that "nations other than [Japan]...can also succeed in these endeavors [of implementing JIT]".

Some aspects of North American culture,however, may prove to be too great a barrier to the verbatim copying of JIT as practised by the Japanese. Most notably, the Japanese concept of lifetime employment with one company would probably not succeed in North America. With the need for multi-functional workers and the time period required for the necessary training to occur, Sugimori et al (1977) conclude that it is imperative that employees be hired with the expectation of long-term employment. This philosophy is not consistent with either employers or employees in North America. Liberman et al (1990) conclude that in the auto industry, in both the U.S. and Japan, productivity improvement over the last 40 years has been attained primarily through more efficient utilization of labour. Studying these options would be a fruitful area of work for those involved in behavioural research.

A company must consider many factors if it is to successfully implement a JIT approach (Lee & Ebrahimpour 1984). Huang et al (1983) simulated the adaptability of JIT to a North American production environment

and, for the variables they examined, concluded that any company wanting to use JIT would require a lengthy transition period in order to prepare the production environment. Furthermore, successful implementation would most likely occur in those firms which most closely resemble the Japanese companies in terms of size, resources and market share [e.g. large manufacturers such as automotive and appliance companies]. Regardless of these caveats, it is very apparent that JIT has been adapted into the North American manufacturing environment within companies of various sizes and diverse technologies and studies of its potential applications must be made. The questions of implementation will be addressed in the next section.

I.4 The Implementation of JIT in North America

When traditional North American manufacturing techniques are mentioned one is usually referring to the systems of MRP and MRP II. MRP (Material Requirements Planning) is a computer based information system designed to handle the ordering and scheduling of the dependent demand inventories (i.e the raw materials, component parts and subassemblies). It begins with a schedule for finished goods that is converted into a schedule of requirements for the subassemblies, component parts and raw materials that will be needed to produce the finished items in the specified time frame (for a complete description of MRP see Orlicky 1975). A production plan for a specified number of finished products is translated backward using lead time information to determine when and how much to order. The requirements for the final products generate requirements for lower-level components so that the ordering, fabrication and assembly can be scheduled for the timely completion of end items, while inventory levels are kept reasonably low (Stevenson 1990,

p583). MRP is an approach to scheduling and inventory control designed to answer the three questions; what is needed, how much is needed and when is it needed (Bevis 1979).

MRP II (Manufacturing Resources Planning) is a second generation approach to MRP that has an expanded scope which involves other areas of the firm (i.e. marketing, finance) in the planning process. [Henceforth, MRP and MRP II will be collectively referred to as simply MRP.] The expected benefits of MRP were; low levels of in-process inventory, the ability to keep track of material requirements, the ability to evaluate capacity requirements generated by a given master schedule, and a means of allocating production time (Stevenson 1990, p606). MRP is a formal, computer-based system concerned with projecting requirements and with planning and levelling capacity using a computer. In contrast, JIT with kanban is a manual, 2-bin system where supplies are replenished when a predetermined level is reached. A kanban system operates in a very different fashion from the way MRP would control the same facility. However, regardless of the approaches of the respective systems, the goals are the same.

Stevenson (1990, p607) notes that MRP has not proved to be the "cure-all that many hoped it would [be]." Lambrecht & Decaluwe (1988) note that MRP experiences a lot of operational problems and Maher (1986) concludes that MRP will not be satisfactory for the 'factory of the future'. Indeed Lambrecht & Decaluwe state that MRP imposes demands on the process that cannot be met and that "inventory is too often the solution." They describe two case studies where JIT offers useful insights which were not gained from the use of MRP. While MRP has long been proclaimed as the entire solution for the

control problems of almost all manufacturing environments (Bridgette 1976), it has turned out to be only partially true.

In light of the successes that the Japanese companies have had with JIT techniques, the question becomes, can JIT be transplanted on a large scale into North America? Hall (1981, p21-31) and Schonberger (1982, p83-102) examine whether the manual kanban systems used in JIT manufacturing may be a viable replacement of MRP. Parnaby (1988) found that while kanban enforces a redesign of the manufacturing system [for the better], MRP creates the "delusion that problems can be solved merely by the imposition of a complex computer system upon a complex factory process." Suri & Treville (1986) state that many firms that currently use MRP are hesitant to switch to JIT due to potential disruptions that may occur during the changeover. Malley & Ray (1988) conclude that an entirely new information system would be necessary for such an implementation. Im (1989) poses the question of whether the largely computerized North American manufacturers would even consider switching to manual, kanban systems; thereby rendering their powerful and generally expensive computer systems virtually redundant. Rao (1989[a]) surveyed fifty-seven FORTUNE 100 companies regarding their software use to support materials management and found that all companies implementing JIT also had an MRP system. Rao (1989[b]) noted that software companies were planning to adapt their current systems to support JIT and that computerized companies were prepared to switch their information systems to support kanban manufacturing (see also Sewell 1990).

Several researchers have examined the process of implementing JIT into the North American manufacturing environment. While some advocate a

complete switch to JIT production, most foresee a hybrid manufacturing system in which both push and pull strategies are used in conjunction (Belt 1982) and the best features of JIT are adopted. Willis & Suter (1989) present a conversion life cycle system for the implementation of JIT. Lee & Ebrahimpour (1984) provide some of the salient features and prerequisites which must be met for such a switchover to JIT production. Rao & Scherage (1988) describe the change from MRP to JIT manufacturing. Olhager & Ostlund (1990) look at the combining of push and pull strategies. They provide an illustrative case of this push/pull integration in a make-to-order environment. Discenza & McFadden (1988) look at the integration of MRP and JIT through the unification of software. Sipper & Shapira (1989) develop a decision rule which enables the classification of an inventory control policy for a production system to be either a JIT system or a traditional system. Flapper et al (1991) describe how JIT can be imbedded into MRP using a 3-step framework when some (or all) of the products are to be produced using JIT. They describe to what extent MRP can be used to support JIT. While it might appear that the only way to implement JIT is to dismantle the existing system, Miltenburg & Wijngaard (1991) provide a 3-phase transformation process for the phasing in of JIT. They construct a 'gentle' procedure for this change which causes as little disruption as possible to the existing production system. Thus, methods have been researched which would allow for the implementation of JIT into North America. If this implementation is to be successful, then analysis of the performance of JIT systems must also be undertaken. Performance analysis will be reviewed in the next section.

I.5 Performance Analysis

If JIT is to be implemented then studies of its potential benefits and drawbacks must be made. Performance analysis requires that some particular measure of a firms efficiency (or some surrogate estimator thereof) be examined, subject to certain limiting assumptions and constraints, which facilitate the study. Whenever these assumptions are made, the inherent limitations alluded to in the opening paragraph of this chapter are invoked.

In this effort, Crawford & Cox (1990) developed certain standards, performance criteria and measurement techniques for evaluating JIT production systems. As the intensive study of JIT has only occurred in the past decade, they attempt to provide a standardized approach to the reporting methods for the analysis of JIT. Similarly, Heiko (1989) attempts to smooth the barriers to comprehension, so that practitioners of JIT can communicate their findings more readily to the corporate managers not necessarily involved in the manufacturing process; thereby allowing the costs and benefits of JIT to be understood by the decision makers in a production environment.

Most of the published studies of JIT to date have been simulated (e.g. see Fallon & Brown 1988) mainly to test conditions of inventory levels and to determine the optimal number of kanbans to allow. For example, Lee (1987), Huang et al (1984), Kimura & Terada (1981), So & Pinault (1988), Sarker & Harris (1988), Philipoom et al (1987), Ebrahimpour & Fathi (1985), Cadley et al (1989) and Gupta & Gupta (1989) all model certain production aspects of a firm operating under JIT while simulating the variation in certain exogenous conditions. Other related work for finding the number of kanbans has also been performed. Bitran & Chang (1987) investigate solution procedures to determine

the number of kanbans in a deterministic setting. Li & Co (1991) present a dynamic programming method for determining the number of kanbans at each stage of production in a multi-stage system. Wang (1990) uses a Markov process to determine the number of kanbans in a particular system. Seidman (1988) develops two tractable and optimal kanban control policies. Miyazaki et al (1988) look at ways to determine the number of kanbans and the time interval between when orders are placed.

Several studies have focused specifically on the effect that JIT has had on inventory. Toomey (1989) studies how inventory control strategies must be adapted in various types of environments when JIT is implemented. Pan & Liao (1989) use an order splitting model to describe a JIT inventory system. Luss & Rosenwein (1990) examine lot-sizing for JIT manufacturing and develop a heuristic procedure that minimizes the inventory holding costs over all items. Other studies of the JIT lot-sizing problem include South (1986) who derives a minimum lot-size formula and Grant & Seastrand (1987) who look at multiple operation lot-sizing. In a much different approach, Buzacott (1989) uses a queueing network with blocking to describe the performance of a kanban system.

I.6 The Scheduling of JIT Systems

Not very much research, however, has been published on the controlling of JIT manufacturing environments using scheduling techniques. Egbelu & Wang (1989) study the role of order scheduling while a firm gradually adopts JIT, with the objective being to minimize inventory costs. Wortman & Monhemius (1984) examine how JIT and MRP would schedule a production facility. Plenert & Best (1986) explore whether traditional MRP systems or JIT

would be better for controlling a particular facility. According to Okamura & Yamashita (1979) , among other things, one problem that must be solved for the effective utilization of mixed-model assembly facilities is the determination of the sequence in which the products must be scheduled on the final assembly process. As JIT manufacturing operates in a pull environment, once the final assembly sequence for the products on a process is set, the scheduling for all parts that feed this process is also set (theoretically). Groeflin et al (1989) develop a heuristic procedure for determining the final assembly sequence for a facility in which a JIT control system is employed. The objective of their algorithm is to sequence the assembly process in such a fashion that the work-in-process inventory of parts in the feeder shops is 'smoothed' (i.e. no large fluctuations in inventory). They note that this sequencing problem is a complex combinatorial problem and that "an efficient, exact algorithm is unlikely to exist " (e.g. this is because if there are n end-products, there will be n! feasible sequences to consider and, therefore, their problem suffers from the 'curse of dimensionality').

In the related literature, the design of scheduling algorithms for flexible flow lines (e.g. integrated circuit board lines that feed parts to the final assembly) has been investigated by Wittrock (1985, 1988), Kochar et al (1987), McCormick et al (1989) and Rege (1988). The output of a final assembly sequencing algorithm, such as the one considered in Groeflin et al, would provide a demand schedule which could serve as the input for the scheduling of such feeder processes. The interrelationships between final assembly and operations in feeder shops is discussed in Luss (1989). Final assembly sequencing is related to other production scheduling problems. For surveys of

various types of scheduling problems see, for example; French (1982), Baker (1974), Conway, Maxwell & Miller (1967), Graves (1981), Gupta & Kyparis (1987) and Lawler, Lenstra & Rinnooy Kan (1982).

French (1983, p9) notes that the objectives of scheduling problems are never easy to state as they are numerous, complex and often conflicting. [For example, Mellor (1966) describes 27 distinct scheduling goals.] The performance measure chosen may often appear to be too limited to allow for the accurate representation of the scheduling goals that tend to arise in practice. A schedule which minimizes a component cost may be very poor in terms of the total cost; where this total cost is a complex combination of processing costs, inventory costs, downtime costs and completion time costs. Even the problems of scheduling using simple performance measures can prove difficult as solution techniques often require heuristic methods (e.g. Groeflin et al).

By the inherent nature of JIT, the scheduling objectives are a different type of measure than those found in the 'traditional' scheduling problems. Namely, the JIT objectives are necessarily non-regular performance measures. A regular measure, R, is one that is non-decreasing in the completion times of the products (French 1982, p14). If there are n products, where product i, i=1,...,n, completes at time C_i, then R is a function of $C_1,C_2,...,C_n$ such that ;

$$C_1 \leq C_1^*,\ C_2 \leq C_2^*,\ \ldots C_n \leq C_n^*$$

together imply that

$$R(C_1,C_2,...,C_n) \leq R(C_1^*,C_2^*,...,C_n^*).$$

One seeks to minimize regular performance measures such as the makespan (i.e the job with the maximum completion time), the total tardiness of all jobs, and the mean flow time. Therefore, if two schedules exist such that no product

in the first schedule completes any later than it would in the second schedule, a regular performance measure would dictate that the first schedule is at least as good as the second.

However, simply minimizing the completion times of the products in a JIT system does not accomplish the goal of producing only the necessary products in the necessary quantities at the necessary time. If a product completes too early, then it will be held as inventory which, to a JIT system, is just as bad as completing a job too late. JIT scheduling objectives, therefore, must necessarily be non-regular performance measures as both the earliness and tardiness of product completion times must be penalized. The goal is to produce a schedule in which jobs are produced neither too early nor too late with repect to their desired completion time.

Baker & Scudder (1990) provide a survey of the machine scheduling problems with earliness and tardiness penalties. They note that for problems where products have distinct due dates, there are apt to be few special cases where efficient solution procedures can be found and that most of the more complicated, "interesting" problems are likely to remain relatively intractable. Rinnooy Kan (1976, p3) notes that if one can prove that a scheduling problem falls into a class of difficult combinatorial problems (i.e. NP-complete problems) with the property that no efficient (i.e. polynomial) algorithm has been found to optimally solve any of the problems within this class, then the use of an enumerative procedure for this problem is justified (see also Garey & Johnson 1979). Garey, Tarjan & Wilfong (1988) consider the problem of minimizing the total earliness and tardiness on a single processor where the products have different due dates. Hall & Posner (1991) note the relative importance of this

(Garey et al) paper, as it contains one of the first proofs of NP-completeness for a single machine scheduling problem with a non-regular performance measure (see Hall, Kubiak & Sethi 1991 for the NP-completeness proof of another non-regular measure).

These features highlight two of the major problems encountered when attempting to find optimal solutions to JIT scheduling problems. Firstly, as the various products in JiT problems tend to have different desired completion times (distinct due dates), solution techniques to problems which consider earliness- and tardiness- type objectives for these JIT models may be intractable. Secondly, non-regular performance measures are employed for studying these JIT problems but trying to prove that the problems using these measures fall into the NP-complete classification may be, in itself, a very imposing task .

Bearing the above points in mind, the scheduling goals for controlling the mixed-model assembly line at Toyota are described by Monden (1983, p181) as determining the product sequence on the final assembly process that;

1. Levels the load (total assembly time) at each station on the assembly line, and
2. Keeps a constant rate of usage of each part on the assembly line from one time period to the next.

With respect to the first goal (in what is referred to as the 'Loading' problem), if products with relatively longer operation times are sequenced successively, then delays in completing the product may occur and could cause the stoppage of the assembly line. To counteract this type of problem, Okamura & Yamashita (1979) developed a complex heuristic procedure for minimizing

the risk of stopping the conveyor for the assembly line model-mix sequencing problem.

The second goal (the so-called 'Usage' problem), is considered to be the most important goal for JIT production systems (Monden 1983, p182). Hall (1983) has referred to this as the problem of levelling the schedule. Monden describes how this usage goal is met using Toyota's "Goal Chasing Algorithm." With a pull system in effect, the product level becomes the focus of control for scheduling. Toyota's goal chasing algorithm is a heuristic procedure which determines this end-product sequence by myopically minimizing the sum of the deviations of cumulative part requirements from the average part requirements necessary to produce an endproduct.

Miltenburg (1989) formulated the usage problem as a quadratic integer programming problem with the assumption that the different products require the same number and mix of parts. This assumption reduces the solution procedure to determining the final assembly sequence of the end products without the need to 'keep track' of the effect that this sequence will have on the lower production levels. This is because, under the assumption, variability occurs only at the product level. In Miltenburg & Goldstein (1990) a joint usage and loading problem is developed. The usage problem is extended to multi-level problems in Miltenburg (1986), Miltenburg & Sinnamon (1989), and Goldstein & Miltenburg (1988) where the assumption that each product requires the same number and mix of parts is no longer made. All of the usage problems have been treated as single processor scheduling problems and require the determination of the final assembly sequence. The papers present several heuristics to solve the various usage problems. The final assembly

sequence generated by the multi-level heuristic procedures determines the production schedules and therefore the fluctuations in the part variability at the lower process levels (i.e. the sub-assembly, component and raw material levels which feed the final assembly level). As with the final assembly sequencing problem of Groeflin et al, the mixed-model, multi-level problem formulations also appear to be complex combinatorial problems.

Heuristics, however, are only as valuable as the solutions that they produce. Pearl (1984, p73) notes that while "...heuristics greatly reduce the search effort...[they] occasionally fail to find...even a near-optimal solution." In order to test how 'well' a heuristic performs one must determine either worst case bounds on the solution values or compare the results to optimal solutions. Thus far, no research has been performed to determine bounds on the heuristic procedures used for the mixed-model, JIT scheduling problems. Miltenburg, Steiner & Yeomans (1990) developed an enumerative dynamic programming procedure for the optimization of the single level usage problem and performed a limited experimental study which measured the performance of some of the heuristics. Kubiak & Sethi (1989, 1991) subsequently proved that the single level usage problem could be reduced to an assignment problem and, hence, solveable to optimum in polynomial time. Inman & Bulfin (1991) recently solved a different version of the single level usage problem in which the objective function to be minimized was the total earliness and tardiness of the schedule. They used due dates which corresponded to the ideal, but infeasible (being non-integer), location in the final assembly sequence. No research has been published for optimally solving the mixed-model, multi-level, JIT scheduling problems.

I.7 Goal of the Study

For the study to be undertaken, the optimization of the usage problem for the scheduling of mixed-model, JIT assembly processes will be examined using a certain minimax objective function. This minimax objective has not been considered previously in the literature for use with mixed-model JIT assembly facilities. The aim is to determine optimal procedures for this new formulation and to examine the performance characteristics of these procedures. Certain processes developed within the study of this minimax problem might be applicable to the 'sum of deviation' objectives encountered previously in the scheduling of mixed-model, JIT assembly systems. However, the operating characteristics for the optimization of this new objective could reasonably be expected to significantly differ from these prior efforts. Before introducing the problem with the new objective, the existing literature on mixed-model, JIT systems will be reviewed in more detail.

CHAPTER 2 REVIEW OF MIXED-MODEL, JIT SCHEDULING

II.1 Introduction

The problem of scheduling mixed-model assembly facilities operating in a JIT environment has come under study only in the past decade. The seminal work in the area is due to Monden (1983) who described the scheduling algorithm employed by Toyota to control their mixed-model assembly line. Miltenburg (1989) presented an integer programming formulation for a particular version of this problem. A number of alternative formulations and approaches to that appearing in Miltenburg have been made and it is these models which will be reviewed in this chapter.

Mixed-model assembly facilities, which are capable of diversified small-lot production, must necessarily have negligible switchover costs from one product to another. JIT production methods, which require producing only the necessary products in the necessary quantities at the necessary time, make it possible for these types of facilities to satisfy customer demands for a variety of products without holding large inventories or incurring large shortages. JIT production systems, which have been described extensively in the literature (for example see, Hall 1983; Ohno 1988; Monden 1983; Fukuda1983), operate under a pull process. If the products, subassemblies, components and raw materials are thought of as inputs or outputs of separate production levels then, under this pull process, production is initiated only by one level's requirement for another level's output.

A production level which receives as input another level's output is deemed to be a higher production level than the one that "feeds" it. Hence, in a JIT assembly facility, the highest production level is the product level, where final assembly takes place. In all of the models of this assembly process, it is assumed that once production of a final product is started that it will be worked upon until completion. Thus, a JIT assembly facility can be thought of as a multi-level production process where, once the sequence of production at the product level is fixed, the production schedules at all other levels are inherently also fixed. As a result, the final assembly level necessarily becomes the focus of control for the entire production facility.

Monden (1983, p181) and Miltenburg & Sinnamon (1989) state that the paramount goal for scheduling in a JIT system is to maintain a constant rate of usage for every part used by the system. Therefore, the scheduling problem for a mixed-model, JIT assembly facility is to find a sequence of production for the final products (the highest level) such that the quantity of each part (over all levels) used within this multi-level production system is kept as constant as possible per unit time. Hall (1983) refers to this as levelling or balancing the schedule, but it has also been referred to as the "usage problem" in the literature.

II.2 Mathematical Model

Before proceeding to descriptions of the various models which have been studied, certain notation and concepts, common to all of these models, will be defined. First assume that there are L different levels of production. A particular level under consideration will be denoted as level j, where j=1,...,L. The highest production level (i.e. the product level) is level 1. The number of

different outputs at level j will be denoted as n_j and d_{ij} represents the demand for output i at level j, where i=1,2,...,n_j.

If t_{ijl} represents the number of units of output i at level j required to produce one unit of product l, l=1,2,...,n_1 , then $d_{ij}=\sum_{h=1}^{n_1} t_{ijh}d_{h1}$ (where t_{11h}=1 if i=h and 0 otherwise). Let $D_j=\sum_{i=1}^{n_j} d_{ij}$ be the total demand for level j's output. Then the demand ratio for output i at level j will be $r_{ij} = \frac{d_{ij}}{D_j}$. [Note that $\sum_{i=1}^{n_j} r_{ij}=1$, for each j=1,...,L].

One of the assumptions of the model is that production of the final products (i.e. output at level j=1) cannot be preempted. That is, once production on a unit of a level 1 product has commenced, it must be completed before the production of another unit can start. This introduces the concept of a stage or cycle. One is said to be at stage k (or in cycle k) if k units of product have been produced at level 1. There will be k complete units of the various outputs i at level 1 produced during these first k stages and the time horizon will consist of D_1 stages in total. Let x_{ijk} represent the number of units of output i at level j produced during stages 1 through k and let $XT_{jk} = \sum_{i=1}^{n_j} x_{ijk}$ be the total number of units produced by level j during stages 1 through k. Therefore the cumulative production at level 1 through the first k stages is XT_{1k}=k. Then by the pull nature of JIT systems and from the fact that lower level parts (from levels 2,3,...,L) are drawn as needed by the final assembly process, the particular combination of the highest level products produced during these k stages determines the

cumulative production at every other level. Hence, for levels $j \geq 2$, the required cumulative production through k stages for output i will be $x_{ijk} = \sum_{l=1}^{n_1} t_{ijl} x_{l1k}$.

If $G_{ij} \geq 0$ is a weighting factor for the i^{th} part at level j then the general mixed-model, JIT scheduling problem is to select the x_{i1k}, $i=1,...,n_1$, $k=1,...,D_1$, which satisfy;

$$\text{Min} \sum_{k=1}^{D_1} \sum_{j=1}^{L} \sum_{i=1}^{n_j} G_{ij} \left(x_{ijk} - XT_{jk} r_{ij}\right)^2$$

s.t.

$$x_{ijk} = \sum_{p=1}^{n_1} t_{ijp} x_{p1k} \qquad i=1,...,n_1,\ j=1,...,L,\ k=1,...,D_1$$

$$XT_{1k} = \sum_{i=1}^{n_1} x_{i1k} = k \qquad k=1,...,D_1$$

$$XT_{jk} = \sum_{i=1}^{n_j} x_{ijk} \qquad j=2,...,L,\ k=1,...,D_1$$

$$x_{i1k} \geq x_{i1(k-1)} \qquad i=1,...,n_1,\ k=1,...,D_1$$

$$x_{i1D_1} = d_{i1}, \quad x_{i10} = 0$$

Because of the no-preemption assumption for production at the highest level, the x_{ijk}'s necessarily must hold integer value. Hence, this generalized mixed-model, JIT scheduling problem is a quadratic integer programming problem. The objective function in the model inherently recognizes the sequence dependent nature of the lower level parts (i.e. the subassembly, the component and the raw material levels). The x_{ijk}'s are calculated directly from the assembly sequence of the (level 1) products and the desired level of production of part i at level j is calculated as the proportion (r_{ij})

of the total cumulative production at level j (XT_{jk}). Level schedules are created by keeping the production of all parts and products as close to this desired level as possible.

Other objective functions have also been proposed for the mixed-model scheduling problem (Miltenburg 1986, Miltenburg 1989). Any of the following objectives could be used for modelling the problem subject to the constraints of the general model;

$$\text{Min} \sum_{k=1}^{D_1} \sum_{j=1}^{L} \sum_{i=1}^{n_j} G_{ij} \left(\frac{x_{ijk}}{XT_{jk}} - r_{ij} \right)^2$$

$$\text{Min} \sum_{k=1}^{D_1} \sum_{j=1}^{L} \sum_{i=1}^{n_j} G_{ij} \left| \frac{x_{ijk}}{XT_{jk}} - r_{ij} \right|$$

$$\text{Min} \sum_{k=1}^{D_1} \sum_{j=1}^{L} \sum_{i=1}^{n_j} G_{ij} \left| x_{ijk} - XT_{jk} r_{ij} \right|.$$

These functions minimize either the squared or absolute deviation of the actual production from the desired production for all levels in the system. The functions involving the $\frac{x_{ijk}}{XT_{jk}}$ terms, try to keep the actual proportions of the production mix (the $\frac{x_{ijk}}{XT_{jk}}$) as close as possible to the desired proportions (the r_{ij}) over all stages k. The other functions try to keep the actual number of units produced (the x_{ijk}) as close as possible to the desired number of units produced ($XT_{jk} r_{ij}$) over all stages. Any of these functions are reasonable and none is more tractable, mathematically, than the others. It is shown in Miltenburg (1989) that all of these functions produce similar schedules and, hence, functions as described in the general, mathematical model above are investigated.

II.3 Review of Mixed-Model JIT Problem Formulations

The various models which have appeared in the literature will now be reviewed and it will be shown how these models relate to the the general model presented above. Solution methods for these models will also be reviewed. The primary goal for scheduling a JIT system is to maintain a constant rate of usage for all parts used within the system. JIT systems can only function effectively when this constant rate of part usage exists. Wide fluctuations in part requirements would necessitate either the holding of large part inventories or the incursion of large part shortages; neither of which can occur if the JIT system is to function appropriately. Recognizing that part usage is determined by the sequencing of the products on the final assembly process, variation of each part is minimized by sequencing the products in very small lots (often of size 1).

II.3.1 Single Level Usage Problem

In Miltenburg (1989) it is assumed that each product requires approximately the same number and mix of lower level parts. A constant rate of part usage is achieved by considering only the demand rates for the products and implicitly ignoring the resulting part demand rates as variability will only occur at the product level. The goal is to find an assembly sequence which maintains a constant rate of production for each product.

Letting $G_{ij} = \begin{cases} 1 & \text{if } j = 1,\ i = 1,\ldots,n_1 \\ 0 & \text{else} \end{cases}$

then the objective function for this model may be written as;

$$\sum_{k=1}^{D_1} \sum_{i=1}^{n_1} \left(x_{i1k} - XT_{1k} r_{i1}\right)^2.$$

As this model focuses entirely upon the highest level of production, the problem is referred to as the single level usage problem.

II.3.2 Multi-Level Usage Problem

If each product does not require the same number and mix of parts then one can no longer focus solely upon the highest level to control the part usage at the lower levels. An accounting for how the level 1 product assembly sequence affects the lower levels must be made. This problem is modelled in Miltenburg (1986) where the objective function measures the variation of a part's actual production from its relative, cumulative amount of production for all levels in the system. If $G_{ij} = W_p$, $i=1,...,n_1$ with $j=p$, where W_p is the weighting factor for the p^{th} level, then the objective function for the multi-level usage model may be written as

$$\sum_{k=1}^{D_1} \sum_{j=1}^{L} \sum_{i=1}^{n_j} W_j \left(x_{ijk} - XT_{jk} r_{ij}\right)^2.$$

This paper introduces the concept of weighting factors for the various production levels. In practice, weighting factors are used to prevent a relatively less important lower level part from dominating more important higher level parts in the calculation of the objective function. However, the implementation of weighting factors requires that some arbitrary (although, perhaps, well thought out and well intentioned) decision be made as to what values are to be assigned to the weights. The type of solution to these problems is biased by and depends very heavily upon the weighting values chosen. An open question is to determine what weighting factors are appropriate for the various JIT models. In a sense, "unbiased" usage problems require that $W_j=1$, $j=1,...,L$ (or $W_j=0$ for some levels not of interest). Hence, for this study, two types of usage problems will be defined:

(i) (Unweighted) Usage problems where $W_j=0$ or 1, $j=1,...,L$, and

(ii) Weighted usage problems where weighting factors other than those in (i)

apply.

The weighting factors used within the various problems considered in the literature have all corresponded to a particular level's weight, whereas, in the general model, a potentially different weight is given for each part and product in order to completely generalize the problem. The distinction between the weighted and unweighted cases will always be made clear and, thus, according to these definitions, the objective function above refers to a multi-level, weighted usage case.

II.3.3 Weighted Multi-Level Usage Problem with Pegging

In Goldstein & Miltenburg (1988) a method is proposed which considerably simplifies the amount of computation required to determine the sequencing of products in a multi-level system. Under what is referred to as the 'pegging' assumption, a distinction is made between t_{ijh} and t_{ijl}, $h \neq l$, for each output i at level j. Parts of output i at level j are dedicated or "pegged" to the level 1 product into which they will be assembled. This pegging scheme separates parts into distinct groups for each product. Whereas the objective functions shown above indicate that the desired relative, cumulative level of production for each part at level $j \geq 2$ depends upon the sequence of product assembly, pegging allows the calculation of specific targets for the amount of each part's production at each stage which is independent of the final assembly sequence. The objective function for this problem is;

$$\sum_{k=1}^{D_1} \sum_{i=1}^{n_1} W_1 \left(x_{i1k} - kr_{i1}\right)^2 + \sum_{k=1}^{D_1} \sum_{j=1}^{L} \sum_{h=1}^{n_1} \sum_{i=1}^{n_j} W_j \left(x_{h1k} t_{ijh} - kt_{ijh} r_{h1}\right)^2.$$

Recalling that $XT_{1k}=k$ and that $t_{i1h}=1$ if $i=h$ and 0 otherwise, then this objective may be rewritten as;

$$\sum_{k=1}^{D_1} \sum_{j=1}^{L} \sum_{h=1}^{n_1} \sum_{i=1}^{n_j} W_j (t_{ijh})^2 \left(x_{h1k} - XT_{1k} r_{h1}\right)^2.$$

By changing the order of summation and setting $G_{h1} = \sum_{j=1}^{L} \sum_{i=1}^{n_j} W_j (t_{ijh})^2$, then the objective function for the weighted, multilevel, pegged formulation becomes;

$$\sum_{k=1}^{D_1} \sum_{h=1}^{n_1} G_{h1}\left(x_{h1k} - XT_{1k} r_{h1}\right)^2,$$

which is equivalent to the formulation of a weighted, single level usage problem.

II.3.4 Loading Formulation

The above models have all dealt solely with the usage-type problems. In Miltenburg & Goldstein (1990) a loading factor term is added on to the weighted, multi-level usage problem in what is referred to as the joint problem. This loading factor term adds an assembly line balancing-type problem to the original formulation. The goal of this inclusion is to smooth the work load on the final assembly process (now considered to be an assembly line) to reduce the chance of production delays and stoppages. The loading goal attempts to prevent the consecutive sequencing of products with relatively longer production times by placing products with shorter processing times on either side of them in the final schedule.

If T_s^i is the production time required by product i at station s and $\bar{T}^s = \sum_{i=1}^{n_1} T_s^i \, d_{i1}/D_1$ is the average production time required at station i, then the objective function for this joint problem is;

$$\alpha_U \sum_{k=1}^{D_1} \sum_{j=1}^{L} \sum_{i=1}^{n_j} W_j \left(x_{ijk} - XT_{jk} r_{ij}\right)^2 + \alpha_L \sum_{k=1}^{D_1} \sum_{s} W_s \left(\bar{T}^s\right)^{-2} \left[\sum_{i=1}^{n_1} x_{i1k} T_s^i - k\bar{T}^s\right]^2$$

where W_s is the weighting factor for station s and α_U and α_L are the weights applied to the usage and loading goals respectively.

II.4 Heuristic Solution Techniques

Thus far, only the various mathematical models for the usage (& loading) problems of mixed-model, JIT assembly processes have been shown and no indication has been given as to how solutions to these problems have been determined. These types of sequencing problems are usually difficult to solve as combinatorial approaches must generally be undertaken. Even "mid-sized" instances of the problems have hundreds of thousands of variables and the number of potential feasible solutions will be $\frac{D_1!}{d_{11}!\, d_{21}!\ldots d_{n_1 1}!}$. Hence, the number of potential solutions to a problem is factorial in nature and any solution method must somehow contend with the large number of feasible solutions in order to obtain a 'good' solution. As no general solution techniques exist for solving large integer programming problems, procedures must be specifically developed.

The methods that have been used for solving the various models will now be addressed. The approach taken in Miltenburg (1989), Miltenburg (1986) and Goldstein & Miltenburg (1988) [and in Miltenburg & Goldstein (1990)] has been to use two related, 'greedy' heuristics for scheduling the usage problems. The one stage or H1 heuristic is to schedule at each stage k, $k=1,\ldots,D_1$, the product p, $p\varepsilon\{1,\ldots,n_1\}$ which minimizes $\sum_{j=1}^{L} \sum_{i=1}^{n_j} W_j \left(x_{ijk} - XT_{jk} r_{ij}\right)^2$. This is a one stage heuristic as it considers only the variation for one stage (i.e stage k) in order to make a decision. No consideration is paid to the effect that

the decision made at this stage will have on the variation at future stages. The two stage or H2 heuristic presents a similar, albeit a slightly more complex, approach than this H1 heuristic. The H2 heuristic schedules the product p, $p\varepsilon\{1,...,n_1\}$, at stage k, $k=1,...,D_1$, which, if scheduled in conjunction with some product p^*, $p^*\varepsilon\{1,...,n_1\}$, at stage k+1, would minimize the total variation of all parts and products for stages k and k+1. H2 is a two stage heuristic as it considers only the variation for two consecutive stages in the decision making process. Once again, no consideration is made for the effect of the decision at this stage on the variation at future stages. Both of these heuristics are easily understandable, easy to implement and require minimal processing time.

Another heuristic approach presented in Miltenburg (1989) and Miltenburg & Sinnamon (1989) involves determining the nearest integer points to the desired level of production for each product at each stage k. The desired, feasible level of production for product i at stage k is the integer nearest to kr_{i1} (but is subject to the constraint that the nearest integer points for all products at a given stage must sum to k). It was proved in these papers that finding these nearest integer points for all products at all stages might (and quite probably would) produce infeasible schedules and, thus, one could not derive a production sequence simply by determining the nearest integer points for all products at all stages. [Note that if a feasible schedule is produced for the single level problem where the production of each product at each stage is identical to its nearest integer point, then this schedule is also optimal]. Methods were presented for regaining feasibility for those instances where an infeasible production sequence was generated. Feasibility was obtained by progressing from the last integer point preceding the first case of infeasibility of

the schedule by using one of H1, H2 or total enumeration until the next feasible integer point for all products on the schedule was found. Further cases of infeasibility in the schedule would be corrected in the same fashion. All of these nearest integer point heuristics have the potential for being very computationally intensive procedures (see Inman & Bulfin 1991).

A heuristic procedure has recently appeared which solves a slightly different problem formulation. Inman & Bulfin (1991) show that if $z_{ij}=\frac{1}{r_i}\left(j - \frac{1}{2}\right)$, $j=1,\ldots,d_i$, $i=1,\ldots,n_1$, is treated as the due date for the j^{th} copy of product i and if y_{ij} denotes the location in the production sequence of the j^{th} copy of product i, the objective functions

$$\sum_{i=1}^{n_1}\sum_{j=1}^{d_i}\left(y_{ij} - z_{ij}\right)^2 \quad \text{and} \quad \sum_{i=1}^{n_1}\sum_{j=1}^{d_i}\left|y_{ij} - z_{ij}\right|$$

may be optimized by sequencing the copies of product using an earliest due date (EDD) rule. These objective functions are somewhat akin to the single level usage problem. [It should further be noted that the due date used is not, in general, a feasible due date as the position that a copy of a product occupies in the production sequence must necessarily hold integer value. The integer ceiling of the above due date is used in the optimization procedure of Kubiak & Sethi (1989, 1991)]. While the EDD sequence is optimal for the above objective functions, it produces good results for the single level usage problem (and for $\sum_{k=1}^{D_1}\sum_{i=1}^{n_1}\left|x_{i1k} - kr_{i1}\right|$) as well.

Inman & Bulfin compare the results from their heuristic to those of the nearest integer point heuristic which appeared in Miltenburg (1989). One hundred problems with 10 products each were tested and, it is reported, that on average their EDD approach slightly outperformed the nearest integer point

heuristic. However, (as could be expected) this EDD heuristic was far more computationally efficient. Although results and comparisons are made between these heuristics, it is not known how well either heuristic approach performs when compared to the optimal solutions and, thus, the value of these heuristics has not been determined.

This section has shown that a number of heuristic approaches exist for solving the mixed model, JIT problems. However, their use and the lack of information on their worst case performance underscores the distinct necessity for the optimization of these problems. The approaches to optimization will be discussed in the subsequent section.

II.5 Optimization

As noted, it is possible for even 'reasonably' sized instances of the mixed-model, JIT scheduling problem to have hundreds of thousands of feasible solutions and a sustantially higher number of decision variables. As standard solution techniques do not exist for solving quadratic, integer programming problems of this size, the use of heuristics appears to be a practical approach for finding solutions. However, the value of a heuristic is only as good as the solution that it produces. Indeed, '...some heuristics greatly reduce search effort but occasionally fail to find...even a near-optimal solution' (Pearl 1984, p73).

Hence, there is a need to find either bounds on the solution value obtained by a heuristic or to develop a method which finds the optimal solution in reasonable time. If a bound has not been theoretically proved for a heuristic approach then it is necessary to be able to calculate the optimal value for a

given problem instance against which the heuristic value can be compared. Two such optimization approaches exist for the single level problems.

II.5.1 Single Level Optimization

In Miltenburg, Steiner & Yeomans (1990) an implicit enumeration procedure using dynamic programming (DP) was presented for determining optimal production schedules for the (weighted) single level problem. Compared to previously available optimization methods, this DP substantially reduced the computation and storage requirements. It was shown that the state space grew at an exponential rate in n_1, but that its growth rate was polynomial in D_1; indicating that the DP would be effective for small n_1 even with large D_1. A limited experimental study ($n_1 \leq 6$, $D_1 \leq 80$) demonstrated that the H1 and H2 heuristics gave good solutions for the problems examined. However, difficulties arose in using this approach for larger problems ($n_1=8$, $D_1=80$) as the "curse of dimensionality" hampered the DP's effectiveness. Thus, the problem of whether the heuristics (both H1 and H2) are capable of determining good schedules for very large problems remains open.

The single-level usage problem, described above, can be rewritten as,

$$\sum_{k=1}^{D} \sum_{i=1}^{n} \left(x_{ik} - kr_i\right)^2$$

if the subscript '1' is dropped and it is realized that $XT_{1k}=k$. Kubiak & Sethi (1989, 1991) have reformulated this into an assignment problem; thereby providing a means for efficiently finding optimal level schedules to the single level problem. A summary of their approach is; to separate each product into its individual copies, to calculate an ideal location for each copy in the final

sequence, to calculate the penalty cost of assigning a copy to a position in the final sequence other than its ideal position, and to finally formulate an appropriate assignment problem. A synopsis of this method will be outlined.

Define $Z_{ij}^{*} = \left\lceil \frac{2j-1}{2r_i} \right\rceil$ (where $\lceil x \rceil$ is the nearest integer not less than x i.e. the ceiling of x) to be the ideal location in the final production sequence for the j^{th} copy of product i, i=1,...,n, j=1,...,d_i. This location is ideal because if copy j is sequenced at any other location then the resultant deviations "attributable" to this scheduling decision are larger than if the copy is scheduled at Z_{ij}^{*}. Let C_{ijk} be the cost attributable to placing the j^{th} copy of product i in the k^{th} position of the final production sequence. Let (i,j) refer to the j^{th} copy of product i. If (i,j) is sequenced in its ideal location (i.e. $k=Z_{ij}^{*}$) then $C_{ijk}=0$. If (i,j) is placed too soon in the production sequence (i.e. $k<Z_{ij}^{*}$) then "inventory" costs ψ_{ijl} are incurred in the positions from l=k to $l=Z_{ij}^{*}-1$. If, however, (i,j) is produced too late (i.e. $k>Z_{ij}^{*}$) then "shortage" costs ψ_{ijl} are incurred from $l=Z_{ij}^{*}$ to l=k-1. If these inventory and shortage costs are calculated as,

$$\psi_{ijl} = \left| (j-lr_i)^2 - (j-1-lr_i)^2 \right|, \qquad i=1,...,n,\ j=1,...,d_i,\ l=1,...,D$$

then the penalties for each copy (i,j) to be assigned to a location $k \in \{1,...,D\}$ in the production sequence are;

$$C_{ijk} = \begin{cases} \sum_{l=k}^{Z_{ij}^{*}-1} \psi_{ijl} & k<Z_{ij}^{*} \\ 0 & k=Z_{ij}^{*} \\ \sum_{l=Z_{ij}^{*}}^{k-1} \psi_{ijl} & k>Z_{ij}^{*}. \end{cases}$$

Defining an indicator variable as

$$x_{jk}^{i} = \begin{cases} 1 & \text{if } (i,j) \text{ is sequenced in location } k \\ 0 & \text{else} \end{cases}$$

then their assignment problem is

$$\text{Min} \sum_{k=1}^{D} \sum_{i=1}^{n} \sum_{j=1}^{d_i} C_{ijk}\, x_{jk}^{i}$$

$$\text{st} \quad \sum_{i=1}^{n} \sum_{j=1}^{d_i} x_{jk}^{i} = 1 \qquad k=1,\ldots,D$$

$$\sum_{k=1}^{D} x_{jk}^{i} = 1 \qquad i=1,\ldots,n,\ j=1,\ldots,d_i$$

In Kubiak & Sethi (1989) it is proved that an optimal sequence for the single level problem can be constructed from any optimal solution to the above assignment problem. By induction, it is shown that the implied ordering for the copies of each product will not be violated in an optimal production sequence (i.e. the $(j+1)^{st}$ copy of product i will not appear before the j^{th} copy in an optimal sequence).

It is also shown in Kubiak & Sethi (1989) that objective functions of the form

$$\sum_{k=1}^{D} \sum_{i=1}^{n} F^{i}\left(x_{ik} - kr_i\right)$$

may also be reduced to this assignment problem formulation provided that $F^{i}(\bullet)$ is a non-negative convex, symmetric function satisfying $F^{i}(0)=0$, $F^{i}(y)>0$ for $y\neq 0$, $i=1,\ldots,n$. This is significant as it means that the weighted, single level problem (and thus the pegged multi-level case) can also be solved to optimum using this assignment problem reduction. Hence, Kubiak & Sethi have shown that a process exists for finding optimal solutions to the single level cases which is

polynomial in the total demand, D. Unfortunately, the extension of this approach to the general, multi-level case cannot be made.

II.5.2 Multi-Level Optimization

No optimization approaches have appeared in the literature for solving any version of the multi-level problem formulation to optimum. Thus, there has been no means available with which to compare the effectiveness of any of the existing heuristics to optimal values for any problems other than those problems small enough to be evaluated 'by hand'. A distinct need exists to develop a technique for multi-level optimization.

II.6 Summary

In this chapter, the literature concerned with the scheduling of mixed-model assembly facilities has been reviewed. The usage problem, also known as the level scheduling problem, has been identified as the problem of paramount importance within this environment.

A general mathematical model of the usage problem was given and the various types of usage formulations were presented. These usage models can be broken down by category; single-level , multi-level, weighted, and unweighted. A multi-level usage problem with the 'pegging' assumption was reviewed and was shown to be reducible to a single-level, weighted usage case. A joint formulation where an assembly-line balancing-type problem to be solved simultaneously with a multi-level usage problem was also mentioned.

Two myopic heuristic methods, H1 and H2, common to all of the models that appear in the literature were described. Two other heuristics, the EDD approach for single level problems and the nearest integer point heuristic,

were also outlined. Optimization of the weighted and unweighted single-level problems could be achieved using either an implicit enumeration, DP procedure or by a reduction to an assignment problem formulation. No effective optimization procedure exists in the literature for solving any of the multi-level problems.

CHAPTER 3 MINIMAX FORMULATION OF MIXED-MODEL, JIT SCHEDULING PROBLEMS

III.1 Introduction

In the previous chapter, various formulations of mixed-model, JIT scheduling problems were reviewed. All of the objective functions introduced dealt with the summation of actual production from the desired level of production over all parts and products at all levels for an entire production run. Due to the nature of these objective functions, these models will henceforth be referred to collectively as the "sum function" models (or simply as the "sum functions"). Heuristic methods have been proposed for the solution of these sum function models and optimization procedures have been developed for only the single level formulations. In this chapter a family of different, yet intuitively similar, scheduling problems, which have not appeared in the literature, will be developed.

III.2 General Minimax Model

This new family of mixed-model, JIT scheduling formulations will all have minimax objective functions. Specifically, following the definitions for the variables previously introduced, the general minimax, mixed-model, JIT scheduling problem is to select the x_{i1k}, $i=1,\ldots,n_1$, $k=1,\ldots,D_1$, which satisfy;

$$\text{Minimize} \quad \max_{i,j,k} \; G_{ij} \left| x_{ijk} - XT_{jk} r_{ij} \right|$$

s.t.

$$x_{ijk} = \sum_{p=1}^{n_1} t_{ijp} x_{p1k} \qquad i=1,\ldots,n_1,\ j=1,\ldots,d_i,\ k=1,\ldots,D_1$$

$$XT_{1k} = \sum_{i=1}^{n_1} x_{i1k} = k \qquad k=1,\ldots,D_1$$

$$XT_{jk} = \sum_{i=1}^{n_j} x_{ijk} \qquad j=2,\ldots,d_i,\ k=1,\ldots,D_1$$

$$x_{i1k} \geq x_{i1(k-1)} \qquad i=1,\ldots,n_1,\ k=1,\ldots,D_1$$

$$x_{i1D_1} = d_{i1}, \quad x_{i10} = 0$$

An additional technical assumption that the weighting factors, G_{ij}, are elements of the set of rational numbers is also made.

Optimization of various usage problem formulations of this minimax model will be examined in detail in the remainder of this study, as the usage goal has been identified as the most important problem to be solved for scheduling in the JIT environment (Monden 1983; Hall 1983; Miltenburg 1989). Where applicable, efficient optimization procedures will be developed for solving the formulations considered.

The same reasoning that applied to the use of weighting factors for the sum functions, also applies to their use for minimax problems (see section II.3.2). When weighting factors other than 1 (or 0) are used, an implied arbitrary decision has been made as to the value of the weights used. The solution to these weighted problems depends very heavily upon the weighting factors chosen. For this study, 'true' (in an unbiased sense) usage problems will

always assign a weight of 1 (or 0) to each part and product level and these problems will be referred to as 'unweighted' problems. [In the unweighted case, for the production levels under consideration, G_{ij} is set to the value '1' and for those levels not to be considered by the problem, G_{ij} is set to '0']. In these unweighted problem instances, the objective function value depends only upon the 'true' usage of each part and product. Henceforth, when a usage problem is referred to it will generally apply to an unweighted problem instance, but the distinction between the weighted and unweighted cases will always be made clear.

The minimax objective function minimizes the absolute deviation of the actual production over all parts and products at all levels for the entire schedule from some desired amount of production (i.e. a part or product's relative, cumulative level of production). Analogously to the case of the general sum function model of section II.2, alternative minimax formulations of the objective function could be considered. However, the objective function above has been selected for a variety of reasons. The points listed below apply to the unweighted version of the usage problem (i.e. G_{ij}=0 or 1 for all i=1,...,n_1, j=1,...,d_i), however, analogous extensions to the weighted versions could be made.

Firstly, the value of the objective function has a 'real life' interpretation. In the unweighted case, this value has one of two meanings;

(i) If the maximum deviation occurs at the product level (level 1), then the value corresponds to the product with the maximum overproduction/ underproduction in the entire schedule. This value corresponds to either the

maximum number of units of this product to be held in inventory or the maximum number of units of shortage that would occur for this schedule.

(ii) If the maximum deviation occurs at a lower level (level 2 or lower), then this value corresponds to the part with the largest deviation between its actual production and its desired, relative level of production. This relative amount of production corresponds to the proportion of the cumulative production at the particular level which should be of this part. As production at the lower levels depends necessarily and implicitly upon the sequence of assembly at the product level (because of the pull nature of the JIT process), the products should be sequenced in such a fashion as to keep the number of each type of part produced at each level as close to its relative level of production as possible; thereby ensuring a constant rate of part usage. Then the value of the objective corresponds to the maximum number of units of overproduction/ underproduction of a part from its desired level of production created by the particular sequence of product assembly.

Note that in both cases, the objective value corresponds to the maximum number of units of overproduction or underproduction for a given schedule. As this measurement is made in units of product or parts, the objective has an applicable, physical interpretation.

Secondly, the value of the objective function places a bound on the problem. This bound states that at no time does the overproduction/ underproduction of the assembly process exceed this amount and, thus, the overproduction/ underproduction for all parts and products will always be no greater than this value for the entire planning horizon. This bound could be used to estimate either the maximum requirement of storage space to hold

excess inventory or as the maximum backorder demand for any part or product. This objective value has some inherent, applicable interpretation.

Thirdly, the goal of the minimax is to keep the deviation of actual production from desired production at a minimum for all parts and products at all stages. It achieves this by minimizing the maximum deviation value that occurs over the entire schedule. The level schedule is created by ensuring that the production of all parts and products never deviates from their desired level by more than this bound. This type of schedule is robust, in the sense that production of all parts and products is close to the desired level and no part or product can fall outside of the minimax value. In contrast, the sum functions create their level schedules by minimizing the sum of the squared deviations of actual production from desired production. The idea behind the use of the sum function is that deviations about the desired levels of production for all the products collectively is low. However, there is no guarantee that all of the products will always be close to their desired levels. Because the aim is to minimize the sum, a case may be encountered in which the deviation for one part or product is large (i.e. larger than the bound of the minimax objective) and all of the other parts and products are very close to their desired levels. This highlights the essential difference between how these two types of functions create their respective level schedules.

Fourthly, this type of JIT scheduling model has not been studied before and is of interest for its own merits. The minimax problem has mathematical properties inherent in it which are far different from those of the sum functions considered previously. As any means of improving the efficiency of production has merit (see section I.1), this type of problem warrants

investigation as it provides insight into the workings of JIT scheduling problems. Some of the open questions within the mixed-model, JIT assembly environment may be answered through the interpretation of the minimax function and these questions can perhaps be answered with applicable, meaningful answers which have a direct, physical interpretation. Furthermore, by studying the JIT scheduling problem from a new perspective, new types of questions arise and, perhaps, questions associated with the inherent difficulties encountered by the sum function approach may be answered.

III.3 Minimax Problem Formulations

Similarly to the cases discussed in the previous chapter, various formulations of the general minimax problem are of direct interest. These special problem formulations will be described in this section.

III.3.1 Single Level Problems

As has been indicated, the primary goal for scheduling in a JIT system is to maintain a constant rate of usage for all parts used within the system. Recalling that part usage is determined by the sequencing of the products on the final assembly process, if the assumption is made that each product requires the same number and mix of lower level parts, then a constant rate of part usage is achieved by constructing a sequence which maintains a constant rate of production for each product. The minimax objective function for this problem is;

$$\text{Minimize} \quad \max_{i,k} \; G_{i1} \left| x_{i1k} - XT_{1k} r_{i1} \right| \qquad i=1,\ldots,n_1,\ k=1,\ldots,D_1$$

If $G_{i1}=1$, $i=1,\ldots,n_1$, then this problem is the (unweighted) single-level usage

problem. If $G_{i1} \neq 1$, for at least one i, then this becomes the weighted single-level usage problem.

III.3.2 Multi-Level Problems

When each product does not require the same number and mix of parts then the effect that the product assembly sequence at the highest level has on the part usage at the lower levels must be determined. A product sequence must be constructed that maintains a constant rate of usage for all parts and products. In this instance, the objective function used is the one from the general model. That is to;

$$\text{Minimize} \quad \max_{i,j,k} \; G_{ij} \left| x_{ijk} - XT_{jk} r_{ij} \right| \qquad i=1,\ldots,n_1, \;\; k=1,\ldots,D_1, \;\; j=1,\ldots,L.$$

Thus, one tries to minimize over the entire production schedule the maximum deviation of all parts and products. If $G_{ij}=1$, for all $i=1,\ldots,n_1$, $j=1,\ldots,d_i$, then this problem is the (unweighted) multi-level usage problem. If $G_{ij} \neq 1$, for at least one i,j, then this becomes the weighted multi-level usage problem.

III.3.3 Multi-Level Problems with the "Pegging" Assumption

An analogous, pegging scheme to that of Goldstein & Miltenburg (1988) could be considered for a minimax objective function which can considerably simplify the computation requirements for the sequencing of products in a multi-level system. Once again, a distinction is made between t_{ijh} and t_{ijl}, $h \neq l$, for each output i at level j. Parts of output i at level j are dedicated to the level 1 product into which they will be assembled, thereby separating parts into distinct groups for each product. Thus although a particular part may be common to more than one product, pegging effectively partitions such a part into distinct groups of this part. The effect is that each group can be considered

separately as analogous to a part in a problem in which each part is assembled into only one product. The objective function for the weighted, multi-level problem with pegging is;

$$\text{Minimize} \quad \max_{h,i,j,k} \left\{ W_1 | x_{h1k} - kr_{h1} | \ , W_j | x_{h1k} t_{ijh} - kt_{ijh} r_{h1} | \right\}$$

$$h=1,...,n_1,\ i=1,...,n_1,\quad k=1,...,D_1,\quad j=2,...,L,$$

where W_j, j=1,...,L, is the weighting factor for level j. Recalling that $XT_{1k}=k$, that $t_{i1h}=1$ if i=h and 0 otherwise, and recognizing that $t_{ijh} \geq 0$, then this objective may be rewritten as;

$$\text{Min} \quad \max_{h,i,j,k} \left\{ W_j (t_{ijh}) \, | x_{h1k} - XT_{1k} r_{h1} | \right\}$$

Letting $G_{h1} = \max_{i,j} \left\{ W_j (t_{ijh}) \right\}$, then this objective becomes

$$\text{Min} \quad \max_{h,k} \left\{ G_{h1} \, | x_{h1k} - XT_{1k} r_{h1} | \right\}$$

Hence, with the appropriate representation of weighting factors, the minimax weighted, multi-level usage problem under the pegging assumption can be reduced to a weighted, single-level usage problem.

III.4 Nearest Integer Points

The ideal production for product i in level 1 at any stage k is kr_{i1}. If one could always produce at this ideal production level for <u>each</u> product then the value of the objective function for a single level problem (both weighted and unweighted) would be zero. As the kr_{i1} values are generally noninteger and the x_{i1k} are restricted to holding integer value, and if one could produce in such a fashion that each x_{i1k} was the integer value closest to the value kr_{i1} for each product i at each stage k, then an optimal production schedule could easily be derived. The subsequent theorem and its corollary show that finding such nearest integer points to the kr_{i1} values does not guarantee the feasibility of a

production schedule and that producing at the levels implied by these nearest integer points need not be feasible.

III.4.1 Nearest Integer Point Theorem

Firstly the maximum norm of a vector will be defined. Namely, if $\mathbf{A}=\begin{bmatrix} a_1 \\ a_2 \\ . \\ a_n \end{bmatrix}$ is a vector with n rows, then $\|\mathbf{A}\|=\max_{1\le i\le n}\{|a_i|\}$ is defined to be maximum norm of **A**. Let **M** and **Y** be points in which $\mathbf{Y}=(y_1,y_2,...,y_n)\varepsilon R^n$ and $\mathbf{M}=(m_1,m_2,...,m_n)\varepsilon Z^n$, where R is the set of real numbers and Z is the set of non-negative integers, such that $k=\sum_{i=1}^{n} y_i$ and $k=\sum_{i=1}^{n} m_i$. If $\mathbf{P}\varepsilon Z^n$ is any integer point, then **M** is defined to be the *nearest integer point* to **Y** if

$$\|\mathbf{M}-\mathbf{Y}\| = \max_{1\le i\le n}\{|m_i-y_i|\} \le \|\mathbf{P}-\mathbf{Y}\|$$

for all **P**.

Lemma III.1

For the nearest integer point **M**, there exists a $\delta\varepsilon[0,1]$ such that $|m_i-y_i|\le\delta$ for all i, where $\mathbf{Y}\varepsilon R^n$ and $\mathbf{M}\varepsilon Z^n$.

Proof

By definition it is known that

$$k=\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} m_i. \qquad (3.1)$$

Then the following steps may be applied.

(i) Suppose that there exists an i such that $m_i-y_i>1$. Then, due to (3.1), there must also exist a j such that $m_j-y_j<0$.

(ii) Let $m_i^* = m_i - 1$ and $m_j^* = m_j + 1$. This implies that $m_i^* - y_i = m_i - 1 - y_i > 0$ and $m_j^* - y_j = m_j + 1 - y_j < 1$.

(iii) Now let $m_i = m_i^*$, and if $m_i - y_i > 1$ return to step (i).

(iv) Repeat the process in steps (i), (ii) and (iii) for all k where $m_k - y_k > 1$ until $m_i - y_i < 1$ for all i. Once this has been achieved, move to step (v).

(v) Now suppose that there exists an i such that $m_i - y_i < -1$. Then, due to (3.1) there must also exist a j such that $m_j - y_j > 0$.

(vi) Let $m_i^* = m_i + 1$ and $m_j^* = m_j - 1$.

Now, $m_i - y_i < -1$ which implies that $m_i + 1 - y_i < 0$ and, thus, $m_i^* - y_i < 0$.

Also, $m_j - y_j > 0$ which implies that $m_j - 1 - y_j > -1$ and, thus, $m_j^* - y_j > -1$.

(vii) Now let $m_i = m_i^*$ and if $m_i - y_i < -1$ return to (v).

(viii) Repeat the process of (v), (vi) and (vii) for all k where $m_k - y_k < -1$, until $m_i - y_i > -1$ for all i. Once this is achieved, then terminate the process.

The above procedure ensures that an integer point exists such that; $-1 \le m_i - y_i \le 1$, implying that

$$0 \le | m_i - y_i | \le 1 \quad , \qquad \text{for all i.} \tag{3.2}$$

Setting $\delta = \max_{1 \le i \le n} \{ |m_i - y_i| \}$ then by definition $\delta = \| \mathbf{M} - \mathbf{Y} \|$ and from (3.2) $\delta \in [0,1]$. □

Hence, without loss of generality, it can be assumed that there exists an optimal **M** satisfying $\| \mathbf{M} - \mathbf{Y} \| \le 1$.

Theorem III.1

For points $\mathbf{Y} = (y_1, y_2, \ldots, y_n) \in R^n$ and $\mathbf{M} = (m_1, m_2, \ldots, m_n) \in Z^n$, the following statements are equivalent.

(I) **M** is the nearest integer point to **Y**.

(II) For all i and j, $(m_i - y_i) - (m_j - y_j) \leq 1$.

(III) There exists an $\alpha \varepsilon [0,1]$ such that for all i we have $\alpha - 1 \leq m_i - y_i \leq \alpha$.

Proof

The proof will be structured as (I)→(II)→(III)→(I).

(I)→(II)

(I) states that **M** is the nearest integer point. Then for any other integer point **P**, $\| \mathbf{M}-\mathbf{Y} \| = \max_{1 \leq i \leq n} \{|m_i - y_i|\} \leq \| \mathbf{P}-\mathbf{Y} \| = \max_{1 \leq i \leq n} \{|p_i - y_i|\}$.

It is required to show that in **M**, for any i and j,

$$(m_i - y_i) - (m_j - y_j) \leq 1. \tag{3.3}$$

Assume that u and v are such that $m_u - y_u = \max_{1 \leq k \leq n} \{m_k - y_k\}$ and that $m_v - y_v = \min_{1 \leq k \leq n} \{m_k - y_k\}$. Now $m_u - y_u \geq 0$ and $m_v - y_v \leq 0$ because of (3.1).

Let **P** be the integer point such that $p_v = m_v + 1$, $p_u = m_u - 1$ and $p_k = m_k$ for $k \neq u,v$. [Note: If there were tied values in the determination of both u and v in **M**, then set **M=P** and redetermine **P** as in the preceding step. Repeat this process until there are no longer ties for both u and v in **M**.]

Suppose that (3.3) does not hold and instead,

$$(m_u - y_u) - (m_v - y_v) > 1$$

implying that

$$(m_u - y_u) > (m_v - y_v) + 1. \tag{3.4}$$

Now by the lemma III.1, $m_v - y_v \geq -1$ and combining this with the assumption for v above, $0 \geq m_v - y_v \geq -1$ and thus, $1 \geq (m_v - y_v) + 1 \geq 0$. Therefore, combining this result with (3.4),

$$(m_u - y_u) > (m_v - y_v) + 1 \geq 0$$

thus,

$$(m_u - y_u) > p_v - y_v \geq 0. \tag{3.5}$$

Also from the definition of **M**, $m_u - y_u \le 1$, and combining this with the assumption for u above yields $1 \ge m_u - y_u \ge 0$ and thus, $0 \ge (m_u - y_u) - 1 \ge -1$.

Therefore, combining this result with (3.4),

$$0 \ge (m_u - y_u) - 1 > (m_v - y_v)$$

and

$$0 \ge (p_u - y_u) > (m_v - y_v). \qquad (3.6)$$

Now, from (3.5) and (3.6), $\| \mathbf{M}-\mathbf{Y} \| > \| \mathbf{P}-\mathbf{Y} \|$ which is a contradiction because **M** is the nearest integer point. That is, (3.4) cannot hold if **M** is the nearest integer point. Hence, it must be the case that $(m_u - y_u) - (m_v - y_v) \le 1$.

Now (3.3) holds for this choice of u and v, and by the definition of u and v,

$$(m_k - y_k) - (m_l - y_l) \le (m_u - y_u) - (m_v - y_v)$$

for any other k and l. Thus (3.3) must hold for any choice of i and j.

(II)→(III)

Given that $(m_i - y_i) - (m_j - y_j) \le 1$ for any i and j it must be shown that there exists an $\alpha \varepsilon [0,1]$ such that $\alpha - 1 \le m_l - y_l \le \alpha$ for all l. As above, choose u such that $m_u - y_u = \max_{1 \le k \le n} \{m_k - y_k\}$ and set $\alpha = m_u - y_u$. Now (II) implies that $(m_u - y_u) - (m_j - y_j) \le 1$ for all j. Hence,

$$\alpha - 1 \le \alpha - [(m_u - y_u) - (m_j - y_j)] = m_j - y_j \le \alpha \quad \text{for all j}$$

by the choice of α.

Noting that $\sum_{l=1}^{n} (m_l - y_l) = \sum_{l=1}^{n} m_l - \sum_{l=1}^{n} y_l = k - k = 0$, then

$$\alpha - 1 \le \min_{1 \le l \le n} \{m_l - y_l\} \le \frac{1}{n} \sum_{l=1}^{n} (m_l - y_l) = 0,$$

implying $\alpha \le 1$.

Furthermore,

$$\alpha = \max_{1\le i\le n} \{m_i - y_i\} \ge \frac{1}{n}\sum_{i=1}^{n} (m_i - y_i) = 0,$$

thus $$0 \le \alpha \le 1.$$

(III)→(I)

Suppose that **M** satisfies (III) and that **P** is the nearest integer point. Then there is some i such that

$$p_i \ge m_i + 1 \tag{3.7}$$

and there is some j such that

$$p_j \le m_j - 1 . \tag{3.8}$$

By (III) there must be an $\alpha\varepsilon[0,1]$ such that for all i $\alpha-1 \le m_i - y_i \le \alpha$. Because $\alpha\varepsilon[0,1]$, $0 \le |\alpha| \le 1$ and $|\alpha - 1| = 1-\alpha$, where $0 \le 1-\alpha \le 1$. Set $\delta = \max\{|\alpha|, |\alpha-1|\}$ where $\delta\varepsilon[0,1]$. Then, given (III), $\alpha-1 \le m_i - y_i$, which when combined with (3.7) implies,

$$\alpha - 1 + 1 \le m_i - y_i + 1 \le p_i - y_i,$$

thus $$\alpha \le p_i - y_i. \tag{3.9}$$

Also from (III), $m_j - y_j \le \alpha$ which when combined with (3.8) implies that,

$$p_j - y_j \le m_j - y_j - 1 \le \alpha - 1$$

thus $$p_j - y_j \le \alpha - 1. \tag{3.10}$$

Now, from (III), it must be that,

$$\| \mathbf{M}-\mathbf{Y} \| = \max_{1\le i\le n} \{|m_i - y_i|\} \le \max\{|\alpha|, |\alpha-1|\} = \delta.$$

And from (3.9) and (3.10),

$$\| \mathbf{P}-\mathbf{Y} \| \ge \max\{|p_i - y_i|, |p_j - y_j|\} \ge \max\{|\alpha|, |\alpha-1|\} = \delta.$$

Therefore, $\| \mathbf{M}-\mathbf{Y} \| \le \delta \le \| \mathbf{P}-\mathbf{Y} \|$, which implies that if **P** is a nearest integer point, then **M** must also be a nearest integer point. □

Corollary III.1

Let $\mathbf{r} = (r_{11}, r_{21}, \ldots, r_{n_1 1})$ be the vector of demand ratios for level 1. Let $\mathbf{Y}_k = k\mathbf{r}$ and let $\mathbf{Y}_{k+1} = (k+1)\mathbf{r}$, where $k \geq 1$ is an integer. Let $y_{ik} = kr_{i1}$ and $y_{ik+1} = (k+1)r_{i1}$ be the i^{th} element in $\mathbf{Y}_k$ and $\mathbf{Y}_{k+1}$, respectively. Let **M** be the nearest integer point to $\mathbf{Y}_k$ and let **N** be the nearest integer point to $\mathbf{Y}_{k+1}$. Then for each product i, $| m_i - n_i | \leq 1$.

Proof

The results of theorem III.1 can be applied directly to the proof of Miltenburg (1989, Appendix 1, p205). This proof, which is by contradiction, is restated below:

Suppose that $| m_i - n_i | \leq 1$ does not hold for some product i. Then, for some i, either (i) $m_i - n_i \geq 2$ or (ii) $m_i - n_i \leq -2$. Note that $\sum_{j=1}^{n} m_j = k$ and $\sum_{j=1}^{n} n_j = k+1$.

(i) Set i such that $m_i - n_i \geq 2$. Then,

$$-1 = \sum_{j=1}^{n} m_j - \sum_{j=1}^{n} n_j = \sum_{j=1}^{n} (m_j - n_j) = \sum_{j \neq i} (m_j - n_j) + m_i - n_i \geq \sum_{j \neq i} (m_j - n_j) + 2$$

hence,
$$-3 \geq \sum_{j \neq i} (m_j - n_j).$$

For some $j \neq i$, $m_j - n_j < 0$, and thus $m_j - n_j \leq -1$ because m_j and n_j are integer. For this j,

$$\begin{aligned}
3 &\leq (m_i - n_i) - (m_j - n_j) \\
&= [\,(m_i - kr_i) - (m_j - kr_j)\,] + [\,(n_j - (k+1)r_j) - (n_i - (k+1)r_i)\,] + r_j - r_i \\
&= [\,(m_i - y_{ik}) - (m_j - y_{jk})\,] + [\,(n_j - y_{j(k+1)}) - (n_i - y_{i(k+1)})\,] + r_j - r_i \\
&\leq 1 + 1 + r_j - r_i && \text{by theorem III.1} \\
&< 1 + 1 + 1 - 0 && \text{because } 0 < r_j, r_i < 1 \\
&= 3
\end{aligned}$$

which is a contradiction.

(ii) Set i such that $n_i - m_i \geq 2$. Then,

$$1 = \sum_{j=1}^{n} n_j - \sum_{j=1}^{n} m_j = \sum_{j=1}^{n} (n_j - m_j) = \sum_{j \neq i} (n_j - m_j) + n_i - m_i \geq \sum_{j \neq i} (n_j - m_j) + 2$$

hence,
$$-1 \geq \sum_{j \neq i} (n_j - m_j).$$

For some $j \neq i$, $n_j - m_j < 0$, and thus $n_j - m_j \leq -1$ because m_j and n_j are integer. For this j,

$$\begin{aligned} 3 &\leq (n_i - m_i) - (n_j - m_j) \\ &= [\,(n_i - (k+1)r_i) - (n_j - (k+1)r_j)\,] - [\,(m_j - kr_j) - (m_i - kr_i)\,] - r_j + r_i \\ &\leq 1 + 1 - r_j + r_i < 1 + 1 - 0 + 1 = 3, \quad \text{which is a contradiction.} \end{aligned}$$

As neither case (i) $m_i - n_i \geq 2$ nor case (ii) $m_i - n_i \leq -2$ can hold, it must be the case that $|\, m_i - n_i \,| \leq 1$. □

Remark III.1

Together theorem III.1 and corollary III.1 prove that $|\, m_i - n_i \,| \leq 1$ but they do not preclude the existence of the case in which $n_i = m_i - 1$. As an assembly process would not destroy products, it is not possible to always produce a schedule that is always at the ideal production level suggested by using the nearest integer points. An example taken from Miltenburg (1989) is equally applicable to the minimax objective. If $\mathbf{r} = \left(\frac{6}{13}, \frac{6}{13}, \frac{1}{13}\right)$, then the nearest integer point for stage k=5 (i.e. the nearest integer point to 5**r**) is **M**= (2,2,1). However, the nearest integer point for stage k=6 (i.e. to 6**r**) is **N**=(3,3,0). In moving from stage 5 to stage 6, the assembly process must create one unit of product 1 and one unit of product 2 while simultaneously destroying one unit of product 3. As only one unit can be assembled in each stage and the assembly process does not destroy products, creating a feasible production schedule using nearest integer points is clearly not feasible for this example. Hence,

scheduling the minimax problem using nearest integer points would prove as computationally intensive as using the approach for the sum functions. □

III.5 Cyclical Solutions

In Miltenburg (1989) and Miltenburg & Sinnamon (1990) a cyclical property is shown to exist for the single- and multi-level sum functions which can be used to reduce the computational effort for constructing production schedules. The underlying concept behind this property is that in a schedule a relatively short production sequence may exist which repeats again and again. This type of recurrent production sequence is also implicitly assumed to exist in the types of scheduling problems considered by Groeflin et al (1989). It can be shown that an analogous cyclical property exists for the minimax functions.

Let $x_{ijk}(\sigma)$ be the number of units of output i at level j produced during stages 1,2,...,k under sequence σ, where $|\sigma|=u$ and $k \leq u$. Let $XT_{jk}(\sigma)$ be the total number of units produced at level j during stages 1,2,...,k under sequence σ. Note that $XT_{jk}(\sigma) = \sum_{i=1}^{n_j} x_{ijk}(\sigma)$. Let $V_{ijk}(\sigma) = G_{ij} \left| x_{ijk}(\sigma) - XT_{jk}(\sigma) r_{ij} \right|$ be the deviation of the i^{th} part at level j in stage k under sequence σ. Define

$$V_k(\sigma) = \left[V_{11k}(\sigma),...,V_{n_1 1k}(\sigma), V_{12k}(\sigma),...,V_{n_L Lk}(\sigma) \right]$$

to be a vector of the deviations of each part i at level j in cycle k under sequence σ. If σ and τ, $|\tau|=v$, are sequences of products over u and v stages, respectively, define $\sigma + \tau$ as concatenation of the two sequences, in the usual way.

Lemma III.2

For any sequences σ, $|\sigma|=u$, and τ, $|\tau|=v$,

$$x_{ij(u+k)}(\sigma+\tau) = x_{iju}(\sigma) + x_{ijk}(\tau)$$

and

$$XT_{j(u+k)}(\sigma+\tau) = XT_{ju}(\sigma) + XT_{jk}(\tau), \quad \text{where } k\varepsilon\{1,2,\ldots,v\}.$$

Proof

Let $\sigma = (p_1,p_2,\ldots,p_u)$ and $\tau = (q_1,q_2,\ldots,q_v)$. Then $\sigma+\tau =$ $(p_1,p_2,\ldots,p_u,p_{u+1},p_{u+2},\ldots,p_{u+v})$, where $p_{u+k} = q_k$, $k = (1,2,\ldots,v)$.

$$\begin{aligned}
x_{ij(u+k)}(\sigma+\tau) &= \sum_{h=1}^{u+k} t_{ijp_h} \qquad k\varepsilon\{1,2,\ldots,v\} \\
&= \sum_{h=1}^{u} t_{ijp_h} + \sum_{h=u+1}^{u+k} t_{ijp_h} \\
&= \sum_{h=1}^{u} t_{ijp_h} + \sum_{h=1}^{k} t_{ijq_h} \\
&= x_{iju}(\sigma) + x_{ijk}(\tau).
\end{aligned}$$

$$\begin{aligned}
XT_{j(u+k)}(\sigma+\tau) &= \sum_{i=1}^{n_j} x_{ij(u+k)}(\sigma+\tau) \qquad k\varepsilon\{1,2,\ldots,v\} \\
&= \sum_{i=1}^{n_j} \left\{ x_{iju}(\sigma)+x_{ijk}(\tau) \right\} \\
&= \sum_{i=1}^{n_j} x_{iju}(\sigma) + \sum_{i=1}^{n_j} x_{ijk}(\tau) \\
&= XT_{ju}(\sigma) + XT_{jk}(\tau). \quad \square
\end{aligned}$$

Lemma III.3

If $V_u(\sigma) = \mathbf{0}$, then $V_{u+k}(\sigma+\tau) = V_k(\tau)$, $k\varepsilon\{1,2,\ldots,v\}$, for any sequence τ.

Proof

Since $G_{ij} \geq 0$ and $V_u(\sigma) = \mathbf{0}$, then for each i,j, $G_{ij} \left| x_{iju}(\sigma) - XT_{ju}(\sigma) r_{ij} \right| = 0$,

which implies that,

$$x_{iju}(\sigma) - XT_{ju}(\sigma)r_{ij}=0 . \qquad (3.11)$$

Now, for each i,j in the vector $\mathbf{V}_{u+k}(\sigma+\tau)$;

$$\begin{aligned} V_{ij(u+k)}(\sigma+\tau) &= G_{ij}\left|x_{ij(u+k)}(\sigma+\tau) - XT_{j(u+k)}(\sigma+\tau)r_{ij}\right| \\ &= G_{ij}\left|\left[x_{iju}(\sigma)-XT_{ju}(\sigma)r_{ij}\right] + \left[x_{ijk}(\tau)-XT_{jk}(\tau)r_{ij}\right]\right| \\ &\qquad\qquad \text{by lemma III.2} \\ &= G_{ij}\left|x_{ijk}(\tau) - XT_{jk}(\tau)r_{ij}\right| \qquad \text{by (3.11)} \\ &= V_{ijk}(\tau) \end{aligned}$$

and thus, $\mathbf{V}_{u+k}(\sigma+\tau) = \mathbf{V}_k(\tau)$. □

Remark III.2

The implications of lemmas III.2 and III.3 are that cyclical solutions may be encountered within a schedule created using the minimax objective. Suppose that an algorithm produces a schedule for which $\mathbf{V}_u(\sigma) = \mathbf{0}$. According to lemma III.3, all subsequent deviations in the schedule are independent of σ providing deviations as if the products scheduled previously in the sequence σ had never been scheduled. Therefore, if an algorithm constructs a sequence in an iterative process such that the choice of which product to sequence depends upon the deviation (current) alone, then it will compose a schedule after σ in the same fashion as it did in the beginning. Hence, a sequence σ, constructed in this iterative fashion, may be repeated again and again. That is, the criterion that sequences product i at stage k will also sequence product i in stage $\sigma+k$, $k \le |\sigma|$. □

However, while a complete schedule could consist of repetitions of some sequence, the length of this particular sequence can only be determined *a posteriori*. For any objective function possessing the cyclical property, the

length of the repeated sequence that minimizes the objective can not be determined until the solution algorithm is actually running. Furthermore, in order to use the cyclical property, an algorithm must be working iteratively either forwards or backwards through the schedule, so that the exact sequence which will be repeated can be determined. Hence, optimal solution techniques, which wish to exploit the cyclical property, must use full, unfactored demand values, d_{i1}, until the first occurrence of $V_u(\sigma) = 0$; after which the number of repetitions of this sequence can be determined because of the cyclical nature of the schedules shown above. Thus, the amount of reduction in the computational effort cannot be determined *a priori*.

Significantly, though, if such an algorithm is employed and it arrives at a stage where the deviation vector equals the zero vector, then the algorithm may be stopped and the final production sequence that the algorithm would construct could be fully determined. Unfortunately, this reduction in computation can only be found while operating with the algorithm on the full, unfactored problem.

III.6 Single Level Assignment Approach

For optimization of the single-level minimax problem, there was hope that the assignment method of Kubiak & Sethi (1989, 1991), described in section II.5.1, might be appropriate for the minimax problems. If a minimax problem could be reformulated into a bottleneck assignment problem, then it could be solved using the algorithm of Gross (1959). However, Kubiak & Sethi require that objective functions be of the form

$$\sum_{k=1}^{D_1} \sum_{i=1}^{n_1} F^i\left(x_{ik} - kr_{i1}\right)$$

where $F^i(\bullet)$ is a non-negative convex, symmetric function satisfying $F^i(0)=0$, $F^i(y)>0$ for $y\neq 0$, $i=1,\ldots,n$, in order for their assignment approach to be applicable. Minimax functions are necessarily of the form $\underset{i,k}{\text{Min Max}}\, F^i\left(x_{ik} - kr_{i1}\right)$. Because there is no summation of the functions, $F^i(\bullet)$, in the minimax problems, Kubiak & Sethi's method is not readily transferable.

Potentially, a similar assignment reduction might exist for the minimax problem. It will be shown below that because no summation takes place in the minimax objective, any assignment reduction must account for the sequence dependence which is inherent in the minimax problem. By contrast, the summation occurring in the functions considered by Kubiak & Sethi effectively eliminates this sequence dependence. An attempt at a bottleneck assignment formulation will be presented.

Recall that (i,j) refers to the j^{th} copy of product i. Let C_{ijk} be cost of placing the j^{th} copy of product i in the k^{th} position of the final production sequence. Define the indicator variable, y^i_{jk}, as

$$y^i_{jk} = \begin{cases} 1 & \text{if (i,j) is sequenced in location k} \\ 0 & \text{else} \end{cases}$$

Then, if the C_{ijk}'s could be calculated, the bottleneck assignment problem for solving the single-level, unweighted problem would be;

$$\text{Min} \quad \max_{i,j,k} \; C_{ijk} y^{i}_{jk} \qquad i=1,\ldots,n_1,\ j=1,\ldots,d_i,\ k=1,\ldots,D_1$$

$$\text{st} \quad \sum_{i=1}^{n_1} \sum_{j=1}^{d_i} y^{j}_{jk} = 1 \qquad k=1,\ldots,D_1$$

$$\sum_{k=1}^{D_1} y^{i}_{jk} = 1 \qquad i=1,\ldots,n_1,\ j=1,\ldots,d_i$$

The obvious choice for the cost function of (i,j) would be $C_{ijk}=|j-kr_{i1}|$, as it is the maximum deviation of some copy j from its desired level of production which is of interest. Unfortunately, this cost function can be shown to produce incorrect objective values.

Consider an example in which $D_1=17$ and focus on a product i with $d_{i1}=3$. Construct a production schedule in which the first copy of product i is sequenced in position 2, the second copy is sequenced in position 15 and the third copy is sequenced in position 16. If the cost function above is used, then the deviations attributable to this schedule for this particular product would be; $C_{i1(2)}=\left| 1-2\left(\frac{3}{17}\right)\right|=.647$, $C_{i2(15)}=\left| 2-15\left(\frac{3}{17}\right)\right|=.647$, $C_{i3(16)}=\left| 1-16\left(\frac{3}{17}\right)\right|=.176$. Under this cost function, the maximum deviation for product i would be .647. However, this is not the true maximum deviation attributable to this product because at position k=14,

$$\left| x_{i1k}-kr_{i1}\right| = \left| 1-14\left(\frac{3}{17}\right)\right| = 1.471 .$$

This demonstrates the key problem for the minimax case using the above cost function; the penalty is sequence dependent. This penalty depends upon two factors: Firstly on the location (the k) in which the (i,j) is scheduled and, secondly, on the scheduled location of the previous copy (i,j-1). Both of these factors affect the penalty cost and both factors must be considered simultaneously.

This sequence dependence is eliminated from the calculation of the penalty costs for the sum functions. The penalty applied to each (i,j) depends only upon the location (k) in which it is sequenced for production. One can calculate the costs attributable to (i,j) by knowing its position in the sequence and the optimal location for this copy, Z_{ij}^*. The penalty attributable to (i,j) does not depend upon the (explicit) sequencing of any other copy of product (i',j'), $j' \neq j$. The reduction to the assignment problem for the sum function works because the costs are expressed in terms of differences between consecutive copy deviation functions and that the summation of all of these costs for all of the copies of each product will be the same value as the original sum function objective (when the infimum, which is a constant corresponding to the lower bound of the sum objective, is added back to this value). The key is the summing of all of the non-negative, symmetric, convex cost functions for each copy of each product. The sequence dependent nature, inherent in the minimax objective, is thereby eliminated by the summation in the sum functions.

Unfortunately the summation reduction for calculating penalty costs, which eliminates sequence dependence, is not transferable to the minimax case since no summation of the penalties occurs. Hence, a reduction to a bottleneck assignment formulation is not readily forthcoming for even the simplest minimax problem. Eliminating the sequence dependent nature of the minimax problems must be overcome if a bottleneck assignment method is to be implemented. If one is to optimize the minimax problems, then other approaches must be developed. These approaches will be described in the subsequent chapters.

CHAPTER 4 SINGLE LEVEL OPTIMIZATION

IV.1 Introduction

It was previously discussed that Miltenburg (1989) had introduced a quadratic integer programming formulation for levelling the schedule of the mixed-model, JIT assembly process with the assumption that the products required approximately the same number and mix of parts. This assumption permits the problem to be treated as a single-level problem, because levelling the schedule of final assembly simultaneously achieves an even rate of part usage in all of the other production levels. It is mentioned in Miltenburg (1989) and Miltenburg and Sinnamon (1989) that Toyota's Goal-Chasing Method is also based upon balancing the schedule of a single production level. This chapter focuses specifically on balancing the schedule for the special case of the minimax model; the unweighted, single-level problem.

IV.2 Mathematical Model

As only the product level (level 1) is of interest in this restricted model, the subscript indicating the production level is unnecessary in the mathematical formulation. Because of this, the notation of the general model may be restated for dealing solely with the product level. Thus, assume that there are n different products to be produced within the planning horizon with respective demands $d_1, d_2, \ldots, d_n$, providing a total demand of $D = \sum_{i=1}^{n} d_i$ units. An implied time horizon

of D time units can be inferred, where one copy of product i, i=1,2,..,n must be produced in each time period. If $r_i = \frac{d_i}{D}$ then the level scheduling objective is to keep the total production per time period as close to r_i as possible. Ideally (recalling that $k=XT_{1k}$) kr_i units of product i should be produced in the first k time periods (k=1,2,...D).

If x_{ik}, i=1,2,..,n, k=1,2,..,D, represents the total production of product i in time periods 1 through k, then the model can be written as:

$$\text{Minimize } \max_{i\ k} |x_{ik} - kr_i| \qquad i=1,2,..,n,\ k=1,2,..,D$$

s.t.

$$\sum_{i=1}^{n} x_{ik} = k \qquad k=1,2,..,D$$

[P1]
$$x_{ik-1} \le x_{ik} \qquad i=1,2,..,n,\ k=1,2,..,D$$

$$x_{i0}=0 \qquad i=1,2,..,n$$

$$x_{iD}=d_i \qquad i=1,2,..,n$$

$$x_{ik} \ge 0 \text{ and integer} \qquad i=1,2,..,n,\ k=1,2,..,D$$

The previously used "sum of deviations" objective functions aimed to produce "smooth" schedules on the average. This did not preclude, however, the possibility of relatively large deviations occurring in certain time periods. In contrast, this minimax objective looks for a "smooth" schedule in <u>every</u> time period. This objective may also possess a more applicable, physical interpretation than that of the sum function. Here, the objective function value provides the maximum overproduction/underproduction (the maximum inventory or shortage) from the desired level of production that occurs at any time during the schedule.

IV.3 Reduction to Release Date/ Due Date Decision Problem

Let (i,j) be defined as the j^{th} copy of product i, i=1,2,..,n, j=1,...,d_i. If (i,j) is produced in period k then x_{ik}=j and the penalty associated with (i,j) in period k is

$$g_j^i(k) = | \; j \; - \; k r_i \; | \qquad i=1,2,..,n, \; j=0,...,d_i, \; k=1,2,..,D. \tag{4.1}$$

(Note, here that j=0 has been introduced to account for the periods k in which x_{ik}=0.) It is possible to plot the individual penalty functions, $g_j^i(k)$, for each copy of a given product over all k (where k is shown as continuous time for ease of exposition), as has been done in fig. IV.1 for d_i=7 and D=20.

Letting k_j be the completion time for (i,j), then the penalties "attributable" to product i for a particular schedule are given by

$$f_j^i(k) = \begin{cases} g_j^i(k) & k_j \le k < k_{j+1} \\ 0 & else \end{cases} \tag{4.2}.$$

Letting $f^i(k)=\underset{0\le j\le d_i}{Max} f_j^i(k)$, then $f^i(k)$ is the "envelope" of the $f_j^i(k)$ functions and possesses a saw-tooth shape, as shown in fig. IV.2. It is clear that this envelope is non-convex for each i. In effect, the deviations are first measured on the g_0^i curve and the deviations on this curve are measured until the first copy is produced. There is then a shift onto the g_1^i curve and the deviations "attributable" to the first copy are measured until the second copy is produced. This pattern of "jumping" from one copy's deviation curve to the subsequent copy's curve proceeds until all of the product's copies have been scheduled. Note that minimizing the objective function in P1 is equivalent to minimizing $f^i(k)$. That is;

$$Z = \text{Minimize} \underset{i \; k}{Max} \; |x_{ik} - k r_i| = \text{Minimize} \underset{i \; k}{Max} \; f^i(k). \tag{4.3}$$

Henceforth, when references to minimizing the objective function are made, specifically, those functions created by (4.2) will be under examination.

Denote a target value for the objective function by the variable T. Hence, if a sequence could be created in which each (i,j) has a completion time k_j such that $f_j^i(k) \le T$ for $k \in [k_j, (k_{j+1} - 1)]$, then $\text{Min} \underset{i}{\text{Max}} \underset{k}{} |x_{ik} - kr_i| \le T$. Necessarily for optimization, the smallest T possible for which such a feasible sequence exists must be determined.

If a copy starts production at time t-1 and completes production at time t, assuming that this is copy (i,j), then the deviation for this copy will be $g_j^i(t) = | j - tr_i |$. For a sequence to be feasible, some copy must start at time t=0 and each remaining copy will start at some time t, t=1,...,D-1, such that the D^{th} unit will complete production at time t=D and no two copies share the same starting time. Any fixed target value T allows the calculation of a release date and a due date for a specific copy of a product. The release date and due date for this copy can be thought of as the lower and upper limits, respectively, of the range of locations in the production sequence where the copy must be produced if the desired target value is not to be exceeded. For target value T, (i,j) can start at $k \ge 1$ if,

$$f_j^i(k+1) = j-(k+1)r_i \le T$$

and cannot start before k if,

$$g_j^i(k) = j-kr_i > T.$$

Letting E(i,j) be the earliest starting time for (i,j) then it must satisfy:

$$j - E(i,j)r_i > T \qquad (4.4)$$

and

$$j - (E(i,j)+1)r_i \le T. \qquad (4.5)$$

Thus, the earliest starting time is the unique integer satisfying;

$$\left(\frac{1}{r_i}\right)\left[\, j - T \,\right] - 1 \le E(i,j) < \left(\frac{1}{r_i}\right)\left[\, j - T \,\right] \qquad (4.6)$$

Note that (i,j) may start at time k=0 if;

$$j-(k+1)r_i \le T$$

implying

$$j \le T + r_i \,. \qquad (4.7)$$

Letting L(i,j) be the latest starting time for (i,j) then the latest time, L(i,j)≤D-2, at which (i,j) can start and still satisfy the target value must satisfy the following two inequalities:

$$L(i,j)r_i - (j-1) \le T \qquad (4.8)$$

and

$$(L(i,j)+1)r_i - (j-1) > T. \qquad (4.9)$$

Thus, the latest starting time is the unique integer satisfying

$$\left(\frac{1}{r_i}\right)\left[\, (j - 1)+T \,\right] - 1 < L(i,j) \le \left(\frac{1}{r_i}\right)\left[\, (j - 1)+T \,\right] \qquad (4.10)$$

Note that copy (i,j) may complete processing at time k=D and therefore may start at time k-1=D-1 if;

$$(k-1)r_i - (j-1) \le T,$$

implying

$$d_i - r_i - j + 1 \le T \qquad \text{because } r_i = \frac{d_i}{D} \qquad (4.11)$$

For a given T, early and late starting dates can be calculated for each copy of each product in a one pass procedure and, hence, could be constructed in $O(D)$ time. This process can be visualized from fig. IV.1 by moving along the target line from left-to-right and finding where this line intersects the $g_i^j(k)$ lines. The applicable early and late start times and the intervals $[E(i,j),L(i,j)]$ are shown at the bottom of the figure. Thus, the decision problem "Is Z≤T?" may be viewed as the problem of determining whether there is a feasible schedule of D

unit time jobs on a single machine with release dates and due dates for each job.

FIGURE IV.1: Level Curves for the Deviation of Ideal Production from Each Copy over all Time Periods

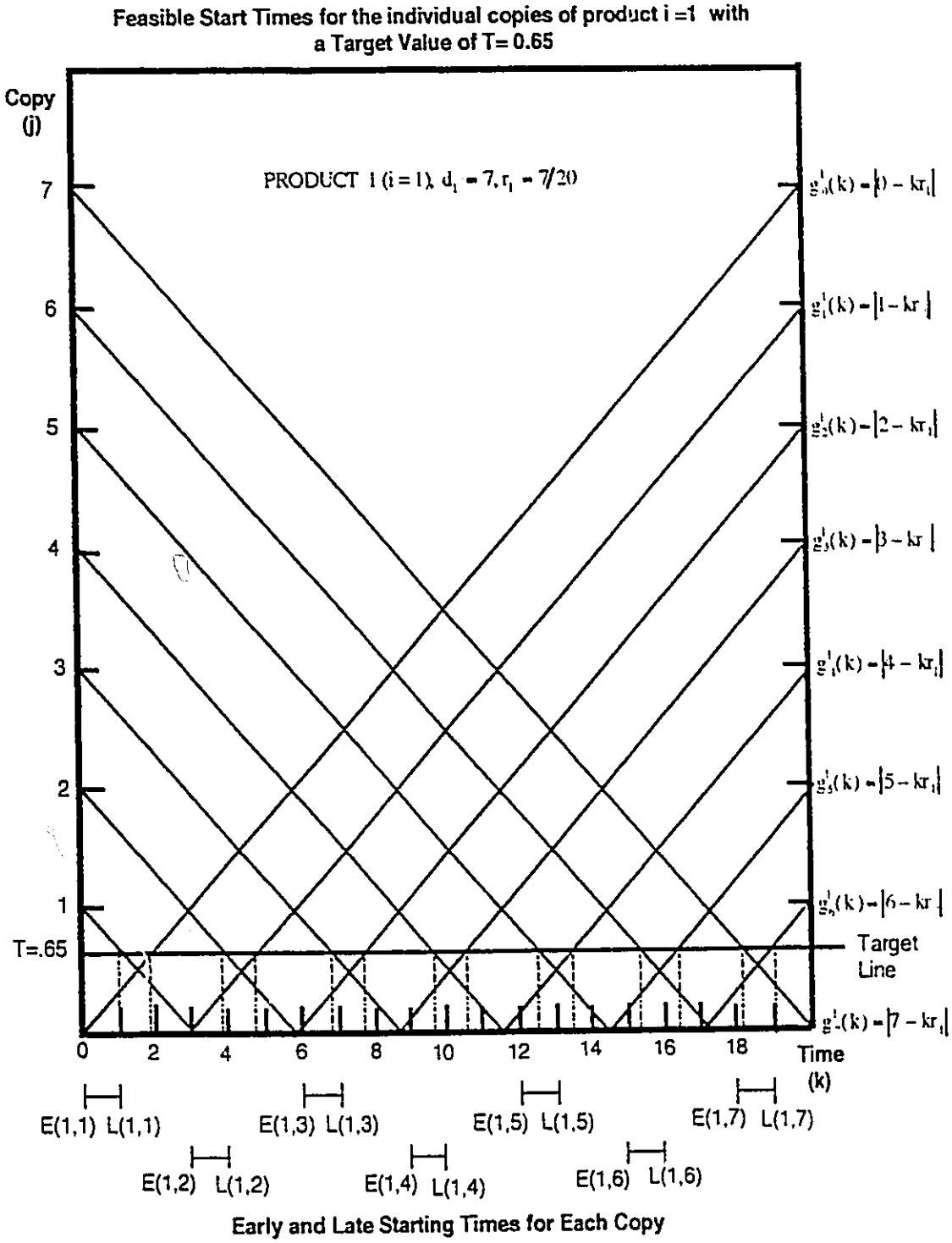
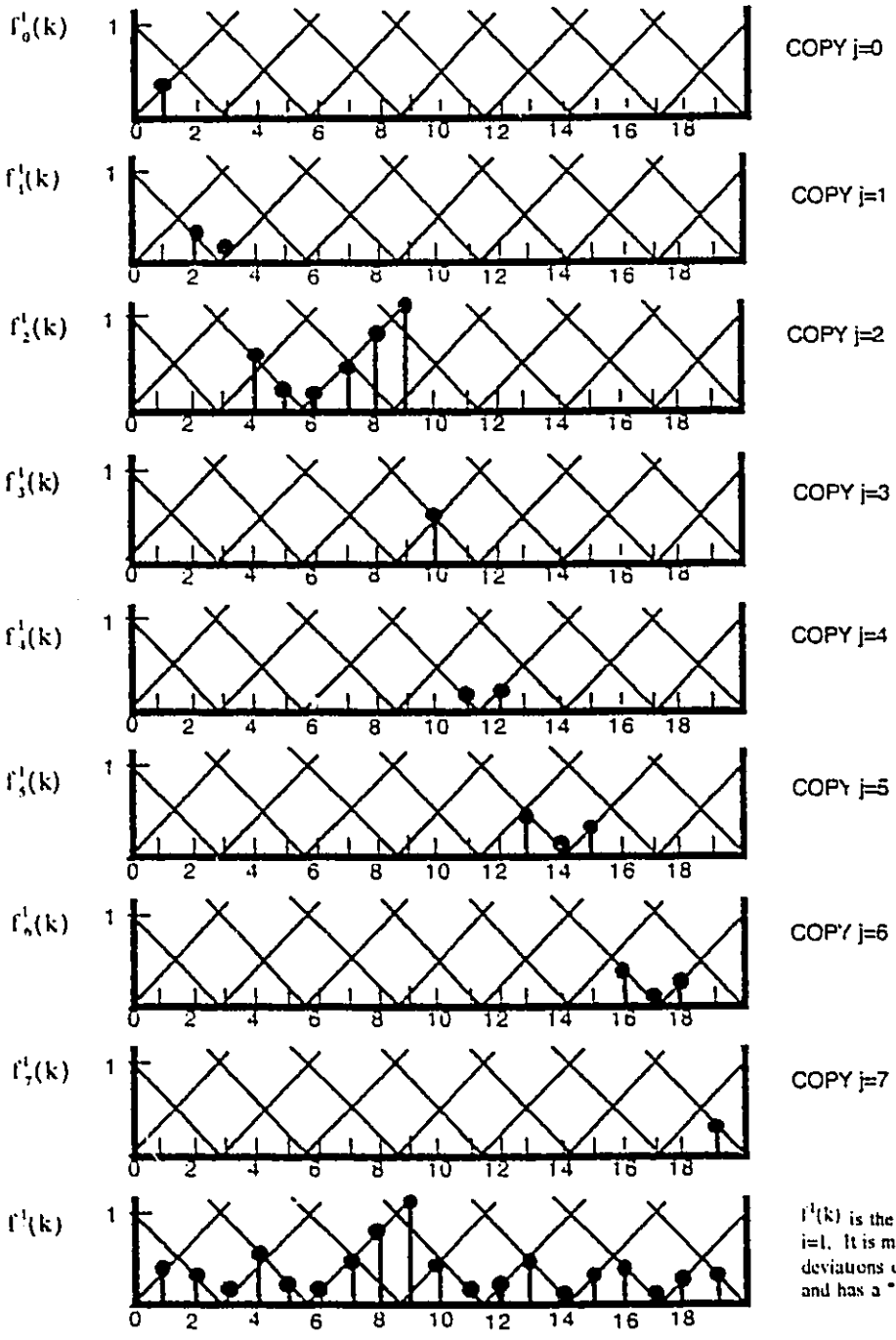


FIGURE IV.2 Deviations "Attributable" to Each Copy of a Product

Given a schedule for product 1, where: $k_1 = 2, k_2 = 4, k_3 = 10, k_4 = 11, k_5 = 13, k_6 = 16, k_7 = 19$ the graphs show the deviation for the individual copies. The "spikes" show that the deviation is measured only at integral times.



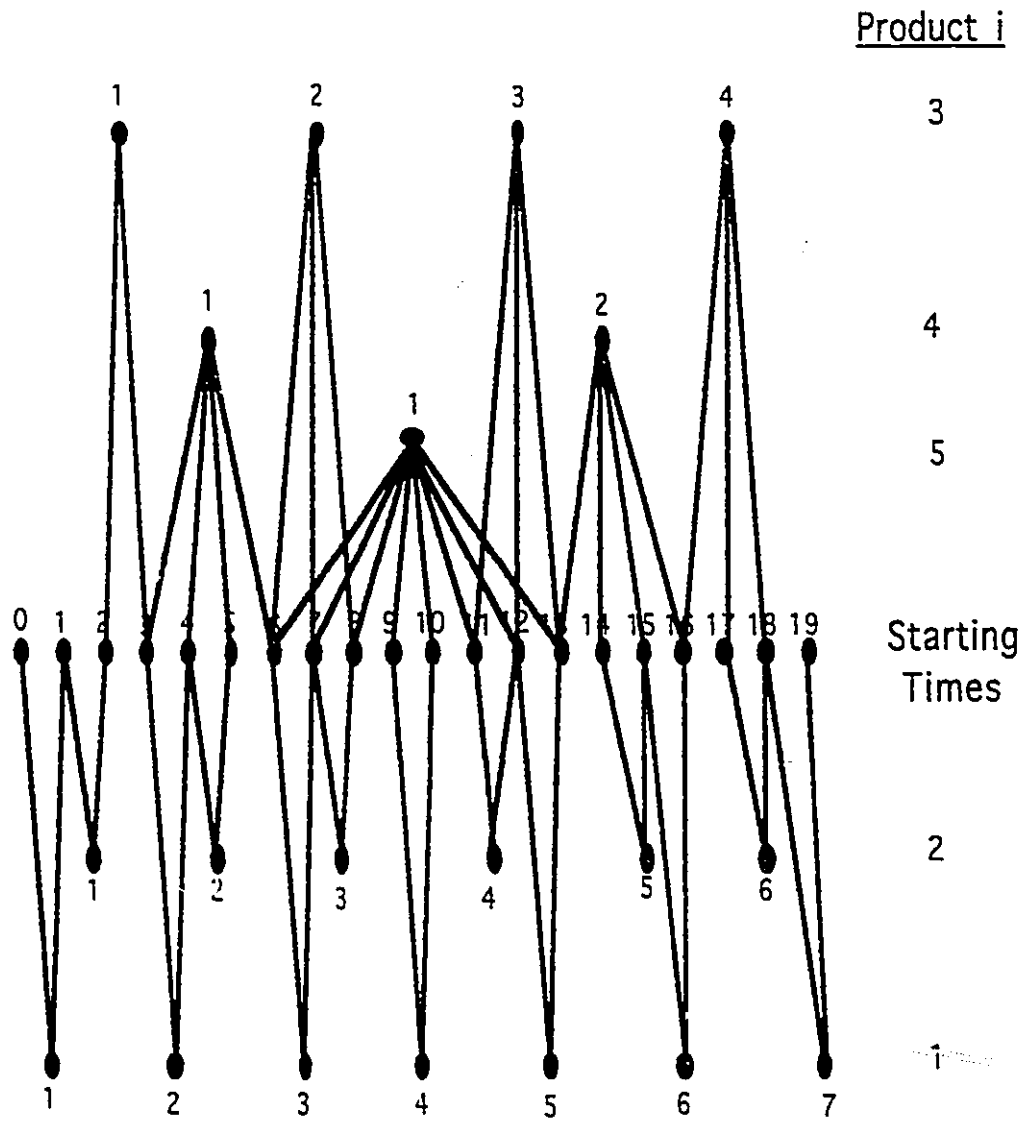
$f^1(k)$ is the deviation for the product i=1. It is made up from the deviations of the individual copies and has a "sawtooth" shape.

IV.4 Graph Theoretic Representation and Due Date Algorithm

The release date/ due date decision problem may be represented as a matching problem in a bipartite graph $G=(V_1,V_2,E)$. Let $V_1=\{0,1,...,D-1\}$ represent the starting times and let V_2 correspond to the copies of each product. Construct an edge between $k\varepsilon V_1$ and $(i,j)\varepsilon V_2$ if (i,j) may start at time k (see fig. IV.3). A bipartite graph is defined to be *V_1-convex* if together $(i,j)\varepsilon E$ and $(k,j)\varepsilon E$ with $i,k\varepsilon V_1$ and $i<k$ implies that $(l,j)\varepsilon E$ for $i\leq l\leq k$ (Derigs et al 1984). Since each $(i,j)\varepsilon V_2$ is incident exactly to the points in $[E(i,j),L(i,j)]\subseteq V_1$, the bipartite graph constructed above is V_1-convex.

A *matching* in a graph is a subset M of edges such that no two edges in M are incident to the same node (Berge 1985). A matching incident to every vertex is a *perfect matching*. Many sufficiency conditions can be stated for the existence of a perfect matching within a graph (Berge; Chartrand & Lesniak 1986). Since the necessary and sufficient conditions for the existence of a perfect matching in a bipartite graph were first presented in Hall (1935), proving that such a matching exists is often referred to as proving *Hall's Theorem* (Bondy & Murty 1976, p72). However, determining whether these conditions actually apply to a particular instance of a graph can prove to be a nontrivial process (Chartrand 1977). Finding a feasible sequence in the release date/ due date decision problem is analogous to finding a perfect matching in the bipartite graph G, with the additional property that lower numbered copies of a product are always matched to earlier starting times than higher numbered copies. Define such a matching to be an *order preserving matching*.

FIGURE IV.3: BIPARTITE GRAPH OF FEASIBLE STARTING TIMES FOR 5 PRODUCTS INDUCED BY A TARGET VALUE OF T=0.65



The demand for each product is; $d_1 = 7, d_2 = 6, d_3 = 4, d_4 = 2, d_5 = 1$

Each copy of a product is labeled by the copy it represents. An edge joins a copy vertex to a time vertex if that copy may feasibly start at that time.

An algorithm will be presented which determines not only whether a perfect matching exists within a graph (i.e. that satisfies Hall's Theorem) but also determines a matching which is order preserving. If $v\varepsilon(V_1 \cup V_2)$, then denote by N(v) the set of vertices which are adjacent to v. Also, if $e\varepsilon E$ is an edge in the graph then let E-e be the edge set obtained by deleting e from E. The algorithm to determine whether a perfect matching exists is as follows;

Algorithm IV.1 (EDD Algorithm)

1. For all $(i,j)\varepsilon V_2$ define $b^i(j) = \max\{t:(t,(i,j))\varepsilon E\}$.
2. Set t=0, M=ø, and $E_{-1}=E$.
3. Select $(t,(i,j))\varepsilon E_{t-1}$ with $b^i(j)$ minimal. If no such edge exists then stop; no perfect matching exists. Else go to step 4.
4. Set $M=M\cup(t,(i,j))$ and $E_t=E_{t-1}-\left[\{(t,k):k\varepsilon N(t)\}\cup\{(k,(i,j)):k\varepsilon N((i,j))\}\right]$.
5. Update $t\leftarrow t+1$. If t=D then stop, a perfect matching, M, has been found. Else, go to step 3.

Various algorithms do exist for determining the maximum matching in a convex, bipartite graph (Lipski & Preparata 1981; Derigs et al 1984; Gallo 1984) and there also exist the analogous earliest due date algorithms (Hodgson & Moore 1968; Simons 1978; Garey et al 1981; Frederickson 1983) used for scheduling unit time jobs with release dates and due dates on a single machine. Algorithm IV.1, which is a modified version of Glover's (1967) $O(|E|)$ algorithm for finding a maximum matching in a V_1-convex bipartite graph $G=(V_1,V_2,E)$, is used because it facilitates the calculation of bounds for the problem. This algorithm can be summarized as follows: For each starting time $k\varepsilon V_1$ (in ascending order) find the previously unmatched products (i,j) with

$(k,(i,j))\varepsilon E$ (i.e. those copies of products which can feasibly start at time k) and match to k the (i,j) with the smallest L(i,j). If no such (i,j) exists, Glover's algorithm would move to vertex k+1, whereas the modified version given above stops, since no perfect matching can possibly exist in G for this case. This modified algorithm will be referred to as *the EDD algorithm.*

The EDD algorithm could stop at a k<D-1 for one of two reasons, discussed below as two separate cases. Define N(i,k,k') to be the number of copies of product i which can be matched to some time $t\varepsilon[k,k']$.

Deficiency Case 1 Too few products for the available time

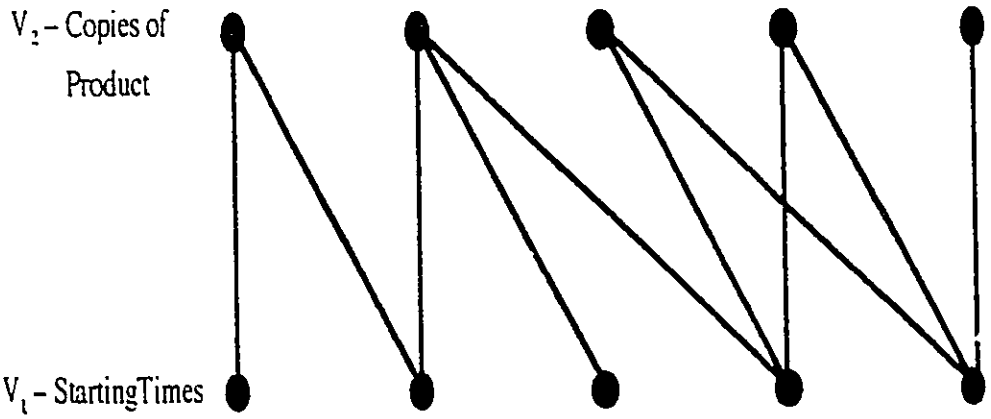
The first reason for stopping is shown in fig. IV.4. This corresponds to the case of having less than p copies available to schedule, in total, in the first p times (i.e $\sum_{i=1}^{n} N(i,0,p-1) < p$).

Deficiency Case 2 Too many products for the available time

In this case, although $\sum_{i=1}^{n} N(i,0,p-1) \geq p$ is satisfied for $1 \leq p \leq D$, the algorithm stops at a k<D-1, because it cannot find a matching product for k (i.e. $\sum_{i=1}^{n} N(i,k,D-1) < D-k$). This would be caused by more than k products having to start at $t\varepsilon[0,k-1]$, as shown in fig. IV.5.

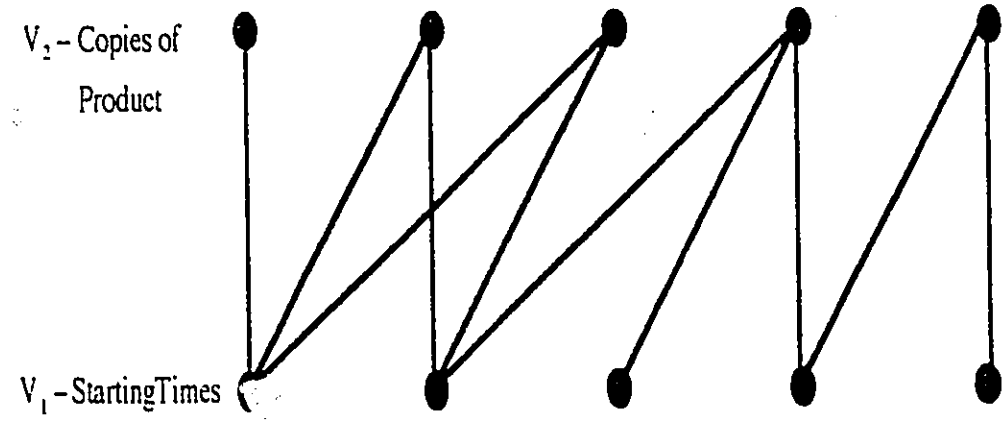
It can be proven that for T=1 neither of these deficiency cases occurs and hence the conditions required to prove Hall's Theorem are satisfied. But first it will be shown that for T values between $1-r_{max} = 1- \underset{i}{Max}\{r_i\}$ and 1 a perfect matching constructed by the EDD algorithm is necessarily an order preserving matching.

FIGURE IV.4: Too Few Copies for Too Many Start Times



In this case, not enough copies are available to start in the first three starting times

FIGURE IV.5: Too Many Copies Required to Start for the Available Time



In this case, three copies must start in the first two starting times

Lemma IV.1:

Only the first copy of a product may start at time t=0 if $1-r_{max} \le T \le 1$.

Proof:

From (4.7), (i,j) may start at time t=0 if $j \le T + r_i$. This implies that $j \le 1 + r_i$ and, because $0 < r_i < 1$, j=1 is the only copy that could start at t=0. □

Lemma IV.2:

The early start times for consecutive copies of each product form a strictly monotonically increasing sequence if $1-r_{max} \le T \le 1$. That is, $E(i,j) < E(i,j+1)$ for $1 \le j \le d_i - 1$ and every product i.

Proof:

From (4.6),

$$E(i,j) < \left(\frac{1}{r_i}\right)[j-T] \qquad \text{and} \qquad E(i,j+1) \ge \left(\frac{1}{r_i}\right)[\, j+1-T\,] - 1 = \left(\frac{j}{r_i}\right) + \left(\frac{1}{r_i}\right) - \left(\frac{T}{r_i}\right) - 1.$$

From this, $E(i,j) < \left(\frac{j}{r_i}\right) + \left(\frac{1}{r_i}\right) - \left(\frac{T}{r_i}\right) - 1 \le E(i,j+1)$ because $0 < r_i < 1$. □

Lemma IV.3:

Only the last copy of a product may start at time t=D-1, with $1-r_{max} \le T \le 1$.

Proof:

From (4.11), (i,j) may start at time t=D-1 if $d_i - r_i - j + 1 \le T$. This implies that $d_i - j \le T - (1-r_i) \le r_i < 1$ and so $j = d_i$ because $j \le d_i$ by definition. □

Lemma IV.4:

The late start times for consecutive copies of each product form a strictly monotonically increasing sequence if $1-r_{max} \le T \le 1$. That is, $L(i,j) < L(i,j+1)$ for $1 \le j \le d_i - 1$ and every product i.

Proof:

From (4.10), $L(i,j) \le \left(\frac{1}{r_i}\right)[\,(j-1)+T\,]$ and $L(i,j+1) > \left(\frac{1}{r_i}\right)[\,j+T\,]-1$.

From this, $L(i,j) < \left(\frac{j}{r_i}\right) + \left(\frac{T}{r_i}\right) - 1 < L(i,j+1)$ because $1 < \left(\frac{1}{r_i}\right)$. □

Theorem IV.1:

For a target value T with $1-r_{max} \le T \le 1$, if the EDD algorithm finds a perfect matching then it is an order preserving matching.

Proof:

By lemmas IV.1 and IV.2, earlier copies of a product become available for scheduling before later copies and the algorithm chooses products in the order of their due dates, which by lemmas IV.3 and IV.4 implies that the order is preserved between different copies of a product. □

IV.5 Bounds on the Target Value

In this section, it will be shown that the target value, T, for the due date decision problem is bounded from above and below.

IV.5.1 Lower Bound

If the first copy of product i is scheduled to be completed at k=1 then $f^i(1) = 1-r_i$. Therefore, a target value T can be feasible only if

$$T \ge \min_i f^i(1) = \min_i \{1-r_i\} = 1 - \max_i r_i = 1-r_{max}$$

Schedules have been found where this bound is achieved (see subsequent example), hence, this lower bound is tight.

IV.5.2 Upper Bound

In this section it will be shown that an upper bound for the target value is T=1. This means, quite suprisingly, that for *any* number, n, of products and *any* set of demand values, $d_1,d_2,...,d_n$, there always exists a schedule in which at any time k (k=1,2,...D) no product is ahead or behind the ideal cumulative production (kr_i for product i) by more than T=1. In order to show this, it will be proved that the EDD algorithm cannot stop at a k<D-1 if T=1.

Lemma IV.5:

With a target value of T=1 the intervals $[E(i,j),L(i,j)]$ and $[E(i,j+1),L(i,j+1)]$ have a non-empty overlap for j=1,2,...,d_i-1and i=1,2,...n.

Proof:

Applying (4.10) with T=1,

$$\left(\frac{j}{r_i}\right)<L(i,j)+1\leq\left(\frac{j}{r_i}\right)+1 . \qquad (4.12)$$

From (4.6),

$$\left(\frac{j}{r_i}\right)-1\leq E(i,j+1)<\left(\frac{j}{r_i}\right) . \qquad (4.13)$$

Combining (4.12) and (4.13) proves E(i,j+1)<L(i,j)+1 and $[E(i,j),L(i,j)]\cap[E(i,j+1),L(i,j+1)]\neq\emptyset$ follows from lemma IV.2. □

Corollary IV.1

The implication of E(i,j+1)<L(i,j)+1 from lemma IV.5 is that, with T=1, consecutive copies of any product will always share at least one starting time

and, when combined with the results of lemmas IV.2 and IV.4, will therefore have at least two feasible starting times.

Lemma IV.6

If $E(i,j)\geq 1$ and the target value $T>0$ is an integer then $E(i,j)r_i$ cannot be an integer value.

Proof:

Assume the contrary to the statement of the lemma. According to (4.4), $j - E(i,j)r_i > T$, but because all values are assumed to be integers, it must be true that, $j - E(i,j)r_i \geq T+1$. Therefore, using $0<r_i<1$, it follows that $j - (E(i,j)+1)r_i \geq T+1-r_i > T$. But this contradicts (4.5). □

Theorem IV.2

An upper bound for the release date/due date decision problem and for the objective function value of the JIT problem is the target value of T=1.

Proof:

(i) First it will be proved that $\sum_{i=1}^{n} N(i,0,k) \geq k+1$ for $k=0,1,\ldots,D-1$.

From (4.4) for $E(i,j)\geq 1$, $j - E(i,j)r_i > 1$, implying

$$j \geq \lceil E(i,j)r_i \rceil + 1 \qquad (4.14)$$

by lemma IV.6 and because j is integer. Here the notation $\lceil x \rceil$ refers to the smallest integer not less than x. From (4.5), $j-(E(i,j)+1)r_i \leq 1$, which by lemma IV.6 implies, $j \leq \lceil E(i,j)r_i \rceil + r_i + 1$, and so

$$j \leq \lceil E(i,j)r_i \rceil + 1. \qquad (4.15)$$

Combining (4.14) and (4.15),

$$j = \lceil E(i,j)r_i \rceil + 1, \text{ if } E(i,j)\geq 1. \qquad (4.16)$$

Note that by lemma IV.1, (4.16) holds also for $E(i,j)=0$.

From (4.8), $L(i,j)r_i-(j-1)\leq 1$, implying

$$\lceil L(i,j)r_i \rceil \leq j. \tag{4.17}$$

From (4.9), $(L(i,j)+1)r_i-(j-1)>1$, implying $\lceil L(i,j)r_i \rceil + r_i > j$, and so

$$\lceil L(i,j)r_i \rceil \geq j. \tag{4.18}$$

By combining (4.17) and (4.18),

$$j=\lceil L(i,j)r_i \rceil \text{ if } L(i,j)\leq D-2. \tag{4.19}$$

The observation that $j=\lceil L(i,j)r_i \rceil$ also holds for $L(i,j)=D-1$ can be made because, in this case, $j=d_i$ by lemma IV.3.

Now let $k\varepsilon\left[E(i,j),L(i,j)\right]$ be any feasible starting time for (i,j). From (4.19), $j=\lceil L(i,j)r_i \rceil \geq \lceil kr_i \rceil$. By the overlapped nature of the starting times stated in lemma IV.5, any starting time k will always be an element of a time interval for some copy of product i. If j is the largest integer number for which $k\varepsilon\left[E(i,j),L(i,j)\right]$ then, in total, j copies of product i have been available to be scheduled in [0,k] and $N(i,0,k)=j=\lceil L(i,j)r_i \rceil$, using (4.19). Summing over all products i,

$$\sum_{i=1}^{n} N(i,0,k) = \sum_{i=1}^{n} \lceil L(i,j)r_i \rceil \geq \sum_{i=1}^{n} \lceil E(i,j+1)r_i \rceil \geq \sum_{i=1}^{n} E(i,j+1)r_i > \sum_{i=1}^{n} kr_i = k,$$

where the first inequality follows by lemma IV.5, the second one from the definition of $\lceil x \rceil$ and the last one from $k<E(i,j+1)$, which is true by the definition of j. Thus,

$$\sum_{i=1}^{n} N(i,0,k) \geq k+1, \text{ for all } k. \tag{4.20}$$

(ii) Next it will be shown that $\sum_{i=1}^{n} N(i,k,D-1) \geq D-k$.

In order to do this, define a new variable p(i) to be the p^{th} last copy of product i. That is, if p(i) is the same as (i,j) from the front then ,

$$p=d_i-j+1 \qquad p=1,2,\ldots,d_i. \tag{4.21}$$

Let $K_1 \geq 1$ be the first time in the algorithm at which there are only p copies of product i remaining unscheduled. If p and j are related as in (4.21) then $K_1=L(i,j-1)+1$ and if $K_2=L(i,j)$, then for each starting time $k\varepsilon[K_1,K_2]$ there are exactly p copies of product i not yet scheduled. For any such k,

$$N(i,k,D-1) \geq p. \tag{4.22}$$

Applying (4.10) to L(i,j-1) with T=1,

$$\left(\frac{1}{r_i}\right)[j-1] < K_1=L(i,j-1)+1 \leq \left(\frac{1}{r_i}\right)[j-1]+1.$$

Substituting (4.21) into this, $d_i-p \leq \lfloor K_1 r_i \rfloor \leq d_i-p+r_i$, where $\lfloor x \rfloor$ is the largest integer not greater than x. Since d_i and p are both integers and $0<r_i<1$, this implies that,

$$p=d_i-\lfloor K_1 r_i \rfloor = d_i - \lfloor (L(i,j-1)+1)r_i \rfloor. \tag{4.23}$$

Substituting (4.23) into (4.22) and summing over all products i;

$$\sum_{i=1}^{n} N(i,k,D-1) = \sum_{i=1}^{n} \left[d_i - \lfloor (L(i,j-1)+1)r_i \rfloor \right] \geq \sum_{i=1}^{n} d_i - \sum_{i=1}^{n} kr_i = D-k.$$

Fact (i) implies that deficiency case 1 can never occur in the application of the EDD algorithm for T=1; and (ii) implies that deficiency case 2 cannot occur either. Thus, the algorithm will not stop before k=D-1and will, therefore, determine a perfect matching. □

Hence the JIT problem always possesses an order preserving, feasible schedule in which, at any given time, the actual production deviates from the ideal level by at most 1 unit for every one of the n products.

Example: Consider a 5 product problem where the respective demands are $d_1=7$, $d_2=6$, $d_3=4$, $d_4=2$, $d_5=1$. The total demand is D=20, hence, the product ratios are $r_1=\frac{7}{20}$, $r_2=\frac{6}{20}$, $r_3=\frac{4}{20}$, $r_4=\frac{2}{20}$, $r_5=\frac{1}{20}$. Figure IV.3 shows the early and late starting times and the bipartite graph induced by the

target value, $T = .65 = 1 - r_1 = \frac{7}{20} = 1 - r_{max}$ for product 1. The EDD algorithm generates the order preserving product sequence, 1-2-3-1-2-4-1-2-3-1-5-2-1-3-2-1-4-2-3-1; implying that a copy of product 1 is produced, then a copy of product 2, then a copy of product 3 and so on. As this sequence is feasible and the target value is set at the problem's lower bound, this solution is also optimal.

IV.6 Optimal Solutions

An easy implementation of the EDD algorithm gives an $O(nD)$ time solution for the decision problem : "Is there a feasible schedule with $Z \leq T$?". Using the approach of Frederickson (1983), the algorithm's complexity can be reduced to $O(D)$ time. It was shown above that for T=1, the answer to the decision problem is always "yes" and for any feasible schedule, $Z \geq 1 - r_{max}$. By performing a bisection search on the interval $[(1 - r_{max}), 1]$, these bounds can be tightened to an interval $[LB, UB] \subset [1 - r_{max}, 1]$ for which $LB \leq Z^* \leq UB$ and $UB - LB \leq \varepsilon$, where Z^* is the optimal objective function value for the problem and $\varepsilon > 0$ is an arbitrary fixed tolerance. To tighten the bounds to LB and UB will require solving $O\left(\log\left(\frac{1}{\varepsilon}\right)\right)$ decision problems, which means that the overall complexity of the procedure is $O\left(D \log\left(\frac{1}{\varepsilon}\right)\right)$. This solution, however, will only be ε-Optimal.

It can be shown that the bisection search algorithm determines the optimal solution in time which is polynomial in D. Instead of requiring the bisection search to proceed until the difference between UB and LB is the tolerance value ε, the procedure can be modified to stop when the convex, bipartite graphs corresponding to the release date-due date decision problems created by the values of UB and LB contain exactly the same edge sets. When

this situation occurs, only one target value could possibly be contained in the interval $[UB,LB]$ and, as in this instance LB and UB must both be feasible, the optimal target value T^* is LB. Hence, the bisection search algorithm with this modified stopping rule would determine the optimal solution.

Theorem IV.3

The bisection search algorithm with the modified stopping rule runs in $O(D \log D)$ time.

Proof

At a time k, for some copy j of product i it must be the case that,

$$T^* = | j - kr_i |.$$

But then,

$$DT^* = D| j - kr_i | = | Dj - kd_i |$$

and thus DT^* is an integer. It has been shown that $1-r_{max} \leq T^* \leq 1$, so DT^* must be some integer in the interval $[D-d_{max} , D]$. A bisection search of the integers in this interval can be performed. Finding the optimal value in this range would require solving $O(\log d_{max})$ decision problems, which means that the overall complexity of the procedure is $O(D \log d_{max}) \leq O(D \log D)$. Hence, optimization using this algorithm would require $O(D \log D)$ time. □

Clearly, not an excessively large number of iterations of the procedure will be performed and it can be seen that for practical purposes, the entire procedure is computationally efficient.

IV.7 Chapter Summary

The JIT scheduling of mixed-model assembly processes involves minimizing the deviation of production from demand. There are certain

penalties involved with the holding of inventories and the occurrences of shortages. This chapter has shown that it is possible to calculate, within $O(D \log D)$ time, an optimal solution to this problem. It has also been shown that a schedule always exists for the JIT problem such that at no time does the actual production deviate from the ideal level of production for any product by more than 1 unit.

As a corollary to the work of this chapter, a parallel has been observed to exist between scheduling unit time tasks on a single processor with release times and due dates and determing the maximum cardinality matching in convex, bipartite graphs. The most computationally efficient algorithm for the convex bipartite graph matching problem appearing in the literature is the "almost linear time" $O(D+D*A(D))$ algorithm of Lipski & Preparata (1981); where A(D) is a very slowly growing function related to a functional inverse of the Ackerman funtion [for a description of the Ackerman function see Tarjan (1983, p24-29)]. Using the same data structures employed by Frederickson (1983) for the scheduling problem (see Appendix 1), the efficiency of the maximum matching problem in convex, bipartite graphs can be improved to $O(D)$ (see Appendix 2).

CHAPTER 5 COMPUTATIONAL EFFICIENCIES FOR SINGLE LEVEL PROBLEMS

V.1 Introduction

The previous chapter provided an optimal, $O(D \log D)$ time algorithm for solving the unweighted, single level problem formulation. In this chapter, ways to significantly reduce the computational effort required of this algorithm will be examined and it will be shown how the algorithm can be modified to solve instances of the weighted, single level problem.

V.2 Relationships Between Copies of a Product

Certain inherent relationships exist within the problem framework which allow for considerable reductions in the amount of computational effort. Intervals (4.6) and (4.10) have been used to calculate the early and late start times, $E(i,j) \geq 1$ and $L(i,j) \leq D-1$, for any target value, T. The special cases, (4.7) and (4.11), were introduced to determine which copies of a product could start at times 0 and D-1 respectively, as in a feasible sequence, no copy could start before time 0 or after time D-1. These special cases were necessary in the proof of the bounds and to show the feasibility of the EDD algorithm. However, (4.6) and (4.10) can be used to derive all of the early and late starting times if it is implicitly recognized that a value of $E(i,j)<0$ implies that the earliest feasible start time corresponding to the copy (i,j) is time 0 and that a value of $L(i,j)>D-1$ implies that the latest feasible start time corresponding to this (i,j) is time D-1. In this chapter, (4.6) and (4.10) will be used to determine E(i,j) and L(i,j), for all

$i=1,...,n$ and $j=1,...,d_i$, as the actual values of both $E(i,j)$ and $L(i,j)$ found within these intervals is of considerable importance in the relationships between the various copies.

V.2.1 Relationships Involving Early and Late Starting Times

A correspondence can be shown to exist between the early start time of one copy and the late start time of some other, related copy. If (i,j) is the j^{th} lowest numbered copy of product i, then the corresponding j^{th} highest numbered copy for this product is (i,d_i-j+1).

Lemma V.1

The sum of $E(i,j)$ and $L(i,d_i-j+1)$ is the constant D-1 for every i and j.

Proof:

Restating (4.6) as (5.1), the earliest starting time for (i,j) is the unique integer in the interval,

$$\left(\frac{1}{r_i}\right)\left[\, j \;-\; T \,\right] - 1 \le E(i,j) < \left(\frac{1}{r_i}\right)\left[\, j \;-\; T \,\right], \qquad (5.1)$$

and, from (4.10), the late starting time for the $(d_i-j+1)^{th}$ copy is the unique integer in the interval,

$$\left(\frac{1}{r_i}\right)\left[\, (d_i-j+1) \;-\; 1+T \,\right] - 1 < L(i,d_i-j+1) \le \left(\frac{1}{r_i}\right)\left[\, (d_i-j+1) \;-\; 1+T \,\right].$$

This implies that,

$$D - \left(\frac{1}{r_i}\right)\left[\, j \;-\; T \,\right] - 1 < L(i,d_i-j+1) \le D - \left(\frac{1}{r_i}\right)\left[\, j \;-\; T \,\right]$$

and thus,

$$\left(\frac{1}{r_i}\right)\left[\, j \;-\; T \,\right] > D - L(i,d_i-j+1) - 1 \ge \left(\frac{1}{r_i}\right)\left[\, j \;-\; T \,\right] - 1. \qquad (5.2)$$

Recognizing that $E(i,j)$, D and $L(i,d_i-j+1)$ are all integers and comparing (5.1)

with (5.2), while recognizing that the intervals contain a single integer, implies that,

$$E(i,j) = D - L(i,d_i-j+1) - 1$$

or

$$E(i,j) + L(i,d_i-j+1) = D - 1. \quad \square \tag{5.3}$$

Equation (5.3) demonstrates that if the value of T changes causing E(i,j) to start earlier/later, then this change in T will simultaneously result in causing $L(i,d_i-j+1)$ to start later/earlier.

Remark V.1

By lemma V.1, it is therefore only necessary to calculate, for each product, either the early start times for all of its copies or the late start times for all of its copies (but not both). Hence, the number of starting time calculations for an instance of the release date-due date decision problem described in chapter 4 can be halved. □

An example of (5.3) can be demonstrated for the special cases of early and late starting times mentioned above. Lemma IV.1 stated that only the first copy of a product could start at time 0 if $0 \le T \le 1$. From (4.7), the first copy of a product could start at time 0 only if $1-r_i \le T$. It can be seen that for $1-r_i \le T < 1$, $E(i,1)=0$ will fall into the interval (5.1), but if T=1, then (5.1) implies that $E(i,1)=-1$, for all i=1,...,n.

Restating (4.10) as (5.4), the late starting time for any copy j of product i is,

$$\left(\frac{1}{r_i}\right)\left[\,(j-1)+T\,\right] - 1 < L(i,j) \le \left(\frac{1}{r_i}\right)\left[\,(j-1)+T\,\right]. \tag{5.4}$$

Lemma IV.3 stated that only the last copy of a product could start at time D-1 if

$0 \le T \le 1$. Using this lemma and setting $j=d_i$ in (4.11), the last copy of a product can start at time D-1 only if $1-r_i \le T$. Using (5.4), if $1-r_i \le T < 1$, $L(i,d_i)=D-1$ falls into the interval, but if $T=1$, then $L(i,d_i)=D$, for all $i=1,\ldots,n$. Hence, using (5.1) and (5.4) with $0 \le T \le 1$, $E(i,1)+L(i,d_i)=D-1$ for all $i=1,\ldots,n$, which is the relationship of (5.3).

When $T=1$ (for which feasible schedules always exist), the cases $E(i,1)=-1$ and $L(i,d_i)=D$ are encountered for each i. These starting times could never occur in a feasible schedule. The EDD algorithm requires that copies can only start from time 0 onwards and thus $E(i,1)=-1$ would simply imply that the copy would be available to start at time 0. Likewise, $L(i,d_i)=D$ would never be the late starting time for any copy in a feasible sequence but would imply that the copy for which this value held would be the last copy considered for scheduling by the algorithm. However, the values of $E(i,1)=-1$ and $L(i,d_i)=D$ are of interest, because if one value is known, then (5.3) allows for the calculation of the other value without having to determine what integer falls into either the interval (5.1) or (5.4). By similar calculations, (5.3) can be shown to hold for all other copies (i,j), $i=1,\ldots,n$, $j=2,\ldots,d_i-1$.

V.2.2 Cyclical Relationships Between Copies of a Product

Two integers α_i and β_i always exist such that $\alpha_i\beta_i=d_i$ (i.e. α_i and β_i are factors of d_i). It is therefore possible, using these integers, to label the copies of product i as;

$$\{1,2,3,\ldots,\alpha_i,\alpha_i+1,\alpha_i+2,\ldots,2\alpha_i,2\alpha_i+1,\ldots,3\alpha_i,\ldots,[(\beta_i-1)\alpha_i+\alpha_i-1],[(\beta_i-1)\alpha_i+\alpha_i]\}.$$

Let each copy of a product be numbered as,

$$(c-1)\alpha_i+j$$

where $c=1,2,\ldots,\beta_i$ and $j=1,2,\ldots,\alpha_i$. Then for each fixed value of c (say $c=a+1$),

there will be a group of α_i copies of product in the range $[a\alpha_i+1, a\alpha_i+\alpha_i]$. This range will be referred to as the $(a+1)^{st}$ tier of copies for product i. There will be β_i such tiers (or ranges) for each product since there are β_i values to which c could be set.

Lemma V.2

If β_i is also a factor of D, then the early start time for the j^{th} copy in any tier is the sum of the early start time of the j^{th} copy in the first tier plus an integer constant which is calculable *a priori* .

Proof:

Applying (5.1) for copy $(q-1)\alpha_i+j$ of product i, where $q\varepsilon\{1,2,...,\beta_i\}$ and $j\varepsilon\{1,2,...,\alpha_i\}$ provides the early starting time for this copy as,

$$\left(\frac{1}{r_i}\right)\left[\ (q-1)\alpha_i+j\ -\ T\ \right]-1 \le E(i,(q-1)\alpha_i+j) < \left(\frac{1}{r_i}\right)\left[\ (q-1)\alpha_i+j\ -\ T\ \right]$$

implying $$\left(\frac{1}{r_i}\right)\left[\ j\ -\ T\ \right]-1 \le E(i,(q-1)\alpha_i+j) - \left(\frac{1}{r_i}\right)\left[\ (q-1)\alpha_i\right] < \left(\frac{1}{r_i}\right)\left[\ j\ -\ T\ \right].$$

Now $d_i=\alpha_i\beta_i$, $r_i=\frac{d_i}{D}$ and $\frac{1}{r_i}=\frac{D}{d_i}=\frac{D}{\alpha_i\beta_i}$ so,

$$\left(\frac{1}{r_i}\right)\left[(q-1)\alpha_i\right]=\frac{D}{\alpha_i\beta_i}\left[(q-1)\alpha_i\right]=\frac{D}{\beta_i}\left[(q-1)\right].$$

Hence,

$$\left(\frac{1}{r_i}\right)\left[\ j\ -\ T\ \right]-1 \le E(i,(q-1)\alpha_i+j) - \frac{D}{\beta_i}\left[(q-1)\right] < \left(\frac{1}{r_i}\right)\left[\ j\ -\ T\ \right]. \quad (5.5)$$

But if β_i is also a factor of D, then $\frac{D}{\beta_i}\left[(q-1)\right]$ will be an integer and its value will be calculable, *a priori* , for each q. If this is the case, then by (5.5) and (5.1),

$$E(i,j) = E(i,(q-1)\alpha_i+j) - \frac{D}{\beta_i}\left[(q-1)\right]$$

or

$$E(i,(q-1)\alpha_i+j) = E(i,j) + \frac{D}{\beta_i}\left[(q-1)\right] \quad (5.6)$$

for $1 \le j \le \alpha_i$ and $1 \le q \le \beta_i$. Hence if β_i is a factor of D and E(i,j) [the early start time of the j^{th} copy in the first tier] is known, then $E(i,(q-1)\alpha_i+j)$ may be calculated as the sum of E(i,j) and a predetermined constant. □

An analogous relationship exists for the late starting times of related copies in different tiers.

Lemma V.3

If β_i is also a factor of D, then the late start time for the j^{th} copy in any tier is the sum of the late start time of the j^{th} copy in the first tier plus an integer constant which is calculable *a priori*.

Proof:

Applying (5.4) for copy $(q-1)\alpha_i+j$ of product i, where $q \varepsilon \{1,2,\ldots,\beta_i\}$ and $j \varepsilon \{1,2,\ldots,\alpha_i\}$ provides the late starting time for this copy as,

$$\left(\frac{1}{r_i}\right)\left[((q-1)\alpha_i+j-1)+T\right] - 1 < L(i,(q-1)\alpha_i+j) \le \left(\frac{1}{r_i}\right)\left[((q-1)\alpha_i+j-1)+T\right]$$

implying $$\left(\frac{1}{r_i}\right)\left[(j-1)+T\right] - 1 < L(i,(q-1)\alpha_i+j) - \left(\frac{1}{r_i}\right)\left[(q-1)\alpha_i\right] \le \left(\frac{1}{r_i}\right)\left[(j-1)+T\right],$$

or $$\left(\frac{1}{r_i}\right)\left[(j-1)+T\right] - 1 < L(i,(q-1)\alpha_i+j) - \frac{D}{\beta_i}\left[(q-1)\right] \le \left(\frac{1}{r_i}\right)\left[(j-1)+T\right]. \quad (5.7)$$

If β_i is a factor of D, then $\frac{D}{\beta_i}\left[(q-1)\right]$ is an integer constant which is calculable *a priori* and by (5.4) and (5.7),

$$L(i,j) = L(i,(q-1)\alpha_i+j) - \frac{D}{\beta_i}\left[(q-1)\right]$$

or $$L(i,(q-1)\alpha_i+j) = L(i,j) + \frac{D}{\beta_i}\left[(q-1)\right] \quad (5.8)$$

for $1 \le j \le \alpha_i$ and $1 \le q \le \beta_i$. Hence if β_i is a factor of D and L(i,j) [the late start time of

the j^{th} copy in the first tier] is known, then $L(i,(q-1)\alpha_i+j)$ may be calculated as the sum of $L(i,j)$ and a predetermined constant. □

Remark V.2

Together, lemmas V.2 and V.3 imply that if D is a multiple of β_i then only the early and late starting times for copies $j=1,2,...,\alpha_i$ in the first tier need be calculated as the start times for all copies in the remaining tiers are a linear function of those in the first tier. This is because $\frac{D}{\beta_i}[(q-1)]$ will be integer, hence the values inside the inequalities (5.5) and (5.7) must be integer, and the inequalities contain a unique integer. In order to minimize the number of required starting time calculations, β_i should be the largest factor of D possible (although it is possible that $\beta_i=1$ could be the largest such factor and that the computational reductions resulting from lemma V.1 are maximal). □

V.2.2 Relationships Between Starts Within Each Tier

In lemma V.1 a relationship between the early starting time of the j^{th} lowest numbered copy and the late starting time of the j^{th} highest numbered copy was shown to exist. It can be shown that a similar such relationship exists between similarly related copies within each tier.

Lemma V.4

Within a tier, the sum of the early start time for the j^{th} copy and the late start time of the $(\alpha_i-j+1)^{st}$ copy is an integer constant which is calculable *a priori*.

Proof:

Integers α_i and β_i always exist such that $d_i=\alpha_i\beta_i$. Of these integers, let β_i be the largest such factor of D (i.e. a factor $\beta_i \geq 1$ always exists). Hence, an integer, H, can be found such that $H=\frac{D}{\beta_i}$. Thus, $r_i=\frac{d_i}{D}=\frac{\alpha_i\beta_i}{D}$ implies that $\frac{1}{r_i}=\frac{D}{\alpha_i\beta_i}=\frac{H\beta_i}{\alpha_i\beta_i}=\frac{H}{\alpha_i}$.

The early starting time for the j^{th} copy, $j=1,2,...,\alpha_i$, within the c^{th} tier, $c=1,2,...,\beta_i$, is, from (5.1),

$$\left(\frac{1}{r_i}\right)\left[(c-1)\alpha_i+j-T\right] - 1 \leq E(i,(c-1)\alpha_i+j) < \left(\frac{1}{r_i}\right)\left[(c-1)\alpha_i+j-T\right]. \qquad (5.9)$$

From (5.4), the late starting time for the $(\alpha_i-j+1)^{st}$ copy in the c^{th} tier is,

$$\left(\frac{1}{r_i}\right)\left[(c-1)\alpha_i+(\alpha_i-j+1)-1+T\right]-1 < L(i,(c-1)\alpha_i+(\alpha_i-j+1)) \leq \left(\frac{1}{r_i}\right)\left[(c-1)\alpha_i+(\alpha_i-j+1)-1+T\right].$$

This implies that,

$$H[c-1] + H - \left(\frac{1}{r_i}\right)\left[j-T\right] - 1 < L(i,(c-1)\alpha_i+(\alpha_i-j+1)) \leq H[c-1] + H - \left(\frac{1}{r_i}\right)\left[j-T\right],$$

or that,

$$\left(\frac{1}{r_i}\right)\left[j-T\right] > Hc - 1 - L(i,(c-1)\alpha_i+(\alpha_i-j+1)) \geq \left(\frac{1}{r_i}\right)\left[j-T\right] - 1.$$

Recognizing that $\left(\frac{1}{r_i}\right)\left[(c-1)\alpha_i\right] = H[c-1]$, then

$$\left(\frac{1}{r_i}\right)\left[(c-1)\alpha_i+j-T\right] > H[2c-1] - 1 - L(i,(c-1)\alpha_i+(\alpha_i-j+1)) \geq \left(\frac{1}{r_i}\right)\left[(c-1)\alpha_i+j-T\right] - 1. \quad (5.10)$$

But because H, c, $L(i,(c-1)\alpha_i+(\alpha_i-j+1))$ and $E(i,(c-1)\alpha_i+j)$ are all integers. expressions (5.9) and (5.10) imply that,

$$E(i,(c-1)\alpha_i+j) = H[2c-1] - 1 - L(i,(c-1)\alpha_i+(\alpha_i-j+1))$$

or

$$E(i,(c-1)\alpha_i+j) + L(i,(c-1)\alpha_i+(\alpha_i-j+1)) = H[2c-1] - 1. \qquad (5.11)$$

Now H and c are integers and will be known prior to the calculation of the

starting times. Hence, the value H[2c-1]-1 must be an integer constant and can be determined for each value of c. □

Remark V.3

The implication of lemma V.4 is that within a given tier, for each product, only the calculations for either all of its copies early starts or all of its copies late starts need to be performed. Expression (5.11) indicates that if the early starting time of copy $(c-1)\alpha_i+j$ changes, then so does the late starting time of copy $(c-1)\alpha_i+(\alpha_i-j+1)$. Hence, the sum of the early and late starting times of the j^{th}copy and the $(\alpha_i-j+1)^{st}$ copy within any tier c is always constant. Combining this result with that of remark V.2, implies that once all of one tier's early (or all of its late) starting times have been calculated, then the early and late starting times for all copies in all tiers can be calculated directly from these values, resulting in a considerable reduction in the amount of computation required for an instance of the release date-due date problem. □

V.2.3 Factoring Properties of the Model

By lemma IV.2, lemma IV.4 and theorem IV.2 feasible schedules created by the EDD algorithm with $T\leq 1$ are necessarily order preserving. In this section, these properties will be exploited to show that certain factoring properties exist for the single level problem.

Let β be the greatest common divisor of each d_i and set $\beta_i=\beta$ for all i. Let **D** be the product requirements vector for the products, thus

$$\mathbf{D} = [d_1,d_2,\ldots,d_n] = [\alpha_1\beta,\alpha_2\beta,\ldots,\alpha_n\beta].$$

Letting $A = \sum_{i=1}^{n} \alpha_i$ be the total number of copies of all products from any given tier, then $D = \sum_{i=1}^{n} d_i = \sum_{i=1}^{n} \alpha_i\beta = \beta \sum_{i=1}^{n} \alpha_i = \beta A$ and $r_i = \frac{d_i}{D} = \frac{\alpha_i\beta}{\beta A} = \frac{\alpha_i}{A}$.

Lemma V.5

If $\beta_p=\beta$ for all products p, p=1,...,n, then, with T=1, the starting times for the last copy of any product in the c^{th} tier, c=1,2,...,β_p, [i.e. copy $(c-1)\alpha_p+\alpha_p$] are such that:

$$E(p,(c-1)\alpha_p+\alpha_p) < A[c]-1 \quad \text{and} \quad L(p,(c-1)\alpha_p+\alpha_p) = A[c].$$

Proof:

From (5.4), with T=1,

$$\left(\frac{1}{r_p}\right)\left[(c-1)\alpha_p+\alpha_p\right] - 1 < L(p,(c-1)\alpha_p+\alpha_p) \le \left(\frac{1}{r_p}\right)\left[(c-1)\alpha_p+\alpha_p\right].$$

Now, $\left(\frac{1}{r_p}\right)\left[(c-1)\alpha_p+\alpha_p\right] = \left(\frac{A}{\alpha_p}\right)\alpha_p\,[c] = A[c]$, and both A and c are integer, hence, it must be the case that,

$$L(p,(c-1)\alpha_p+\alpha_p) = A[c]\,. \tag{5.12}$$

From (5.1), the early start time for this copy is,

$$\left(\frac{1}{r_p}\right)\left[(c-1)\alpha_p+\alpha_p-1\right] - 1 \le E(p,(c-1)\alpha_p+\alpha_p) < \left(\frac{1}{r_p}\right)\left[(c-1)\alpha_p+\alpha_p-1\right].$$

But,

$$\left(\frac{1}{r_p}\right)\left[(c-1)\alpha_p+\alpha_p-1\right] = \left(\frac{1}{r_p}\right)\left[(c-1)\alpha_p+\alpha_p\right] - \left(\frac{1}{r_p}\right) = A[c] - \left(\frac{1}{r_p}\right) < A[c]-1$$

because $\left(\frac{1}{r_p}\right) > 1$. Thus,

$$E(p,(c-1)\alpha_p+\alpha_p) < A[c]-1. \quad \square \tag{5.13}$$

Lemma V.6

If $\beta_p=\beta$ for all products p, p=1,...,n, then, with T<1, the late starting time for the last copy of any product in the c^{th} tier, c=1,2,...,β_p, [i.e. copy $(c-1)\alpha_p+\alpha_p$] is such that:

$$L(p,(c-1)\alpha_p+\alpha_p) \le A[c]-1.$$

Proof:

By (5.4), the late starting time must satisfy,

$$\left(\frac{1}{r_p}\right)\left[(c-1)\alpha_p+\alpha_p-1+T\right] - 1 < L(p,(c-1)\alpha_p+\alpha_p) \le \left(\frac{1}{r_p}\right)\left[(c-1)\alpha_p+\alpha_p-1+T\right].$$

From the proof of lemma V.5, $\left(\frac{1}{r_p}\right)\left[(c-1)\alpha_p+\alpha_p\right]=A[c]$, which is integer and by definition $\frac{1}{r_p}>1$ and $\delta=T-1<0$, thus $\varepsilon=\delta\frac{1}{r_p}<0$. Hence,

$$L(p,(c-1)\alpha_p+\alpha_p) \le A[c]+\varepsilon < A[c]$$

Now because $L(p,(c-1)\alpha_p+\alpha_p)$ is integer, it must be the case that,

$$L(p,(c-1)\alpha_p+\alpha_p) \le A[c]-1. \quad \square \qquad (5.14)$$

Lemma V.7

If $\beta_q=\beta$ for all products q, q=1,...,n, then, with T=1, the starting times for the first copy of any product in the e^{th} tier, where e=c+1, c=1,2,...,β_q-1, [i.e. copy $(e-1)\alpha_q+1$] are such that:

$$E(q,((e-1)\alpha_q+1) = A[c]-1 \quad \text{and} \quad L(q,((e-1)\alpha_q+1) > A[c].$$

Proof:

From (5.4) with T=1, the late starting time must satisfy,

$$\left(\frac{1}{r_q}\right)\left[(e-1)\alpha_q+1\right] - 1 < L(q,((e-1)\alpha_q+1) \le \left(\frac{1}{r_q}\right)\left[(e-1)\alpha_q+1\right].$$

Now $\left(\frac{1}{r_q}\right)\left[(e-1)\alpha_q\right] = \left(\frac{A}{\alpha_q}\right)\alpha_q\,[c+1-1] = A[c]$ and by definition $\gamma = \left(\frac{1}{r_q}\right)-1>0$, hence,

$$A[c] < A[c]+\gamma < L(q,((e-1)\alpha_q+1). \qquad (5.15)$$

From (5.1), the early start time must satisfy,

$$\left(\frac{1}{r_q}\right)\left[(e-1)\alpha_q\right] -1 \le E(q,((e-1)\alpha_q+1) < \left(\frac{1}{r_q}\right)\left[(e-1)\alpha_q\right]$$

and thus, $$E(q,((e-1)\alpha_q+1) = A[c]-1. \quad \square \qquad (5.16)$$

Lemma V.8

If $\beta_q=\beta$ for all products q, q=1,...,n, then, with T<1, the early starting time for the first copy of any product in the e^{th} tier, where e=c+1, c=1,2,...,β_q-1, [i.e. copy $(e-1)\alpha_q+1$] is such that:

$$E(q,((e-1)\alpha_q+1) \ge A[c].$$

Proof:

From (5.1), for a target value T<1, the early start time must satisfy,

$$\left(\frac{1}{r_q}\right)\left[(e-1)\alpha_q+1-T\right] -1 \le E(q,((e-1)\alpha_q+1) < \left(\frac{1}{r_q}\right)\left[(e-1)\alpha_q+1-T\right]$$

but because $\frac{1}{r_q}>1$, 1-T>0 and $\Omega=\left(\frac{1}{r_q}\right)\left[1-T\right]>0$, then

$$A[c]-1 < A[c]+\Omega-1 \le E(q,((e-1)\alpha_q+1)$$

and thus, $$A[c] \le E(q,((e-1)\alpha_q+1). \quad \square \qquad (5.17)$$

Theorem V.1

If T≤1 and $\beta=\beta_i$, i=1,...,n, is the greatest common divisor of each d_i, then an optimal schedule exists which consists of β repetitions of the optimal sequence to the subproblem where $\mathbf{D}=\left[\alpha_1,\alpha_2,...,\alpha_n\right]$.

Proof:

Theorem IV.2 demonstrated that for T=1 a feasible schedule always exists. If feasible schedules exist, then the EDD algorithm may be used to find

one such feasible schedule and, hence, by lemmas IV.2 and IV.4, an order preserving sequence may always be determined for a feasible $T \leq 1$.

If $T=1$, then by lemmas V.5 and V.7, for consecutive tiers c and e=c+1, c=1,2,...,β-1, (5.13) and (5.16) show that

$$E(p,(c-1)\alpha_p+\alpha_p) < A[c]-1 = E(q,((e-1)\alpha_q+1)$$

and (5.12) and (5.15) show that

$$L(p,(c-1)\alpha_p+\alpha_p) = A[c] < L(q,((e-1)\alpha_q+1)$$

for p=1,...,n and q=1,...,n. Thus, if the EDD algorithm is employed, a copy of any product in one tier will never be sequenced ahead of a copy of any product in a preceding tier. For tier 1, one copy must be scheduled to start at time 0 and, as all A copies of all products within tier 1 are scheduled before any copies in tier 2, using the assumptions of the EDD algorithm and the fact that the sequence is feasible, the last copy in the first tier to be sequenced must begin processing at time A-1 (completing at time A). Furthermore, the first tier 2 copy must start at time A and all other tier 2 copies must be sequenced before those in tier 3. As there are A tier 2 copies, the last copy of this tier must begin processing at time 2A-1 (completing at time 2A). Proceeding inductively, all copies in one tier (tier c) must be sequenced ahead of those copies in the subsequent tier (tier e) by the EDD algorithm, and the last copy in each tier c will start processing at time cA-1 (completing at time cA).

From lemmas V.6 and V.8, if $T<1$, (5.14) and (5.17) show that

$$L(p,(c-1)\alpha_p+\alpha_p) \leq A[c]-1 < A[c] \leq E(q,((e-1)\alpha_q+1)$$

so that for <u>any</u> feasible schedule created in any manner whatever (i.e. not necessarily using the EDD algorithm) all copies in one tier must be sequenced ahead of any copies in the subsequent tier.

In lemmas V.2 and V.3, it was shown that for any product i, the early and late starting times for the j^{th} copy, $(j=1,2,\ldots,\alpha_i)$, in any tier are related by the equations

$$E(i,(q-1)\alpha_i+j) = E(i,j) + \frac{D}{\beta_i}[(q-1)]$$

and

$$L(i,(q-1)\alpha_i+j) = L(i,j) + \frac{D}{\beta_i}[(q-1)] \qquad q=1,2,\ldots,\beta_i.$$

But if $\beta_i=\beta$, $i=1,\ldots,n$, then $\frac{D}{\beta_i}=\frac{A\beta}{\beta}=A$, and therefore the early and late starting times for the j^{th} copy of product i in one tier are simply the early and late starting times of the preceding tier plus the constant A; which is the total number of copies within each tier. Coupling this fact with the knowledge that all copies in one tier are sequenced prior to the copies of the subsequent tier, it can be noted that the copies of each product become available for scheduling in exactly the same order that they became available for scheduling in the preceding tier. Therefore, with $T\leq 1$, the EDD algorithm will repeat exactly the same sequence of production within each tier.

Thus, for any $T\leq 1$, if a feasible sequence has been determined for the first tier, then a feasible sequence for the entire problem will consist of β repetitions of this sequence. Furthermore, if an optimal sequence has been determined for the first tier, then an optimal sequence has also been determined for the entire problem. □

Remark V.4

As a result of theorem V.1, if the d_i's possess a common divisor β, only the sub-problem where this β has been factored out of the d_i's need be considered (that is, consider the new problem where demands are given by $\frac{d_i}{\beta}$, $i=1,\ldots,n$). This is a far stronger assertion than that of the "cyclical solutions"

described in section III.5 as it means that, *a priori*, a substantial portion of the computation that may have been required to determine the optimal solution need never be undertaken. For example, if the demand vector for a problem is given as $\mathbf{D} = [3000,2000,1000]$, then the optimal solution will consist of 1000 repetitions of the optimal solution to the significantly smaller problem with demand vector $\mathbf{D} = [3,2,1]$. □

Henceforth, the assumption can be made that any problem examined has had the common divisor, β, factored out. It should be noted that even after this factoring has occurred, the computational reductions described by remark V.3 still hold for the newly created, reduced problem.

V.3 Determining the Target Values which create new Release Date Due Date Decision Problems

In section IV.5 an optimization algorithm was presented for the single level problem. It was shown that by searching within the upper and lower bounds only a finite number of convex, bipartite graphs corresponding to different values of the target would be created. In this section, it will be shown that the values of the target at which these new graphs are created can be pre-determined.

Lemma V.9

An increase in the target value from T to $T'=T+r_i$ will cause the early and late starting times of all of the copies of product i to change.

Proof:

Consider copy $u \le d_i$ of product i and assume that for T, u is such that E(i,u) has just become its early starting time. That is, in (5.1), E(i,u) achieves equality with the left endpoint of the interval. Hence,

$$E(i,u) = \left(\frac{1}{r_i}\right)\left[\, u \; - \; T \,\right] - 1. \tag{5.18}$$

If T increases, then E(i,u) will remain as the early starting time for (i,u) until it achieves equality with the right endpoint of the interval (5.1), at which time the early starting time will shift. This occurs at a certain T' for which

$$E(i,u) = \left(\frac{1}{r_i}\right)\left[\, u \; - \; T' \,\right]. \tag{5.19}$$

Subtracting (5.18) from (5.19) results in,

$$0 = 1 + \left(\frac{1}{r_i}\right)\left[\, T' \; - \; T \,\right]$$

implying that $T' = T + r_i$.

By the choice of u, this must be the copy which requires the largest increase in the target value in order for its early start time to change. As, by (5.1), the early starting times for all other copies would shift for smaller increases to T, this increase to T' would cause the early start times of all other copies to change also. By (5.3), as all of the early start times change, all of the late starting times must also change. Hence, increasing the target value by r_i would add at least $2d_i$ edges to the convex, bipartite graph of the release date-due date decision problem. □

Lemma V.10

The values of T which change the early and late starting times for every copy of each product may be determined prior to the commencement of the optimization algorithm.

Proof:

By remark V.3, in order to determine all of a product's early and late starting times, only the late starting times for the copies $j=(1,2,\ldots,\alpha_i)$ within the first tier need to be calculated. For a product i, the total number of calculations is minimized if β_i is the largest factor of D. Hence, fix i and set β_i to be the largest such factor.

Setting T=0 then, by (5.4), L(i,1)=0 and L(i,1) is such that it is at the equality extreme of the range given by (5.4). From lemma V.9, the next time that L(i,1) will change occurs when $T=r_i$. Late starting times can also be derived from (5.4) for all of the other copies j. [Note that for T=0 the starting times will not be feasible as $L(i,j)<E(i,j)$, $j=1,\ldots,d_i$. However, feasibility is not the issue examined here; the concern is merely with determining the values of T which cause the starting times to shift].

Calculate for each j,

$$\delta_j = \left\lceil \frac{j-1}{r_i} \right\rceil - \frac{j-1}{r_i}$$

and order the δ_j's in increasing order. The next change in starting times for each j will occur at $T=\delta_j r_i$ and, by lemma V.9, at $T=\delta_j r_i+mr_i$ (m=1,2,...) in the future. By definition of δ_j and r_i, $\delta_j < 1$, $j=1,\ldots,d_i$ and thus, $\delta_j r_i < r_i$. Hence if T is set initially to 0 and is then increased, all copies within the first tier will shift starting times before copies that have already shifted once must shift again. Therefore, the target value which causes the initial shift for each j can be set at $\delta_j r_i$ and at

increments of r_i for ever after. This can be done for the first tier of each product. Hence, the values of T which change the early and late start times for every copy of each product could be determined before the algorithm starts. □

By lemma V.10, the target values which add edges to the convex bipartite graph, thereby creating new release date-due date decision problems, can be predetermined. It would be possible to optimize the single level problem by starting initially at T=0 and incrementing T to the next (predetermined) larger value that would create a new release date-due date problem in such a fashion until the first feasible schedule was found. By the way in which T was initially set, this first feasible solution would also be optimal. However, it is far more efficient to perform a bisection search within the calculated bounds and to stop when only one feasible target value remains within these bounds.

V.4 Extensions to Weighted Single-Level Problems

In chapter 3, the minimax objective function for the single-level problem was presented in the form,

$$\text{Minimize} \quad \max_{i,k} \; G_{i1} \left| x_{i1k} - XT_{1k} r_{i1} \right| \qquad i=1,\ldots,n_1,\ k=1,\ldots,D_1$$

where G_{i1} is the weighting factor for product i. If $G_{i1} \neq 1$, for at least one i, then this objective represents that of a weighted single-level usage problem. Dropping the superfluous subscript 1 and substituting $XT_{1k}=k$, level curves can be constructed for each copy j of every product i in an analogous fashion to the unweighted case (see fig. V.1). The effect of the weighting factors is to shift the slopes of these level curves. Once again, a target value for the value of the

FIGURE V.1 Calculation of Early and Late Starting Times of a Product for an Instance of the Weighted, Usage Problem

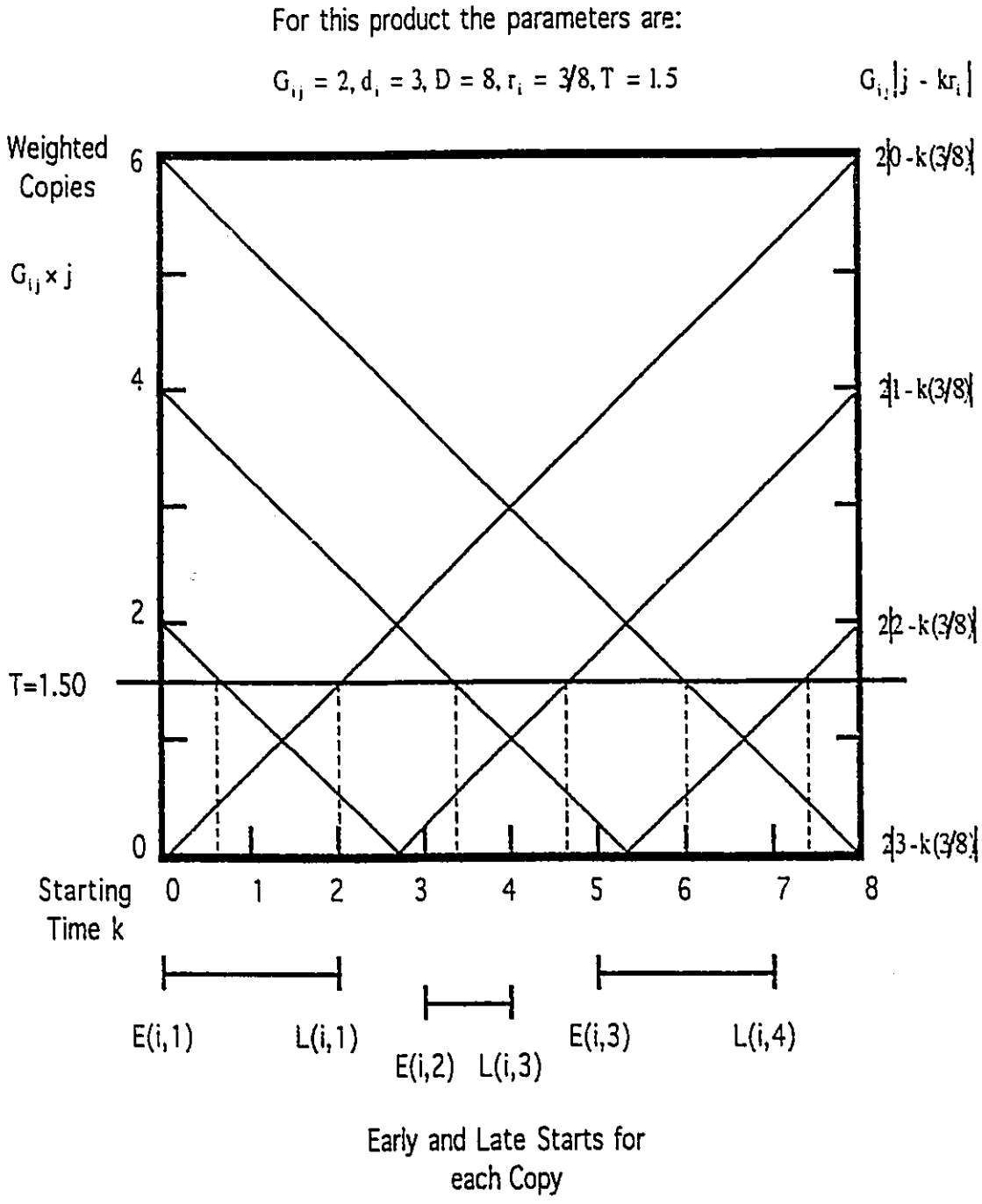
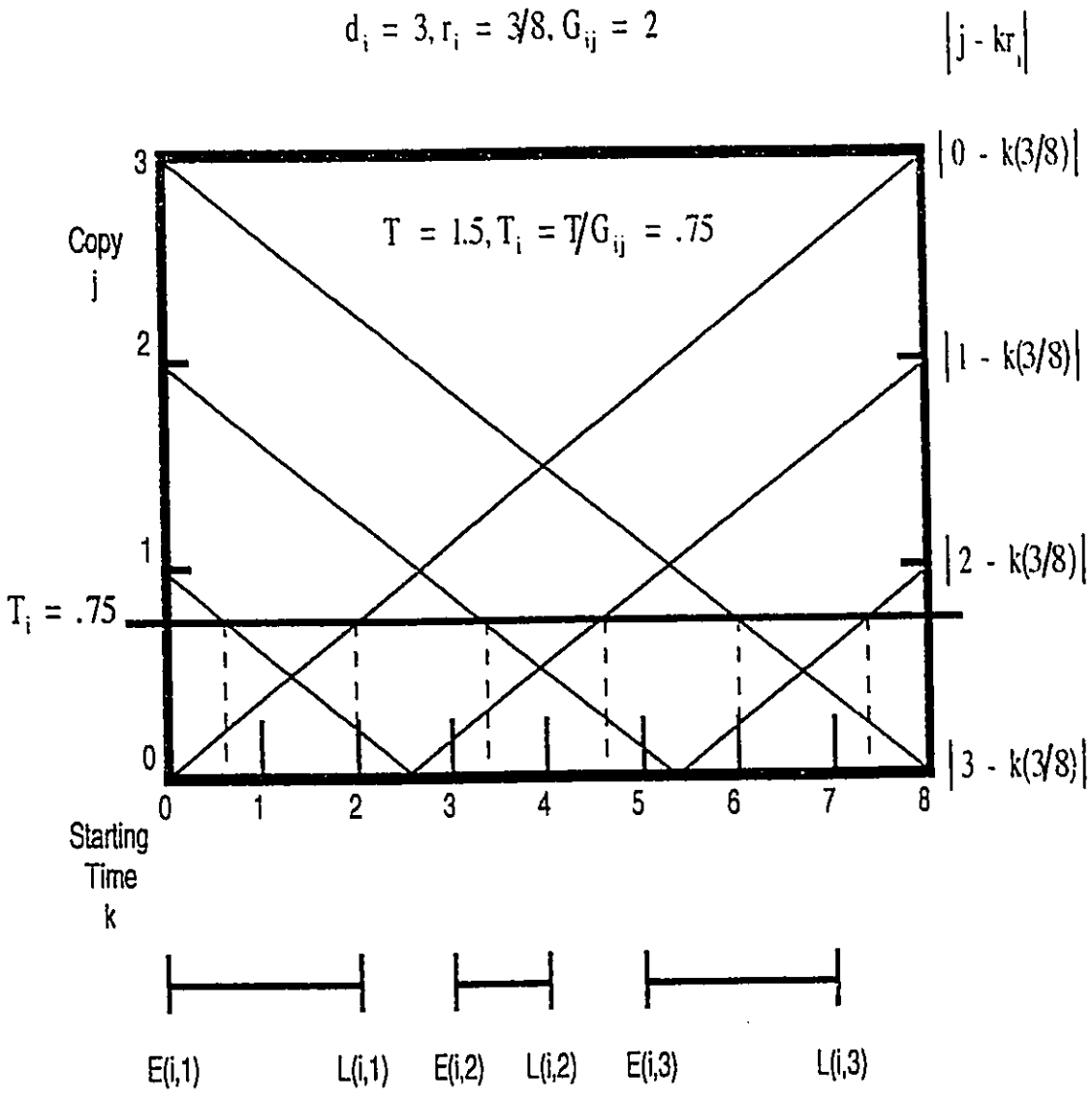


FIGURE V.2: Early and Late Starts for a Weighted Problem Calculated from an Unweighted Figure with Product Specific Target Line



The Early and Late Starting for Unweighted Problem with Product Specific Target Line are Identical to the Starting Times for the Weighted Problem.

objective function can be hypothesized, but now, a sequence must be determined such that,

$$G_i \, |j - kr_i| \leq T, \qquad i=1,...,n,\ j=1,...,d_i$$

in order for T to be a feasible target.

For a given T, increasing the slope of one product's level curves relative to the slopes of the remaining products reduces the ranges of feasible starting locations in the final production sequence for each copy of that product. Hence, weighting a product more heavily restricts where copies of that product can be scheduled and also increases the separation of consecutive copies of that product within the assembly sequence.

Define $T_i = \frac{T}{G_i}$ to be the target line for the i^{th} product induced by T.

Thus, an instance of the weighted problem could also be thought of as an instance of the unweighted problem in which each product i has its own target line T_i. Replacing T by T_i in (5.1) and (5.4) implies that early and late starts can be calculated for every copy of each product for a fixed T (see fig. V.2).

Corollary V.1

An upper bound for the objective value of the weighted, single level problem is G_{max}.

Proof:

If $T = G_{max} = \underset{i}{\text{Max}}\{G_i\}$, then $T_i \geq 1$ for all i. This would be equivalent to an unweighted problem instance in which $T \geq 1$, and, by theorem IV.2, a feasible sequence could be determined using the EDD algorithm. Therefore, $T = G_{max}$ is an upper bound for the target value of the weighted problem. □

Analogously to the unweighted problem, product i cannot start at time 0 unless $T \geq G_i(1-r_i)$, thus a lower bound for the problem is $LBW = \underset{i}{\text{Min}}\{G_i(1-r_i)\}$.

If a bisection search for the feasibility of the target values is performed in the interval $[\ LBW,\ G_{max}\ ]$, then an implementation of the EDD algorithm described in section IV.5 could determine an ε-optimal solution for the weighted problem. For the decision problem resulting from each T , $T_i = \frac{T}{G_i}$ is calculated for every product and the starting times induced by the target value for each copy are determined. As T_i is a function of G_i, the proof of the factoring theorem V.1 is not applicable to the weighted problem. However, using T_i, lemmas V.2, V.3 and V.4 still hold and remark V.3 can be used for each i, as it can be easily checked that their proofs carry through even for different T_i values. Hence, calculation of all of the early (or all of the late) starting times for only one tier need be performed in order to determine all of the copy's starting times. For each product (starting from T=0 and increasing), the values of T (and therefore of T_i) for which additional early and late starting times occur can be determined. These shifts could occur up to $T_i = \frac{G_{max}}{G_i}$; implying that there will be more shifts occurring than in the unweighted problem. The important fact is that only a finite number of these shifts can occur. If the bisection search stops when the edge set of the bipartite, convex graphs induced by LB and UB does not change, then $T^* = LB$ will be the optimal solution and could be determined in a finite number of steps.

Theorem V.2

The bisection search algorithm finds the optimal solution for the weighted, single-level usage problem in $O\left(D \log D\Phi G_{max}\right)$ time, where Φ is a positive integer constant which can be easily computed from the problem data.

Proof

Applying lemma V.10 with T_i, at a time k, for some copy j of product p it must be the case that,

$$T^* = G_p \lfloor j - kr_p \rfloor,$$

for at least one product p. Under the assumption that the G_i's are elements of the set of rational numbers, then an integer Φ must exist such that ΦG_i holds integer value for all products i. But then,

$$D\Phi T^* = D\Phi G_p \lfloor j - kr_p \rfloor = \Phi G_p \lfloor Dj - kd_p \rfloor$$

and $D\Phi T^*$ must be integer. From above $T^* \leq G_{max}$, so $D\Phi T^*$ must be an integer in the interval $[\, D\Phi LBW , D\Phi G_{max} \,]$. A bisection search of this interval could be performed to determine the optimal target value and, hence, optimization using the algorithm would require $O\left(D \log D\Phi G_{max}\right)$ time. □

Theorem V.2 indicates that the weighted, single-level usage problem could be solved to optimum in time which is polynomial in D and in the size of the weighting factors, if these weighting factors are restricted to being rational numbers.

V.5 Summary

This chapter has demonstrated that weighted instances of the single level problems may be solved to optimum using an algorithm that is polynomial in the total demand. This also implies that the extension to the multilevel problem under the assumption of Goldstein & Miltenburg (1988), shown to be equivalent to the weighted single-level problem in section III.3.3, can also be solved to optimum in polynomial time. A strong, *a priori* factoring property that allows for a significant reduction in the amount of computation, often assumed

to exist as a cyclical property of JIT problems, has been proved to exist for optimal solutions to the unweighted, single-level problem. Other relationships which allow for computational reductions have also been shown to exist.

CHAPTER 6 MULTI-LEVEL OPTIMIZATION

VI.1 Introduction

In this chapter, the general minimax problem will be transformed into a more convenient matrix notation. Using this representation, a dynamic programming (DP) procedure will be developed for its optimization. This DP procedure can be modified to solve cases of the general sum function problem too; hence, an optimization process can also be constructed for solving problems with multi-level sum functions. The time and space requirements of the minimax DP will be derived and certain methods for improving its performance will be discussed.

VI.2 Matrix Representation

Because of the many subscripts and variables of the multi-level problem, a more compact notation is desirable. It is possible to present the general minimax problem in a more concise matrix representation. By definition, at a stage k, the deviation of the i^{th} part at level j would be $G_{ij}|x_{ijk} - XT_{jk}r_{ij}|$. It has been shown (Miltenburg & Sinnamon 1989) that,

$$\begin{aligned} x_{ijk} - XT_{jk}r_{ij} &= \sum_{l=1}^{n_1} t_{ijl}x_{l1k} - r_{ij}\sum_{l=1}^{n_1}\sum_{h=1}^{n_j} t_{hjl}x_{l1k} \\ &= \sum_{l=1}^{n_1}\left(t_{ijl} - r_{ij}\sum_{h=1}^{n_j} t_{hjl}\right)x_{l1k} \\ &= \sum_{l=1}^{n_1}\delta_{ijl}x_{l1k}, \qquad \text{where } \delta_{ijl}=t_{ijl}- r_{ij}\sum_{h=1}^{n_j} t_{hjl}\,. \end{aligned}$$

Then because $G_{ij} \geq 0$, $x_{ijk} \geq 0$ and $r_{ij} > 0$, it is possible to write the deviations for the minimax problem as,

$$\begin{aligned} G_{ij}\,|x_{ijk} - XT_{jk}r_{ij}| &= |G_{ij}(x_{ijk} - XT_{jk}r_{ij})| \\ &= |G_{ij}(\sum_{l=1}^{n_1} \delta_{ijl}x_{l1k})| \\ &= |(\sum_{l=1}^{n_1} \gamma_{ijl}x_{l1k})|, \qquad \text{where } \gamma_{ijl}=G_{ij}\delta_{ijl}\,. \end{aligned}$$

The inherent pull nature of the JIT process is very much apparent within this notation. At any particular stage, the deviation of any part at any level depends explicitly on the production sequence of the product level. Note that in the last step of the calculation above, the deviation for any part i at any level j is dependent solely upon calculable, and hence known, constants (the γ_{ijl}'s) multiplied by the cumulative production at level 1 (the x_{l1k}'s). These γ_{ijl}'s are a measure of the weighted deviations in usage of part i in level j from the proportional usage per unit of product l.

The γ_{ijl}'s can be placed into a matrix Γ (the "gamma matrix") of dimension $n \times n_1$, where $n = \sum_{j=1}^{L} n_j$ is the total number of different parts and products (see Miltenburg & Sinnamon for the gamma matrix of the sum functions). Each row of Γ corresponds to one of the parts or products at each of the various production levels. The rows of Γ start with product 1 of level 1in row 1 and continue with product 2 of level 1 in row 2,..., product n_1 of level 1 in row n_1, part 1 of level 2 in row n_1+1,..., part n_L of level L in row $\sum_{j=1}^{L} n_j$. The value γ_{ijl} will be the element appearing in the $\left(\sum_{m=1}^{j-i} n_m + i\right)^{th}$ row and the l^{th} column of

the matrix Γ. For example $\gamma_{ijl}=\gamma_{324}$ will be the entry appearing in row n_1+3, column 4 of Γ.

If $\mathbf{A}=[a_1,a_2,\ldots,a_n]$ is a vector with n rows, then define $\|\mathbf{A}\|_1=\max_{1\le i\le n}\{|a_i|\}$ to be the maximum norm of **A**. Define $\|\mathbf{A}\|_2=\sqrt{\sum_{i=1}^{n}(a_i)^2}$ to be the Euclidean norm of **A**. Define $\mathbf{X}_k = (x_{11k},x_{21k},\ldots,x_{n_1 1k})$ to be the cumulative, level 1 production vector through the first k stages. Let $\omega(\mathbf{X}_k)$ denote the maximum deviation at cycle k over all parts and products (i.e. over all i and j). Then,

$$\omega(\mathbf{X}_k) = \max_{i\ j}\{G_{ij}\,|x_{ijk} - XT_{jk}r_{ij}|\} = \|\Gamma\,\mathbf{X}_k\|_1 \ .$$

Hence, the objective function in the general minimax problem could be reformulated as,

$$\text{Minimize } \max_{ijk}\{G_{ij}\,|x_{ijk} - XT_{jk}r_{ij}|\} = \min\ \max_{k}\omega(\mathbf{X}_k) = \min\ \max_{k}\|\Gamma\,\mathbf{X}_k\|_1 \ .$$

VI.3 Optimization using Dynamic Programming

Using this matrix representation, a dynamic programming (DP) procedure can be constructed which optimizes the general, minimax problem. Employing the gamma matrix notation, this procedure builds upon the single-level sum function algorithm of Miltenburg, Steiner & Yeomans (1990), providing an optimization method for solving any of the minimax problem formulations.

VI.3.1 DP Algorithm for the General Minimax Problem

Let $\mathbf{d} = (d_{11},d_{21},\ldots,d_{n_1 1}) = (d_1,d_2,\ldots,d_{n_1})$ be the level 1, product requirements vector. For simplicity, redefine subsets in a schedule by $\mathbf{X} = (x_1,x_2,\ldots,x_{n_1})$, where x_i is a non-negative integer representing the production of exactly x_i units of product i, $x_i \le d_i$. Let e_i be the usual i^{th} unit vector (i.e. with n_1

entries, all of which are zero except for a single 1 in the i^{th} row). A subset **X** could be scheduled within the first k stages only if,

$$k = \sum_{i=1}^{n_1} x_i \ .$$

Define $\phi(\mathbf{X})$ to be the minimum value of the maximum deviation for all parts and products over all k stages for any schedule where the products in **X** are produced during these first k stages. That is, $\phi(\mathbf{X})$ is the best path of production, in the minimax sense, from time 0 to time k when the products in **X** are produced. Then, by definition, $\| \Gamma\mathbf{X} \|_1$ is the maximum deviation of actual production from desired production for all parts and products when **X** is the amount of product produced. The following DP recursion holds for $\phi(\mathbf{X})$:

$$\phi(\varnothing) = \phi(\mathbf{X} : \mathbf{X}=\mathbf{0}) = 0$$

$$\phi(\mathbf{X}) = \phi(x_1,x_2,\ldots,x_{n_1}) = \min_i \left\{ \max\left\{ \phi(\mathbf{X}-\mathbf{e}_i),\ \| \Gamma\mathbf{X} \|_1 \right\} : i=1,2,\ldots,n_1,\ x_i-1\geq 0 \right\}$$

It can be seen that $\phi(\mathbf{X})\geq 0$, for any set **X**, and it follows from the definition of the r_{ij}'s that,

$$\| \Gamma\mathbf{d} \|_1 = \omega(\mathbf{X} : \mathbf{X}=\mathbf{d}) = 0$$

where **d** is the product requirements vector. The DP proceeds in the 'forward direction' by examining sets in increasing order of their cardinality (i.e. by moving from a set **X** of a given cardinality to the higher cardinality successors (or children) of this set). The steps involved in the solution of the DP are presented below in algorithm VI.1. Some of these steps are elaborated upon in greater detail in the subsequent sections of this chapter.

Algorithm VI.1: DP Algorithm

1. Read the product demand vector, **d**, and sort the elements within this vector into non-decreasing order of their magnitude. Read the part requirements for each product.
2. Calculate, for each product i, the units u(i) for the encoding (or addressing) scheme and place these units into the vector **U**. This scheme must allow for the calculation of a unique encoding (or address) for each subset **X**. Check that **Ud** (the inner product of **U** and **d**) does not exceed the largest available integer allowed by the programming language. If it does, then the encoding will have to be broken up into several segments (dimensions) [see section VI.5.3]. [Note: It is assumed that only one address dimension is required in this description of the DP algorithm. Additional dimensions could be easily accomodated. An addressing scheme is necessary as a set **X** could be generated in several ways (i.e. from more than one predecessor set) and only the best solution needs to be stored. The addressing system allows sets to be efficiently stored and quickly referenced to determine if they have been generated previously by some other predecessor set.]
3. Construct the matrix Γ.
4. Calculate a screening value, SCREEN, for the DP (see section VI.5.2). SCREEN corresponds to an upper bound on the objective value. Any sets, **X**, examined which have a maximum deviation greater than SCREEN will be fathomed.
5. Initialize the first stage so that **X**←**0**. Initialize the matrices **OLD**←**0** and **NEW**←**0**. **OLD** and **NEW** each contain 3 columns.

6. Generate all of the feasible sets $\mathbf{X}^{+i} = \mathbf{X} + e_i$ (where feasibility implies that $x_i + 1 \leq d_i$) which could be successors to **X**. If $\| \Gamma\mathbf{X}^{+i} \|_1 \leq$ SCREEN then encode the set $\mathbf{X}^{+i}$ as $\mathbf{UX}^{+i}$ and place $\mathbf{UX}^{+i}$, i, and $\| \Gamma\mathbf{X}^{+i} \|_1$ into the matrix **OLD** which is sorted in the following way;

   If $\mathbf{UX}^{+i}$ is the q^{th} largest address value of the sets to be stored at a stage then,

   $\mathbf{OLD}(q,1)=\mathbf{UX}^{+i}$, $\mathbf{OLD}(q,2)=i$, $\mathbf{OLD}(q,3)= \| \Gamma\mathbf{X}^{+i} \|_1$.

   That is, **OLD** is sorted by increasing value of the set address **OLD**(_ ,1). **OLD**(_ ,2) corresponds to the last product produced on the best schedule for producing the set $\mathbf{X}^{+i}$. And **OLD**(_ ,3) is the minimum value of the maximum deviation which would be encountered by any schedule used for producing the products in the set $\mathbf{X}^{+i}$. Let PREV be the number of sets stored in **OLD** at this stage.

7. Initialize k←1 and COUNTER←0.

8. Decode **OLD**(k,1) to determine the set **X**.

9. For each i, i=1,2,...,n_1, determine if $\mathbf{X}^{+i} = \mathbf{X} + e_i$ is a feasible successor set to **X**. If no feasible successor sets exist then go to step 11. For each feasible $\mathbf{X}^{+i}$, if $\| \Gamma\mathbf{X}^{+i} \|_1 \leq$ SCREEN then calculate $\mathbf{UX}^{+i}$ and perform a binary search of the address values in **NEW**(p,1), p=1,2,...,COUNTER, to determine if $\mathbf{X}^{+i}$ has been generated previously.

   If $\mathbf{X}^{+i}$ has not been previously generated, then COUNTER← COUNTER+1 and the information for the set $\mathbf{X}^{+i}$ is added to **NEW** in such a way that the matrix remains sorted in increasing order of the values in **NEW**(_ ,1). That is, if $\mathbf{UX}^{+i}$ becomes the p^{th} largest set address in **NEW**, p≤ COUNTER, then,

NEW(p,1)=UX^{+i} , **NEW**(p,2)=i, **NEW**(p,3)=max $\left\{ \mathbf{OLD}(k,3),\ \| \Gamma X^{+i} \|_1 \right\}$.

As with **OLD**, **NEW** is sorted by increasing value of the set address **NEW**(_ ,1) and **NEW**(_ ,2) corresponds to the last product produced on the best schedule for producing the set X^{+i}. **NEW**(_ ,3) is the minimum value of the maximum deviation which would be encountered by any schedule used for producing the products in the set X^{+i}. This deviation is the maximum of either the deviation corresponding to the set X^{+i} or the deviation corresponding to the best schedule to produce the products in the predecessor set **X**.

If X^{+i} has been previously generated, then it must be the case that UX^{+i} = **NEW**(m,1) for some m≤COUNTER. If

$$\max\left\{ \| \Gamma X^{+i} \|_1,\ \mathbf{OLD}(k,3) \right\} \geq \mathbf{NEW}(m,3)$$

then leave **NEW** unchanged. If, however,

$$\max\left\{ \| \Gamma X^{+i} \|_1,\ \mathbf{OLD}(k,3) \right\} < \mathbf{NEW}(m,3)$$

then a better schedule to the set X^{+i} has been found. Hence, update **NEW** so that,

$$\mathbf{NEW}(m,2) = i \quad \text{and} \quad \mathbf{NEW}(m,3) = \max\left\{ \| \Gamma X^{+i} \|_1,\ \mathbf{OLD}(k,3) \right\}.$$

10. Increment k←k+1. If k≤ PREV then return to step 8. Otherwise, write **OLD** to a file in secondary storage. This file will possess two columns; one column will be for the values of the address and the other column will be for the last product to be produced on the best path to the set corresponding to this address. Rename **NEW** to **OLD**. Set PREV← COUNTER, **NEW**←0 and return to step 7.

11. The optimal minimax value for the problem is **OLD**(1,3).

12. The production sequence corresponding to this optimal minimax value may be constructed by examining the information contained in the file on secondary storage (see section VI.5.4).

Thus, a DP algorithm exists for optimally solving the general, minimax problem formulation, but would be used only for solving the multi-level problem instances for which no specific, polynomial time procedures have been found.

VI.3.2 Optimization of Multi-Level Sum Functions

It was noted in chapter 2 that no method had been presented in the literature for optimally solving multi-level sum functions. It is possible to construct a DP approach for solving the sum functions, similar to the method used for the minimax formulation. Recall that the objective function for the general multi-level sum function was,

$$\sum_{k=1}^{D_1} \sum_{j=1}^{L} \sum_{i=1}^{n_j} G_{ij} \left(x_{ijk} - XT_{jk} r_{ij}\right)^2 .$$

Letting $\gamma_{ijl} = \left(G_{ij}\right)^{\frac{1}{2}} (\delta_{ijl})$, where δ_{ijl} is defined as above, and Γ being the matrix of the γ_{ijl}, then for the given part i at level j in stage k,

$$\left(G_{ij}\right)^{\frac{1}{2}} \left(x_{ijk} - XT_{jk} r_{ij}\right) = \sum_{l=1}^{n_1} \gamma_{ijl} x_{11k} .$$

Miltenburg & Sinnamon (1989) demonstrated that the variability at stage k for a set **X** with $|\,\mathbf{X}\,| = k$ could be written as,

$$\sum_{j=1}^{L} \sum_{i=1}^{n_j} G_{ij} \left(x_{ijk} - XT_{jk} r_{ij}\right)^2 = \left(\| \Gamma \mathbf{X} \|_2\right)^2 ,$$

and that, therefore,

$$\text{Min} \sum_{k=1}^{D_1} \sum_{j=1}^{L} \sum_{i=1}^{n_j} G_{ij} \left(x_{ijk} - XT_{jk}r_{ij}\right)^2 = \text{Min} \sum_{|X|=1}^{D_1} \left(\| \Gamma X \|_2 \right)^2.$$

Define $\varphi(\mathbf{X})$ to be the minimum total deviation for all parts and products over all k stages for any schedule in which the products in $\mathbf{X}$ are produced during these first k stages. Let,

$$\theta(\mathbf{X}) = \left(\| \Gamma X \|_2 \right)^2$$

be the squared sum of the deviations of actual production from desired production for all parts and products when $\mathbf{X}$ is the amount of product produced. The following DP recursion can be used for determining the value of $\varphi(\mathbf{X})$:

$$\varphi(\mathbf{0}) = \varphi(\mathbf{X} : \mathbf{X}=\mathbf{0}) = 0$$
$$\varphi(\mathbf{X}) = \text{Min} \left\{ \varphi(\mathbf{X}-e_i)+\theta(\mathbf{X}) : i=1,2,\ldots,n_1 \, , \, x_i-1 \geq 0 \right\}$$

where it is the case that $\varphi(\mathbf{X}) \geq 0$, for any set $\mathbf{X}$, and $\theta(\mathbf{X} : \mathbf{X}=\mathbf{D}) = 0$. Hence, a DP algorithm can also be constructed for optimally solving any formulation of the sum function problem, but more specifically for use in the optimization of multi-level problems.

VI.4 Time and Space Requirements of the DP

In this section, the time and space requirements of the DP will be shown for the minimax case (however, the results are extendable to the sum function DP).

In a given stage k for the set $\mathbf{X}$, the value of $\| \Gamma X \|_1$ provides the maximum deviation of the cumulative production of each part/product from the relative amount of the total production at the level which should be of this part/product. The minimization in the recursion is performed over all possible

choices of the level 1 product which could be in the last position (i.e. the last product to be produced in **X**). As x_i could hold any of the values $0,1,...,d_i$, the total number of sets or states in the DP recursion is,

$$\prod_{i=1}^{n_1} (d_i+1) \quad .$$

The address of **X**, value $\phi(\mathbf{X})$ and the product i, where the minimum occurs in the recursion, must be stored for each set **X**, so that the optimal schedule can be constructed at the conclusion of the algorithm. Therefore, the space requirements are

$$O\left(\prod_{i=1}^{n_1} (d_i+1)\right).$$

For any set **X**, there could be at most n_1 values $\phi(\mathbf{X}-e_i)$; each of which would have to be compared to $\| \Gamma\mathbf{X} \|_1$. The computation time for $\| \Gamma\mathbf{X} \|_1$ is $O(n_1 n)$. Therefore, the computation time for the entire problem is,

$$O\left(n_1 n \prod_{i=1}^{n_1} (d_i+1)\right).$$

For any problem instance, the number of feasible schedules is,

$$\frac{D_1!}{d_1!\, d_2!...d_{n_1}!}$$

which is considerably larger than the number of states in the DP recursion since,

$$\prod_{i=1}^{n_1} (d_i+1) \leq \left[\frac{d_1+d_2+...+d_{n_1}+n_1}{n_1}\right]^{n_1} = \left[\frac{D_1+n_1}{n_1}\right]^{n_1} ,$$

where the inequality follows since the product of n_1 numbers is not greater than the mean of those numbers multiplied by itself n_1 times.

For example, if $\mathbf{d}=[9,10,10,11,12,13]$ then the number of feasible sequences would be 1.4497×10^{46} which is significantly larger than the 2,642,640 states which would be created in the DP algorithm. Hence, the implicit enumeration process of the DP substantially reduces the computational effort which would be required by any procedure employing explicit enumeration.

It can be observed that the number of sets grows exponentially with n_1, but that this growth rate is polynomial in D_1. This implies that the DP would be effective for those cases in which n_1 is small, even when the total product demand, D_1, is large .

VI.5 Implementation of the DP

While the DP is an efficient implicit enumeration algorthm, it is still an exponential time procedure. The state space grows exponentially with n_1 and thus the computational burden for even moderately sized problem instances may be excessive. There are a number of features of the DP which can be exploited to substantially improve its efficiency.

VI.5.1 In-Core Space Requirements

The DP procedure advances from feasible sets at one stage to the feasible sets that could be generated at the subsequent stage. It is possible to exploit this stepwise process to significantly reduce the current storage requirements for data under consideration. The in-core space requirements can be substantially reduced by generating the sets in increasing order of their cardinality. Namely, all sets $\mathbf{X}$ with $|\mathbf{X}|=k$ are generated before proceeding to sets $\mathbf{X}$ with $|\mathbf{X}|=k+1$. Using this set generation procedure, only those sets

belonging to two consecutive stages need to be stored in-core at any one time. All other states would be stored on some secondary storage medium, for later use in the construction of the optimal sequence. This stepwise progression through sets of increasing cardinality can be seen in algorithm VI.1. The in-core information regarding sets is stored in the matrices **OLD** and **NEW** and the sets stored in **NEW** could only be generated from those stored in **OLD**.

VI.5.2 Filtering Methods and Heuristic Solutions

While the DP process examines far fewer sets than the number of feasible solutions, the total number of these feasible sets still grows at a rate exponential in n_1. The growth rate in the number of states is the primary complicating factor for these types of algorithms. For example, if $\mathbf{d}=[10,49,68,70,590,2,22,50,68,71]$ then the total number of feasible sets to be generated by the DP is 27,839,309,794,520,400. Assuming that the computer could perform all the necessary computations at a rate of 10^6 sets/second (a very fast computer), then the optimal solution for the problem would be determined by the DP in approximately 887 years; clearly not a feasible rate if the demand vector corresponded to the production requirements of one 8-hour shift. If technology progressed rapidly enough to produce a computer 1000 times faster than the one used above (i.e. a rate of 10^9 sets/second), then the optimal solution could be found in 324 days. Although this new machine would allow an optimal sequence to be determined within the scheduler's expected lifespan, in all likelihood, it would still not be timely enough for scheduling the aforementioned 8-hour production shift.

Hence, while the implicit enumeration algorithm is more efficient than explicit enumeration, if the DP must examine all of the possible sets in the state

space, an excessive amount of time could potentially still be required. Furthermore, even by generating the sets in increasing order of their cardinality, the number of feasible sets generated at any particular stage (i.e. all of those sets with equivalent cardinality) could conceivably be larger than the space available on the storage medium. This suggests the need for some form of filtering process in order to reduce the size of the algorithm's state space. These filtering methods could be used to eliminate the generation and storage of any intermediate states which could provably never lead to the optimal solution in the given problem instance.

One approach is to generate a schedule using some fast heuristic and to use this result as a filter to eliminate any sets from the DP's state space that would lead to a worse solution. As the optimal solution must be at least as good as that of the heuristic, continuing the search from any set that has a deviation larger than that of the heuristic value on the best "path" leading to it, must clearly produce a suboptimal solution. Depending upon the accuracy of the heuristic, this filter could potentially eliminate a considerable portion of the state space. Greedy procedures, similar to those created for the sum functions, can be developed for the minimax cases and these can be used for this filtering process. Two such heuristics for the general minimax problem will be outlined below.

VI.5.2.1 One Stage Heuristic

For this heuristic, a myopic decision rule is used at each stage k, ($k=1,2,\ldots,D_1$). If **X** is the cumulative production vector through the first k stages ($\sum_{i=1}^{n_1} x_{i1k} = k$), then the one stage decision rule is to schedule the product p in

stage k+1 which best continues this schedule. Thus, the product p which minimizes

$$\eta(\mathbf{X},p) = \| \Gamma(\mathbf{X} + e_p) \|_1$$

would be scheduled at stage k+1. This simple, one pass heuristic is very easy to implement. However, it provides no consideration for the effect that this decision will have on the deviations which occur in any future stage.

VI.5.2.2 Two Stage Heuristic

If $\mathbf{X}$ is the cumulative production through the first k stages, then the two stage heuristic proceeds as follows:

1. For each product p which can be scheduled at stage k+1, tentatively schedule product p for stage k+1 and calculate $\eta(\mathbf{X},p) = \| \Gamma(\mathbf{X} + e_p) \|_1$. Let $\mathbf{X}_p = \mathbf{X}+e_p$.
2. Tentatively schedule the product q for stage k+2 which minimizes $\eta(\mathbf{X}_p ,q) = \| \Gamma(\mathbf{X}_p + e_q) \|_1$.
3. Let $\beta(p) = \max\left\{\eta(\mathbf{X},p),\ \eta(\mathbf{X}_p,q)\right\}$.
4. Choose the product p^* which minimizes $\beta(p)$ and schedule this product for stage k+1.

The two stage heuristic examines the deviations at consecutive stages in its decision making process. The reason for using this heuristic is its ease of use and the assumption that, by considering an additional stage, it will provide better solutions than the one stage heuristic. As with the one stage heuristic, no consideration is made for the effect of the decision at this stage on the deviations at future stages.

VI.5.2.3 Implementation of the Heuristics

Together, the one- and two-stage heuristics are very easy to understand and implement. It is possible to exploit the structure of these heuristics in such a way that their running time is very fast. One such approach for the one stage heuristic will be outlined briefly in this section.

Define Γ_i, i=1,2,..,n_1, to be the i^{th} column of the gamma matrix, and $\mathbf{X}$ to be the cumulative level 1 production vector through k stages. Let g($\mathbf{X}$) and c($\mathbf{X}$) be two vectors of dimension nx1. Then proceed as follows:

1. Initialize k=0, $\mathbf{X}=\mathbf{0}$, g($\mathbf{X}$)=c($\mathbf{X}$)=$\mathbf{0}$. Construct the Γ matrix.
2. Then for stage k+1, k=0,1,..,D_1-1, if $x_{p1k}+1 \le d_{p1}$, p=1,2,..,n_1, calculate $c(\mathbf{X}+e_p)=g(\mathbf{X})+\Gamma_p$ and $c_p=\| c(\mathbf{X}+e_p) \|_1$.
3. Schedule the product p^* with the minimum c_p value for production at stage k+1.
4. Update, k←k+1, $g(\mathbf{X}+e_{p^*}) = g(\mathbf{X}) + \Gamma_{p^*}$, $\mathbf{X} \leftarrow \mathbf{X}+ e_{p^*}$.
5. If k=D_1 then stop, otherwise return to step 2.

This procedure illustrates that if the product i was scheduled at stage k+1 then $\Gamma(\mathbf{X}+e_i) = \Gamma\mathbf{X}+\Gamma_i$.

In the procedure, g($\mathbf{X}$) is the vector of deviations from the cumulative production for the first k stages [$g(\mathbf{X})=\Gamma\mathbf{X}$] and c($\mathbf{X}+ e_p$) is the temporary vector of deviations which would be obtained if product p was to be scheduled in stage k+1, [$c(\mathbf{X}+ e_p)=\Gamma(\mathbf{X}+ e_p)$]. The heuristic passes through each stage k only once and passes through each eligible product p only once for determining which product to schedule at any particular stage. It is possible to eliminate much unnecessary arithmetic by performing calculations in this vector format. The

value $\Gamma(\mathbf{X}+ e_i)$ need not be calculated in full for each i; each column Γ_i is simply added individually to $\Gamma\mathbf{X}$ to determine which product provides the best continuation of the schedule. Once the product p, which provides this best continuation, has been determined the matrix $\Gamma\mathbf{X}$ is updated to $\Gamma\mathbf{X} + \Gamma_i$. A similar (though only slightly more complex) vector method is required to run the two stage heuristic. Using this approach, both heuristics can be designed to run very quickly and will be used as filters in the testing of the DP algorithm. Each heuristic can be run prior to the DP and the best heuristic value (SCREEN in algorithm VI.1) can be used as the filter.

VI.5.3 Addressing of Sets

If the filtering device employed by the DP works effectively, then a considerable portion of sets at each stage will be eliminated. At any particular stage, it can not be known, *a priori*, which of these sets or even how many of the sets may actually be fathomed. Ways must be devised which allow for the efficient storing and addressing of the remaining sets. This is important for the DP, as a set at one stage could have been generated from several sets of the previous stage and only the product providing the best continuance of the schedule need be stored. Hence, there must be a way of determining if a set has been previously generated and, if so, being able to locate this set in order that a comparison of maximum deviation values along the best path to it can be performed. Simply storing sets in the order of their creation and searching through this list would be inefficient.

One approach is to address (or encode) each set in such a way that a one-to-one correspondence between each address and each set exists and to store the information for the sets created at each stage in increasing order of

this address index. If this ordering system is maintained, then the sets for the stage can be stored in a sorted array and a binary search along the address index within this array would quickly determine if the set had been generated previously. Furthermore, if the maximum and minimum values of the addresses generated-to-date are maintained, then, when a new set is generated, it could be immediately determined if its address fell within the range of addresses previously generated; thus, it would be known whether to simply update the list of sets or to search the addresses to determine whether the set had been generated previously.

Suppose that the total number of possible states, **X**, in a problem is N. Define an addressing scheme to be *compact* if the maximum address is the number N-1 and each state within the problem is addressed by a unique integer in the range $[0,N-1]$. One such compact addressing scheme will be described.

Assume, without loss of generality, that the level 1 products are numbered such that $d_i \le d_{i+1}$, $1 \le i \le n_1-1$. Let $d_0=0$ and define the address units of product i to be,

$$u(i) = \prod_{j=0}^{i-1} (d_j+1).$$

Let $\mathbf{U}=[u(1),u(2),\ldots,u(n_1)]$ be the vector of these address units. The address for each set **X** will be the inner product,

$$\mathbf{UX} = \sum_{i=1}^{n_1} u(i)x_i .$$

This addressing scheme provides a unique address for each possible set **X** and also has the property of being compact. Hence for demand vector **d**, the total number of sets **X** which could possibly be generated is **Ud**+1; the maximum address plus the set corresponding to **0**.

For example, consider the product demand vector $\mathbf{d}=[1,2,3,4]$. Using the above procedure, the address units will be,

$u(1)=1$, $u(2)=(2)(1)=2$, $u(3)=(3)(2)(1)=6$, and $u(4)=(4)(3)(2)(1)=24$, hence, $\mathbf{U}=[1,2,6,24]$. The maximum possible number of sets, $\mathbf{X}$, which could be generated by this demand vector is $\mathbf{Ud}+1 = (1)(1)+(2)(2)+(3)(6)+(4)(24)+1 = 120$.

Unfortunately, this maximum address index grows exponentially with the number of products. Moreover, the number of significant figures (i.e. the largest available integer) within any computer language is limited. Hence, the address for the sets may necessarily have to be broken up into several segments (dimensions) if the problem has a large number of different products and the total demand is large. The address within each dimension will correspond to a certain fixed group of products (i.e. products 1 to 5 could be used for the address value in dimension 1, products 6 to 10 could be used for the address value of dimension 2, etc.) where products within a group are ordered in increasing order of their demand and each index has an addressing scheme as defined above.

Consider the example presented above and assume that a very restrictive computer language must be employed which does not permit integer values to exceed 12. If the encoding system above is used, then 107 sets would have addresses which exceed this maximum and some alternative scheme must be developed. Partition $\mathbf{d}$ in the following way;

$$\mathbf{d}' = [\mathbf{d}^1 \mid \mathbf{d}^2] = [1,4 \mid 2,3].$$

Then use the above encoding scheme on each of $\mathbf{d}^1$ and $\mathbf{d}^2$ separately. If $\mathbf{U}' = [\mathbf{U}^1 \mid \mathbf{U}^2]$ is the partitioned address vector, then $u^1(1)=1$, $u^1(2)=(1)(2)=2$,

$u^2(1)=1$, $u^2(2)=(1)(3)=3$ and hence $\mathbf{U'} = [1,2 \mid 1,3]$. The maximum integer corresponding to this partitioning will be the maximum of the inner products of either $\mathbf{U^1d^1} = (1)(1)+(2)(4) = 9$ or $\mathbf{U^2d^2} = (1)(2)+(3)(3) = 11$. Since neither of these values exceeds the maximum allowable integer, 12, this two-dimensional partitioning can be employed to address the sets. Hence, in the DP algorithm, two matrix elements would be needed to address each set. [Note that if no partitioning of the demand vector into two groups allows for a feasible addressing scheme, then partitions into three or more groups would be necessary].

The sets generated at a particular stage would then be ordered in increasing value of the addresses within each dimension and by increasing magnitude of their address index (sorted first along index 1, then along index 2 etc.). If a search must be performed to determine if a set has been generated previously, binary searches can be performed within each dimension index by increasing magnitude of the dimension. The array that stores the relevant information for each set must contain; the address in each dimension, the minimum value of the objective encountered along the best path to the set ,and the last product produced along this best path.

VI.5.4 Optimal Sequence

The optimal production sequence is constructed only after the determination of the objective value (step 12 in algorithm VI.1). The process for finding this sequence is to work back through the stages in decreasing order of the cardinality of their set sizes to determine which product is scheduled at each stage. This requires examining the generated sets placed onto secondary

storage by the DP. The sequence construction and set examination process is greatly facilitated by the addressing scheme.

At stage D_1 only one set would have been generated; the set which led to the optimal objective. As the DP stores the last product to be produced along the best path to a set, this product will also be the last product produced in the optimal sequence. Furthermore, as the set at the final stage must necessarily be the set **X=d**, the address of this set will be **UX=Ud** (where **U** may have been partitioned to produce an address consisting of more than one dimension) and only one such set will exist. At this address, the last product on the best path leading to it will also have been stored. Assuming that this last product is product i, then i will be the last product to be scheduled and will therefore be the product sequenced in stage D_1. In order to determine which product is sequenced at stage D_1-1, the last product produced along the best path to the set $\mathbf{d}-\mathbf{e}_i$ must be found. This is equivalent to finding the information for the set with the address $\mathbf{U}(\mathbf{d}-\mathbf{e}_i)$ amongst those sets in secondary storage. Once this address has been found, the product to be sequenced at this stage will be the last product produced along the best path to this set; which is, once again, stored with the set's address. This sequencing process will continue until a product has been sequenced at each of the stages from 1 to D_1.

Note how each set has been uniquely encoded and identified by its address. Hence, the sets need not be stored explicitly; the address of the set completely specifies which products and the quantity of each product that corresponds to this address in a compact representation of the set's makeup. The DP sorted the sets at each stage in increasing magnitude of their addresses and it is therefore possible to write these addresses as a block to

secondary storage preserving this ordering. It is easy to record the number of sets generated at each stage and thus to find a specific address during the sequence construction phase requires the determination of where the sets corresponding to a particular stage begin (which also implies, by the way in which the sets were written to secondary storage, that the end of the range of sets of equal cardinality can also be determined). Then a binary search of the addresses corresponding to the stage will quickly locate the address of the set which is being sought. Hence, the optimal sequence can be constructed in time proportional to the number of sets generated by the DP.

VI.6 Summary

This chapter has shown that the minimax problems can be formulated in convenient matrix notation. A DP procedure for the optimization of the minimax problems was presented. This specifically implies that an efficient, implicit enumeration approach exists for determining optimal solutions to multi-level problems. The minimax DP procedure was modified to provide a DP algorithm for determining solutions to multi-level sum functions. The time and space requirements of the DP were developed and from these it could be seen that this implicit enumeration algorithm provides a significant improvement over explicit enumeration procedures.

Implicit enumeration is, however, an exponential time process. Therefore, filtering methods should be employed to screen out any solutions which would clearly be suboptimal. Two fast heuristics were developed for use by the DP to fathom these inferior solutions from the state space. As a set at some stage could potentially be generated from any one of several sets at the preceding stage, efficient ways of accessing and storing the data become

increasingly important. This involved the creation of addressing methods for each set, which become even more important when the sets from the state space are reduced by a filter. The arrays storing the data at each stage would consist of only as many rows as the number of sets generated at that stage and the sets would be stored within the arrays in increasing order of their address. This would allow for a set to be found (or it could be determined that the set had not been generated previously) by performing a bisection search along the address values. If the size of the problem data required that more than one address index would be needed to store each set, then the bisection search would be performed within each dimension of the address index. It was noted that by generating the sets in increasing order of their cardinality, only two consecutive stages would ever need to be stored in-core at any time. The other stages would be held in secondary storage for later use in the construction of the optimal sequence.

In the minimax DP, the maximum deviation along the best path to a given set could occur very early in the state space. Hence, the fathoming process could be expected to start very early in the minimax problems and that a filtering method could prove to be substantially beneficial in the pruning of the state space. The implementation and testing of this DP will be examined in the next chapter.

CHAPTER 7 EXPERIMENTATION USING THE DP ALGORITHM

VII.1 Introduction

Chapter VI introduced the DP algorithm for optimally solving general multi-level problems which, although growing exponentially with n_1, is far more efficient than any explicit enumeration procedure. While this growth rate is exponential, computational tests of the algorithm are necessary to determine its performance capabilities, characteristics and limitations.

The most important test criterion must be the time required to determine optimal solutions to problems of a predetermined size. If tests reveal that problems of all practical interest can be optimally solved in time sufficient for realistic applications, then the DP could be used for any such implementations. If, however, reasonable time constrains the size of the problems which can be solved, then it is necessary to know what sized problems are so constrained. The results obtained from the experimentation (on whatever sized problems are optimized in the testing) could then serve as benchmarks to evaluate the accuracy of heuristics which would be subsequently developed for these problems.

"Reasonable time" is a rather arbitrary term which depends upon the assumptions made and the appropriate time horizon of the scheduling application. Since Monden's (1983) applications for scheduling mixed-model, JIT processes were initially for automobile assembly, the types of problems to be tested in this chapter can be thought of as those that would correspond to similar

such production systems. A typical automobile assembly process generally consists of a number of production shifts in a day (i.e. two at Toyota, three at General Motors). Therefore, for testing purposes, the time horizon is assumed to be one 8-hour production shift. If, in this shift, finished products are produced at the rate of one per minute, then the total production over the entire shift would be 480 units. Therefore, practical applications will be assumed to require the production of approximately 500 units of finished (or level 1) product. The concept of time reasonableness must also be addressed in some manner, as problems can always be generated for which so much descriptive data is required as to prove computationally prohibitive for use with the DP. Hence, if the solution time required by the DP could be considered timely enough to act as input for the scheduling of these 500 products in one such 8-hour production shift, then it would be deemed "reasonable".

A second major test criterion is to ascertain whether the storage capacity constrains the size of the problems which can be optimized in this reasonable solution time; assuming that a filtering scheme is employed concurrently. The rate of growth for the generation of feasible sets was identified in the previous chapter as one of the main concerns with using this type of DP algorithm; this is the so-called 'curse of dimensionality'. Without the use of a filtering mechanism, the state space of virtually any problem of a practical size would require prohibitive amounts of storage; both in-core and on secondary storage. There is a direct link between the effectiveness of the filter and the storage space required. Thus, as disk storage space is not of infinite capacity, it is necessary to know whether storage space (and therefore the effectiveness of

the filter) constrains the size of the problem upon which the DP can be implemented.

The experimentation process could also reveal characteristics inherent to the solution of multi-level problems and may uncover important relationships not readily apparent from the problem formulation. Hence, testing must be performed on a significant variety of problem instances. These computational experiments are the focus of this chapter.

VII.2 Testing Preliminaries and Assumptions

As a polynomial time algorithm exists for single-level problems, all experimentation is to be performed on multi-level problems. In order to test the DP, requisite limiting assumptions, in addition to those mentioned above, are necessary. These assumptions will be outlined below.

VII.2.1 Filters

The problems to be tested must be large enough to be of sufficient interest. To evaluate problems of even moderate size using the DP would require the generation of a significantly large number of intermediate sets. If all of these sets have to be generated, then the storage requirements and the computational time would very quickly become excessive (see section VI.5.2). This necessitates the introduction of some screening mechanism to eliminate those sets which could clearly be shown to produce suboptimal results. The filter to be used is the best objective value of the one- and two-stage heuristics described in the previous chapter.

It would seem that the two-stage heuristic should produce solutions that are at least as good as those produced by the one-stage heuristic.

Preliminary testing of the heuristics revealed only one instance (out of several hundred trials) where this was not so. This one case occurred as a result of how the heuristics were programmed to treat tied solutions in the intermediate stages. When the decision rule for ties was changed, the result of this one instance was reversed (i.e. the two-stage heuristic provided the better solution). However, since the running time of both heuristics is negligible, as a precaution both heuristics were run and the filter value chosen was the best heuristic solution calculated.

The screening mechanism works in the following way. If a set is generated such that the best objective value along any path through this node exceeds the heuristic value, then no further branching from this node to the remainder of the state space need occur. The reduction in the search effort depends heavily upon the pruning potential, and hence the accuracy, of these myopic heuristics. In using the heuristics, it must be determined whether or not their accuracy is independent of the size and assumed parameters of the test problems. If accuracy is influenced by the parameters, then alternative filters may have to be considered.

VII.2.2 Generation of Product Requirements

For a particular problem instance, n_1 will be set at a certain value and it will be ensured that the product requirements sum to a fixed value of D_1. In order to do this, n_1-1 product demand values will be generated in some range (approximately D_1/n_1) and the demand for the n_1^{th} product will be the remaining quantity required to fix the production total at D_1. The time and space requirements have both been shown to be functions of the number of different

products. Thus, it is also necessary that the product requirements are generated subject to the constraint that $d_i \geq 1$, $i=1,...,n_1$.

VII.2.3 Generation of the Number of Parts and the Part Requirements

Recall that t_{ijl} is the number of units of output i at level j required for the production of each unit of product l. The value of t_{ijl} will be randomly generated for all i,j,l, $j=2,...,L$, $i = 1,...,n_j$, $l = 1,...,n_1$, in such a way that t_{ijl} is an integer uniformly distributed in the interval $[0,R_j]$; where R_j represents the range of the units of production for level j. The number of different parts, n_j, at level j will be set as a randomly generated integer in the respective intervals;

$$n_2 \varepsilon [15,25], \qquad n_3 \varepsilon [26,50], \qquad n_4 \varepsilon [51,75].$$

The number of distinct parts at any level is set at one of the arbitrary amounts in these intervals and the values selected are used to approximate relative differences in the numbers of different parts at the various levels.

VII.2.4 Generation of Problem Instances

In order to generate a problem instance, the above parameters will be fixed in some manner and these values will serve as inputs to the DP. The time required for the algorithm will be recorded along with information regarding the number of sets generated, the accuracy of the heuristics and how effective the filtering method performs. For experimental purposes, several replicates must be generated at the fixed values of the parameters in order for any types of conclusions to be drawn.

Fixing the parameters entails setting a value for D_1 and n_1, fixing the number of production levels L, and setting a value for the range, R_j of the part requirements at each level. For programming simplicity, only a two-dimensional

addressing scheme will be employed, which will limit the number of different products which could be examined. However, if this two-dimensional scheme appears to be unnecessarily restrictive, then additional dimensions can be added. The DP was coded in FORTRAN 77 and run on a VAX 8530. The use of the VAX implies that the running times of the DP are dependent upon the speed of this machine; a different computer would provide different solution times. Hence, any time results apply explicitly to the VAX itself. However, relative differences in the solution times of different problems will be provided by this testing.

VII.3 Two Level Experiment

The first test of the DP's capabilities was performed on problems with L=2 production levels, where the total demand for the level 1 output was set at D_1=1000. With these parameters fixed, problem instances were generated in which the number of different products was set at the values n_1=8 and n_1=10 and in which the range values at level 2 were set at R_2=20 and R_2=100. The factor involving the levels of n_1 will be referred to as the "size" of the problem and the factor involving the levels of R_2 will be referred to as the "range" of the problem.

The problem parameters are fixed at these values for a number of reasons. Since two production levels is the minimum number for a multi-level system, if the DP is incapable of solving problems with L=2, then it would be futile to extend the testing to problems with more levels. Hence, initially testing two level problems is appropriate. Setting the total production at one thousand is twice the amount considered appropriate for one production shift (see section VII.1) and could be considered as an attempt to potentially extend the DP to the limit of its capabilities. In a mixed-model manufacturing process, it could realistically be expected that eight to ten different types of product would be

produced. Although the range of each product's part requirements cannot be "too variable" in a JIT system (otherwise the JIT system could not function effectively), the range values of twenty and one hundred allow for the testing to be performed over a sufficiently broad interval. These range values could also be considered as set at extreme levels in an attempt to push the testing of the DP to its limits. Depending upon how the DP solves problems with these parameter values, subsequent adjustments could be made, either up or down, in any supplementary testing.

The first test examined each possible treatment combination and thus provided a 2x2 factorial experiment. Fifteen replicates were randomly generated at each of these treatment combinations and the results are summarized in tables VII.1-4. For ease of interpretation, the replicates within each table have been ordered by the increasing amount of the solution time required by the DP; hence, the first replicate in each table required the minimum amount of time, the last replicate required the maximum amount of time, and the time taken by the eighth replicate is the median time for that particular treatment combination.

Table VII.1

n_1=10, R_2=100, D_1=1000, L=2

	Optimal Value	Heuristic Value	Heuristic / Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	89.923	96.965	1.078	8.10×10^{16}	54,539	6.72×10^{-11}	16.15
2	88.732	95.526	1.076	4.13×10^{15}	61,316	1.48×10^{-9}	18.47
3	74.768	84.088	1.124	2.59×10^{16}	87,839	3.38×10^{-10}	20.40
4	84.955	96.123	1.131	2.93×10^{16}	83,386	2.84×10^{-10}	23.45
5	91.173	99.666	1.093	4.24×10^{16}	85,087	2.00×10^{-10}	23.53
6	84.955	96.123	1.131	2.93×10^{16}	83,386	2.84×10^{-10}	24.18
7	75.281	85.554	1.136	3.78×10^{16}	98,832	2.61×10^{-10}	24.22
8	80.911	90.904	1.123	2.32×10^{16}	89,060	3.83×10^{-10}	24.85
9	110.978	118.196	1.065	1.01×10^{16}	113,972	1.12×10^{-7}	28.60
10	82.645	96.433	1.166	5.03×10^{16}	147,963	2.94×10^{-10}	38.65
11	102.686	114.585	1.115	1.68×10^{16}	163,943	9.72×10^{-10}	46.26
12	66.314	78.254	1.180	2.78×10^{16}	231,489	8.31×10^{-10}	51.32
13	105.672	120.871	1.143	2.47×10^{15}	308,765	1.25×10^{-8}	79.02
14	95.607	112.236	1.173	7.53×10^{16}	309,150	4.10×10^{-10}	91.82
15	89.877	113.383	1.261	6.22×10^{15}	670,764	1.07×10^{-8}	191.12

Table VII.2							
n_1=10, R_2=20, D_1=1000, L=2							
	Optimal Value	Heuristic Value	Heuristic Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	18.027	19.201	1.065	7.30×10^{14}	30,106	4.11×10^{-14}	8.77
2	17.222	18.730	1.087	8.12×10^{15}	59,497	7.32×10^{-10}	17.20
3	15.159	17.053	1.124	1.90×10^{16}	126,037	6.63×10^{-10}	28.23
4	16.058	18.210	1.134	9.31×10^{15}	132,312	1.42×10^{-9}	33.20
5	18.843	20.310	1.077	2.38×10^{15}	105,141	4.40×10^{-11}	34.02
6	16.724	18.851	1.127	6.43×10^{16}	119,188	1.85×10^{-10}	34.78
7	19.525	21.743	1.113	2.11×10^{16}	128,510	6.06×10^{-10}	37.68
8	21.120	22.981	1.088	1.08×10^{15}	131,869	1.21×10^{-8}	39.13
9	15.613	17.835	1.142	5.54×10^{16}	148,878	2.68×10^{-10}	40.13
10	16.154	18.502	1.145	1.74×10^{16}	179,731	1.03×10^{-10}	42.42
11	19.490	22.226	1.140	2.63×10^{16}	220,339	8.37×10^{-10}	61.58
12	18.209	20.908	1.148	1.03×10^{16}	277,622	2.68×10^{-9}	64.83
13	18.763	19.888	1.059	1.61×10^{16}	256,821	1.59×10^{-9}	71.65
14	18.138	21.377	1.178	1.74×10^{15}	320,777	1.84×10^{-9}	81.32
15	18.733	20.227	1.079	3.23×10^{15}	410,202	1.26×10^{-10}	109.20

Table VII.3							
n_1= 8, R_2=100, D_1=1000, L=2							
	Optimal Value	Heuristic Value	Heuristic Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	77.275	79.904	1.034	1.85×10^{15}	15,219	8.19×10^{-10}	1.68
2	81.259	91.388	1.124	2.90×10^{13}	33,393	1.14×10^{-7}	2.07
3	81.227	89.877	1.106	5.36×10^{14}	16,348	3.04×10^{-8}	2.52
4	77.671	85.094	1.095	1.10×10^{14}	17,717	1.61×10^{-8}	3.35
5	71.414	76.692	1.073	3.58×10^{14}	19,733	5.51×10^{-9}	3.37
6	76.884	83.066	1.080	1.41×10^{14}	27,405	1.94×10^{-8}	3.95
7	76.450	83.981	1.098	1.31×10^{13}	31,531	2.38×10^{-7}	4.48
8	70.697	81.899	1.158	7.08×10^{13}	29,208	4.12×10^{-8}	4.50
9	87.610	98.185	1.120	2.03×10^{14}	33,507	1.64×10^{-8}	6.13
10	69.366	79.493	1.145	1.24×10^{15}	42,258	3.38×10^{-9}	6.50
11	89.393	99.085	1.112	5.64×10^{14}	41,905	7.43×10^{-9}	7.85
12	85.751	94.088	1.097	2.91×10^{13}	46,395	1.59×10^{-7}	8.03
13	80.338	91.005	1.132	1.92×10^{14}	60,926	3.16×10^{-8}	8.23
14	82.364	97.016	1.177	6.54×10^{14}	61,760	9.43×10^{-9}	10.25
15	102.117	115.793	1.133	1.07×10^{13}	135,820	1.26×10^{-5}	20.03

Table VII.4

n_1=8, R_2=20, D_1=1000, L=2

	Optimal Value	Heuristic Value	Heuristic Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	13.671	13.991	1.023	1.37×10^{15}	11,619	8.44×10^{-10}	1.75
2	14.937	16.257	1.088	5.34×10^{13}	20,325	3.80×10^{-8}	2.80
3	13.495	14.938	1.106	1.12×10^{14}	19,773	1.76×10^{-8}	3.10
4	15.510	17.259	1.127	1.03×10^{14}	24,919	2.41×10^{-8}	3.78
5	16.461	17.634	1.071	9.06×10^{12}	23,714	2.61×10^{-7}	3.82
6	16.914	18.857	1.114	4.74×10^{13}	25,638	5.40×10^{-10}	4.72
7	17.802	18.971	1.065	5.39×10^{11}	33,477	6.21×10^{-6}	5.38
8	18.976	21.085	1.111	8.70×10^{12}	46,280	5.31×10^{-7}	7.65
9	18.519	20.394	1.101	2.46×10^{13}	45,884	1.86×10^{-6}	7.98
10	17.499	19.838	1.133	1.75×10^{13}	73,216	4.16×10^{-7}	10.22
11	20.382	23.703	1.162	9.01×10^{12}	68,472	7.59×10^{-7}	11.05
12	18.624	21.203	1.138	1.29×10^{14}	90,677	7.02×10^{-8}	12.50
13	13.261	16.664	1.256	4.95×10^{14}	82,839	1.67×10^{-8}	15.05
14	20.940	22.693	1.083	4.91×10^{14}	107,509	2.18×10^{-7}	15.65
15	19.895	21.766	1.094	9.97×10^{12}	91,815	9.20×10^{-7}	16.48

VII.3 .1 Analysis

Because of the 2x2 factorial design, ANOVA's of the output can be constructed for analyzing both the time taken to determine optimal solutions and for evaluating the accuracy of the heuristics used as filters. The ANOVA for evaluating the solution time is;

Source	SS	d.f.	MS	F_0
Range	16.110	1	16.110	.022
Size	23,653.659	1	23,653.659	33.444
Interaction	12.050	1	12.050	.017
Error	39,605.993	56	707.249	
Total	63,287.812	59		
$F_{.01,1,56}=7.08$			$F_{.25,1,56}=1.35$	

The results of this ANOVA indicate that the solution time of the DP significantly depends upon the the size of the problem, but does not depend upon the range of each products' part requirements. Thus, problems in which there are more product types require significantly more solution time than problems with fewer product types. This conclusion is consistent with the bounds for the time requirements developed for the DP in the previous chapter. An unforeseen result is that the time required for solving problems in which the part requirement ranges differ is not significant at all. This indicates, quite surprisingly, that the algorithm's solution time is essentially independent of the part requirements make-up of the individual products. No significant interaction between the different levels of the size and range treatments is apparent.

One of the assumptions for using an ANOVA is that the variances of the time for each treatment combination are equal. By examining the distributions of the times shown in tables VII.1-4, this assumption might be called into question. A nonparametric, Mann-Whitney test was subsequently employed to counteract this potential bias and to further substantiate the conclusions drawn regarding solution times. Mann-Whitney tests are used to compare two random samples from independent populations and require no assumptions regarding the normality of the data or of the equality of their variances. Descriptions of this test can be found in any nonparametric statistics text; for example, see Conover(1980), Mosteller & Rourke(1973), Kendall(1962). The test is based upon a pooling of the two populations and an ordering of this combined sample by increasing magnitude of the variable of interest. A test statistic, W, is obtained by calculating the sum of the ranks of one of the populations and a comparison of this statistic to tabulated values is then made. The restriction imposed by this test is that all of the treatment combinations of the 2x2 experiment cannot be considered simultaneously. This requires that several two-way comparisons must be made. These comparisons can be summarized below.

1. Fixing the range at R_2=20 and testing to see if the times differ at this range depending on the value of n_1 produces W=126. At a level of significance less than 0.01%, the conclusion can be drawn that the times taken for n_1=8 and n_1=10 at this value of R_2 are different.
2. Fixing the range at R_2=100 and testing to see if the times differ at this range depending on the value of n_1 produces W=122. At a level of significance less than 0.01%, the conclusion can be drawn that the times taken for n_1=8 and n_1=10 at this value of R_2 are different.

3. Combining the data for each value of size over the two range levels and testing to see if the times differ regardless of range but depending on the value of n_1 produces W=477. At a level of significance less than 0.01%, the conclusion can once again be drawn that the times taken for n_1=8 and n_1=10 are different regardless of the value of the range.
4. Fixing the size at n_1=8 and testing to see if the times differ at this size depending on the value of R_2 produces W=257. This W is significant only at the 31.95% level and the conclusion can be drawn that the times required at R_2=20 and R_2=100 do not significantly differ at this value of n_1.
5. Fixing the size at n_1=10 and testing to see if the times differ at this size depending on the value of R_2 produces W=258. This W is significant only at the 29.98% level and the conclusion can be drawn that the times required at R_2=20 and R_2=100 do not significantly differ at this value of n_1.
6. Combining the data for each value of range over the two size levels and testing to see if the times differ regardless of size but depending on the value of R_2 produces W=963. This is only significant at the 48.25% level, hence, the conclusion can be drawn that the times taken for R_2=20 and R_2=100 do not significantly differ regardless of the value of the size.

Examining the results of this series of nonparametric tests substantially strengthens, and further confirms, the conclusions drawn from the ANOVA; namely, that the time required to solve a problem using the DP depends significantly upon the number of different products in the problem. The time

required does not significantly depend upon the range of the part requirements. This is a significant result as it implies that in order to evaluate the effectiveness of the DP with a set number of distinct parts in terms of time requires that only variations in the value of n_1 need be considered.

These results are very interesting, but the actual solution times have not been addressed as yet. That is, the actual values of the times themselves are of considerable interest because it is necessary to know whether the DP can be realistically implemented for problems of anything other than a trivial size. Below is a table for the 2x2 experiment indicating the average solution time required for each treatment combination and for each factor.

	R_2=20	R_2=100	
n_1 = 8	8.1 min	6.1 min	7.16 min
n_1 = 10	46.9 min	46.8 min	46.8 min
	27.5 min	26.4 min	27.0 min

This table clearly demonstrates where the time differences occur. A significant increase in the solution time occurs when moving from problems in which the size is n_1=8 to those in which the size is n_1=10; while no such increase is apparent when moving from range values of R_2=20 to range values of R_2=100. Furthermore, these average times indicate that, with D_1=1000, two level problems in which n_1=8 can be optimally solved in a realistic amount of time. The problems in which n_1=10 could perhaps be considered solveable in a realistic period of time (for an 8-hour production shift) if the information for the production shift was made available sufficiently in advance of production. However, if a faster computer was employed and a professional programmer could code the DP more

efficiently, then even the problems in which n_1=10 could be optimally solved in an amount of time requisite for realistic applications.

The second criterion for evaluating the effectiveness of the DP is the storage requirement. As the storage issue is directly related to the success of the filtering system, and hence the accuracy of the heuristics, this entails examining the effectiveness of the heuristics. In evaluating the heuristics' accuracy, the scaling effect of the range values prevents direct comparisons based strictly upon deviations of the absolute magnitude between the heuristic and optimal values. Contrasting the percentage errors provides a better comparative measure. The ANOVA for the percent overestimation of the optimal solution by the heuristic is;

Source	SS	d.f.	MS	F_0
Range	15.100	1	15.100	.780
Size	19.837	1	19.837	1.025
Interaction	12.789	1	12.789	.661
Error	1083.095	56	19.340	
Total	1130.821	59		
$F_{.01,1,56}$=7.08			$F_{.25,1,56}$=1.35	

This ANOVA table indicates that the heuristic performed equally accurately over all levels of both the size and the range of the problems. Furthermore, no interaction between the factors could be detected. This result is important as it demonstrates that the simple one- and two-stage heuristics can be used with a confidence that their accuracy depends upon neither the size nor the range of the problem. As the heuristics are essential for the screening of the

suboptimal sets, the solutions produced must be equally accurate for problems of any size or range. As this accuracy is reflected in the results of the ANOVA, it can be concluded that the screening potential of the heuristics is independent of the size and the range of the problem data.

More remarkable than this accuracy outcome is the screening capability of these simple-minded heuristics. In the worst instance, the maximum percentage of the total possible number of feasible sets generated in any of the replicates over all levels of the factors was 1.26×10^{-5}. Phrased another way, this implies that the percentage of screened sets in any problem instance was at least 99.9987%. Thus, the use of very non-technical heuristics ensures that the potentially vast state space is significantly reduced to more manageable levels. In no problem instance did storage capacity become a constraining factor. The in-core storage is a function of the number of sets generated by two consecutive stages. At no time did the number of sets in consecutive stages ever exceed 2000 and thus in-core storage never constrained the size of the problem to be solved. Hence the screening method employed effectively eliminates the potential storage space constraint.

However, the sheer magnitude of the set space size belies the actual number of sets which must be examined in any given problem. The sets retained by the DP correspond to those sets through which feasible paths to the optimal solution could potentially pass. From each of these feasible sets, all extensions to sets of cardinality in the subsequent stage must be examined before their feasibility can be assessed. In the experiment, the average number of feasible sets for each problem was 110,259 and thus, on average, $110,259 \times n_1$ sets had

to be generated in each problem. This accounts for the time required by the DP to determine optimal solutions.

The results of this 2x2 experiment have demonstrated two points. Firstly, that the solution time for the DP depends upon the size of the problem but does not significantly depend upon the range of the part requirements. And secondly, that two-level problems of size n_1=8 can be optimally solved in reasonable time and that perhaps problems of size n_1=10 can also be solved in a reasonable period of time. However, the average solution time taken by the larger sized problems is considerably longer. Knowing the time required to solve problems of size n_1=10 is of considerable importance. It would be desireable to have more data on a larger number of test problems of this size. This suggests that further investigation of the two level problems is desirable .

VII.3.2 Supplementary Testing of Two Level Problems

As the value of the range had no significant affect on the solution time, there was no further necessity to consider this as a factor for the two level problems. Thirty additional replicates were generated for problems with n_1=10 and the range set arbitrarily at R_2=100. The results of these problems are shown in table VII.5.

The average solution time for these additional problems is 57.94 minutes. This average time is 11 minutes more than the average time for those replicates generated in the factorial experiment. However, the largest time is 342 minutes which is considerably longer than the times generated by the initial experiment. This instance would appear to be an outlier as the average time for the remaining 29 problems is 48.12 minutes; which is comparable to the average

time for the replicates generated in the factorial experiment. This outlier demonstrates that not all of the instances of this size can necessarily be solved in reasonable time. However, the conclusion to be drawn is that problems of size n_t=10 can generally be solved to optimum in approximately 47 minutes although the actual time may vary about this value. Furthermore, this average solution time is quite consistent although there is a potential for certain rare instances to require an excessive amount of time.

In no instance did the storage capacity ever become a constraining factor; indicating that the heuristics used for filters work very effectively. In general, there does not appear to be a relationship between the accuracy of the heuristic and the solution time. The heuristic overestimates the optimal solution in the range of 5.5% to 22.3% (excluding the outlier) with a mean accuracy of 11.8%. Only for the outlier, where the heuristic value exceeds the optimal solution by 30%, does a link appear between the accuracy and the time. The problem data for this outlier problem do not significantly differ from the data of the other problems. No apparent cause appears for the heuristic's inaccuracy in the case of the outlier; the heuristic simply seems to have provided a poor solution.

Table VII.5

n_1=10, R_2=100,D_1=1000, L=2

	Optimal Value	Heuristic Value	Heuristic / Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	84.244	88.893	1.055	2.18×10^{16}	34,423	1.57×10^{-10}	9.97
2	74.479	82.851	1.112	7.62×10^{16}	40,724	5.34×10^{-11}	11.22
3	78.806	84.800	1.076	8.48×10^{13}	47,936	5.64×10^{-8}	11.48
4	86.806	94.187	1.085	3.07×10^{19}	61,047	1.98×10^{-10}	13.78
5	98.964	108.262	1.093	2.85×10^{15}	155,709	5.45×10^{-9}	14.25
6	87.102	96.951	1.113	1.74×10^{16}	57,843	3.32×10^{-10}	16.73
7	64.551	74.148	1.148	6.20×10^{15}	84,092	1.35×10^{-9}	18.65
8	72.429	82.663	1.141	2.76×10^{15}	90,482	3.27×10^{-9}	19.57
9	78.411	88.152	1.124	1.22×10^{17}	67,878	5.56×10^{-11}	20.15
10	85.166	92.391	1.084	1.83×10^{17}	82,729	1.07×10^{-11}	22.70
11	100.252	108.032	1.077	1.29×10^{15}	89,270	6.88×10^{-9}	22.75
12	87.658	95.742	1.092	5.49×10^{15}	111,540	2.02×10^{-9}	25.38
13	85.151	93.595	1.099	4.17×10^{16}	104,075	2.49×10^{-10}	26.50
14	78.736	87.885	1.116	1.09×10^{17}	101,905	9.28×10^{-11}	27.15
15	93.758	102.906	1.097	1.20×10^{15}	91,758	7.61×10^{-9}	27.72

Table VII.5 (cont.)

n_1=10,R_2=100,D_1=1000, L=2

	Optimal Value	Heuristic Value	Heuristic / Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
16	86.098	98.821	1.147	1.62×10^{17}	101,987	6.29×10^{-11}	29.40
17	80.777	93.262	1.154	6.19×10^{15}	143,933	2.32×10^{-9}	35.45
18	87.073	93.321	1.071	3.71×10^{16}	178,176	4.79×10^{-10}	39.80
19	66.314	78.254	1.180	2.78×10^{16}	231,489	8.31×10^{-10}	50.82
20	94.231	107.009	1.135	2.81×10^{16}	229,326	1.07×10^{-10}	55.30
21	88.189	107.940	1.223	1.80×10^{16}	205,538	1.13×10^{-9}	55.75
22	87.753	101.088	1.151	1.62×10^{16}	214,571	1.31×10^{-9}	62.04
23	98.039	105.317	1.074	4.56×10^{16}	227,264	4.98×10^{-10}	63.32
24	90.765	103.953	1.145	6.57×10^{15}	280,018	4.25×10^{-9}	71.70
25	91 754	98.399	1.072	1.05×10^{16}	205,352	1.93×10^{-9}	94.28
26	79.346	93.828	1.182	1.15×10^{17}	479,135	4.16×10^{-10}	114.13
27	90.282	107.384	1.189	6.90×10^{16}	452,734	6.55×10^{-10}	116.23
28	95.445	107.505	1.123	1.66×10^{16}	608,682	3.65×10^{-10}	156.05
29	101.512	113.748	1.120	2.32×10^{15}	629,271	2.71×10^{-8}	163.28
30	85.543	111.645	1.305	1.05×10^{17}	1,376,070	1.30×10^{-9}	342.85

From this experimentation, it appears that the mean solution time for problems of a given size remains reasonably consistent. The mean time was observed to increase substantially when the problem size was increased from n_1=8 to n_1=10. Therefore, it would be interesting to observe the magnitude of time increase if the size were to be increased from n_1=10 to some larger size. Table VII.6 shows the data for problems of size n_1=12.

Table VII.6

n_1=12, R_2=100, D_1=1000, L=2

	Optimal	Heuristic	Heuristic / Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	92.639	103.110	1.113	3.10×10^{16}	293,009	9.42×10^{-10}	72.92
2	97.493	114.246	1.171	1.03×10^{17}	656,331	6.33×10^{-10}	195.90
3	95.361	111.141	1.165	8.13×10^{15}	950,989	1.18×10^{-8}	265.88
4	115.977	135.032	1.164	1.40×10^{16}	938,485	6.67×10^{-9}	322.72
5	111.533	129.280	1.159	3.44×10^{15}	1,234,534	3.58×10^{-8}	399.68

The time increase for these problems is self evident. Necessarily, only five replicates could be generated for instances of this size. The average solution time for these problems was 251 minutes which is significantly longer than the time required for problems of size size n_1=10. The time ranges from 72 minutes to 399 minutes indicating that there is substantial variability. However, it appears that these problems cannot be considered solveable in reasonable time. Once again, the screening method performed admirably. Only a minimal portion of the potential set space was examined; although the number of sets examined can be seen to be larger than that required for the smaller problems. The heuristic

overestimated the optimal solution by an average of 15% with very little variability from this amount.

The conclusions to be drawn from all of the two level testing with D_1=1000 is that problems of size n_1=8 and size n_1=10 can be solved to optimum in a reasonable length of time, while the time required to solve problems larger than this is excessive. The simple heuristics used as filters perform admirably and problems would be impossible to solve without their use. The solutions produced by the heuristics are surprisingly accurate given their simplicity. The major results of the two level testing are that the solution time depends upon the number of different products but does not significantly depend upon the range of the part requirements. In the next section, testing will be performed on four level problems with a wider variety of range levels to see if these conclusions can be extended to more general, multi-level problems.

VII.4 Four Level Experiment

The second major experiment involves fixing D_1=500 and setting the number of production levels at L=4. The value of 500 for the total production at level 1 is consistent with one shifts' production as described in section VIII.1. Extending the DP to four level problems allows for the testing of problems which more closely resemble realistic types of applications. With these parameters fixed, the levels of the size factor are set at n_1=8, n_1=10 and n_1=12. The levels of the range factor are set at each of the five levels:
$(20,20,20)$, $(20,40,60)$, $(20,100,400)$, $(20,400,1000)$ and $(20,1000,5000)$
where each triple represents the range values for the respective production levels (R_2,R_3,R_4). These range values are selected to reflect a broad spectrum of a

product's part requirements and to also test how well the DP handles diverse problems. The last triple may not be indicative of a realistic application as the ranges considered could be too variable for a JIT system. The reason for its inclusion is to provide a case which could push the DP to the limits of its capabilities. Fifteen replicates are generated for each treatment combination, thus the experiment is a complete 3x5 factorial experiment. The data are presented in tables VII.7-21.

Table VII.7							
n_1=8, R_2=20,R_3=20, R_4=20, D_1=500, L=4							
	Optimal Value	Heuristic Value	Heuristic Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	17.854	18.026	1.009	1.21×10^{12}	1,800	1.50×10^{-6}	1.75
2	17.163	17.553	1.022	6.92×10^{12}	3,068	4.44×10^{-9}	2.22
3	19.454	19.536	1.004	8.60×10^{12}	3,140	3.65×10^{-7}	2.42
4	18.041	18.404	1.020	1.68×10^{12}	3,132	3.13×10^{-7}	2.45
5	19.180	19.514	1.017	1.81×10^{11}	2,912	1.60×10^{-6}	2.62
6	18.204	18.602	1.021	8.39×10^{12}	4,012	4.83×10^{-7}	2.68
7	17.658	17.879	1.012	5.08×10^{13}	3,219	6.43×10^{-9}	2.98
8	17.502	17.929	1.024	1.27×10^{13}	3,668	2.88×10^{-7}	3.02
9	15.952	17.516	1.098	8.40×10^{13}	3,429	4.08×10^{-9}	3.10
10	17.881	19.250	1.076	2.04×10^{13}	4,318	2.15×10^{-8}	3.12
11	20.027	20.607	1.028	7.42×10^{10}	4,467	6.03×10^{-6}	3.35
12	19.371	20.039	1.034	3.80×10^{12}	4,899	1.28×10^{-6}	3.43
13	19.216	20.770	1.080	2.38×10^{12}	5,248	2.62×10^{-7}	4.38
14	19.279	20.266	1.051	5.14×10^{11}	5,245	1.02×10^{-6}	4.38
15	21.174	23.301	1.100	1.29×10^{12}	11,361	1.13×10^{-6}	8.77

Table VII.8

n_1=10,R_2=20,R_3=20, R_4=20, D_1=500, L=4

	Optimal Value	Heuristic Value	Heuristic / Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	18.966	19.152	1.009	1.51×10^{16}	4,012	2.67×10^{-11}	4.17
2	19.376	19.535	1.008	9.33×10^{15}	4,102	4.41×10^{-11}	5.32
3	20.656	21.135	1.023	8.82×10^{14}	6,539	7.41×10^{-10}	7.08
4	19.884	21.033	1.057	2.77×10^{15}	8,189	4.09×10^{-10}	7.30
5	19.613	20.992	1.070	8.51×10^{15}	7,542	9.42×10^{-11}	8.48
6	19.784	20.774	1.050	1.08×10^{15}	7,384	7.38×10^{-10}	8.63
7	21.416	22.386	1.045	2.11×10^{16}	7,958	3.78×10^{-8}	8.75
8	19.873	21.703	1.092	1.36×10^{14}	7,872	5.78×10^{-9}	9.05
9	19.681	21.395	1.087	1.08×10^{16}	9,092	9.09×10^{-11}	9.07
10	19.779	20.792	1.051	2.51×10^{15}	6,832	3.41×10^{-10}	9.87
11	19.826	21.499	1.084	7.77×10^{14}	9,004	1.15×10^{-9}	10.38
12	20.435	22.306	1.091	5.99×10^{14}	10,995	1.83×10^{-9}	10.72
13	21.070	21.995	1.043	1.68×10^{14}	10,172	6.05×10^{-9}	11.92
14	21.070	21.995	1.043	1.68×10^{14}	10,172	6.05×10^{-9}	12.23
15	20.395	22.555	1.105	1.02×10^{15}	12,682	1.26×10^{-9}	12.85

Table VII.9

n_1=12,R_2=20,R_3=20, R_4=20, D_1=500, L=4

	Optimal Value	Heuristic Value	Heuristic Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	20.679	21.345	1.032	3.41×10^{17}	9,844	2.88×10^{-12}	12.83
2	20.260	20.761	1.024	5.08×10^{17}	9,123	1.79×10^{-12}	15.12
3	20.721	21.624	1.043	1.56×10^{17}	11,397	7.30×10^{-12}	18.00
4	20.226	21.591	1.067	4.74×10^{17}	12,716	2.68×10^{-12}	18.30
5	20.809	22.510	1.081	4.36×10^{17}	14,851	3.40×10^{-12}	20.28
6	21.872	22.856	1.044	7.12×10^{16}	13,025	1.83×10^{-11}	20.55
7	22.071	23.455	1.062	2.75×10^{19}	17,067	6.32×10^{-11}	22.95
8	21.414	22.692	1.059	1.83×10^{17}	15,277	8.34×10^{-12}	23.53
9	20.979	22.305	1.063	1.70×10^{18}	15,811	1.58×10^{-12}	23.73
10	21.827	23.095	1.058	6.98×10^{17}	17,179	2.46×10^{-12}	27.12
11	21.696	23.262	1.072	2.15×10^{17}	23,053	1.07×10^{-11}	29.93
12	21.472	22.637	1.054	2.38×10^{17}	16,911	6.55×10^{-12}	31.75
13	22.196	24.172	1.089	3.22×10^{17}	23,744	7.37×10^{-12}	35.82
14	21.980	24.071	1.095	4.81×10^{17}	22,479	4.67×10^{-12}	38.08
15	22.283	25.486	1.143	1.33×10^{17}	52,343	3.93×10^{-11}	65.80

Table VII.10							
n_1=8,R_2=20,R_3=40, R_4=60, D_1=500, L=4							
	Optimal Value	Heuristic Value	Heuristic Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	45.868	46.33	1.010	1.63×10^{12}	1,683	1.05×10^{-7}	1.23
2	47.918	48.488	1.011	3.30×10^{12}	2,347	7.10×10^{-8}	1.80
3	49.296	50.708	1.028	1.85×10^{12}	2,083	1.15×10^{-6}	1.88
4	49.963	50.882	1.018	1.20×10^{13}	2,363	1.96×10^{-8}	1.90
5	52.718	54.025	1.024	5.04×10^{12}	2,366	4.73×10^{-8}	2.03
6	44.998	45.447	1.010	1.63×10^{12}	1,782	1.05×10^{-7}	2.03
7	51.638	54.318	1.051	5.07×10^{11}	3,731	7.35×10^{-7}	2.47
8	55.566	56.522	1.017	4.62×10^{11}	3,995	8.64×10^{-7}	2.65
9	52.263	55.979	1.071	7.25×10^{11}	3,597	4.96×10^{-7}	2.95
10	52.696	55.885	1.060	3.63×10^{12}	3,795	1.05×10^{-7}	3.37
11	53.652	58.542	1.092	9.81×10^{11}	5,208	5.30×10^{-7}	3.62
12	56.532	61.332	1.084	5.82×10^{11}	5,096	8.75×10^{-7}	4.20
13	58.901	62.965	1.069	2.37×10^{10}	8,265	3.59×10^{-5}	5.22
14	53.780	64.510	1.199	8.10×10^{11}	7,612	9.39×10^{-7}	5.85
15	56.814	60.450	1.064	9.70×10^{14}	7,165	7.38×10^{-10}	10.32

Table VII.11

n_1=10,R_2=20,R_3=40, R_4=60, D_1=500, L=4

	Optimal Value	Heuristic Value	Heuristic / Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	52.231	55.946	1.071	1.51×10^{16}	5,713	3.80×10^{-11}	5.33
2	57.729	59.972	1.038	8.82×10^{14}	6,468	7.33×10^{-10}	7.12
3	53.645	57.413	1.070	1.08×10^{15}	7,732	7.73×10^{-10}	9.65
4	57.991	60.523	1.043	2.77×10^{14}	11,094	4.00×10^{-9}	9.98
5	53.694	57.095	1.063	1.02×10^{15}	9,628	9.62×10^{-10}	10.07
6	54.642	58.256	1.066	7.77×10^{14}	9,017	1.16×10^{-9}	10.35
7	51.944	57.601	1.108	3.40×10^{15}	10,910	3.20×10^{-10}	11.08
8	56.881	61.315	1.078	6.11×10^{15}	10,550	1.72×10^{-10}	12.18
9	58.126	63.566	1.093	9.50×10^{13}	10,805	1.13×10^{-8}	12.40
10	57.849	63.313	1.094	1.36×10^{14}	11,142	8.19×10^{-9}	12.45
11	54.660	60.670	1.110	1.08×10^{16}	13,132	1.31×10^{-10}	12.65
12	56.226	60.967	1.084	1.10×10^{14}	14,583	1.32×10^{-8}	13.97
13	57.123	63.306	1.108	1.16×10^{15}	15,562	1.41×10^{-9}	15.25
14	59.422	64.834	1.091	1.68×10^{14}	13,933	8.29×10^{-9}	15.90
15	59.886	70.391	1.175	2.51×10^{15}	25,579	1.02×10^{-9}	32.60

Table VII.12							
n_1=12, R_2=20,R_3=40, R_4=60, D_1=500, L=4							
	Optimal Value	Heuristic Value	Heuristic Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	56.983	59.681	1.047	6.98×10^{17}	10,478	1.50×10^{-12}	16.55
2	57.638	61.737	1.071	1.83×10^{17}	12,150	6.63×10^{-12}	19.00
3	60.572	64.314	1.061	2.93×10^{16}	16,257	5.60×10^{-11}	23.85
4	58.605	62.820	1.071	4.70×10^{17}	14,895	3.16×10^{-12}	24.77
5	60.201	63.070	1.047	2.58×10^{17}	15,300	5.93×10^{-12}	25.37
6	58.588	62.568	1.067	1.33×10^{17}	14,986	1.12×10^{-11}	25.52
7	65.875	69.374	1.053	2.75×10^{16}	19,913	7.37×10^{-11}	25.90
8	57.118	62.695	1.097	3.24×10^{17}	27,718	8.55×10^{-12}	26.23
9	60.786	66.282	1.090	3.22×10^{17}	20,416	6.3×10^{-12}	30.33
10	59.004	63.327	1.073	2.39×10^{16}	25,121	1.05×10^{-10}	39.13
11	63.790	69.363	1.087	8.86×10^{16}	33,815	3.81×10^{-11}	41.33
12	59.659	66.347	1.112	2.02×10^{17}	31,341	1.55×10^{-11}	46.72
13	61.963	68.831	1.110	8.21×10^{16}	37,095	4.52×10^{-11}	50.07
14	62.917	68.896	1.095	1.00×10^{16}	44,930	4.49×10^{-10}	53.15
15	57.307	63.141	1.101	8.03×10^{16}	32,461	4.04×10^{-11}	53.90

Table VII.13

n_1=8, R_2=20,R_3=100, R_4=400, D_1=500, L=4

	Optimal Value	Heuristic Value	Heuristic Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	340.595	342.363	1.005	3.46×10^{11}	2,844	8.21×10^{-7}	1.93
2	358.204	365.062	1.019	1.90×10^{12}	3,189	1.67×10^{-7}	2.33
3	354.944	358.271	1.011	8.73×10^{10}	3,321	3.81×10^{-8}	2.42
4	342.831	365.337	1.067	1.15×10^{13}	3,835	3.48×10^{-8}	2.98
5	339.342	363.341	1.070	1.68×10^{13}	4,541	2.70×10^{-8}	3.00
6	352.147	366.484	1.039	3.64×10^{13}	3,815	1.05×10^{-8}	3.28
7	332.901	354.557	1.066	6.55×10^{12}	4,048	6.22×10^{-8}	3.32
8	369.760	382.526	1.035	1.43×10^{12}	4,236	3.02×10^{-7}	3.47
9	352.157	391.969	1.110	1.63×10^{13}	5,904	3.62×10^{-8}	3.53
10	382.405	402.637	1.052	1.36×10^{11}	8,599	6.32×10^{-6}	4.35
11	342.202	368.522	1.076	5.03×10^{12}	5,005	1.00×10^{-7}	4.42
12	382.405	410.943	1.073	1.36×10^{11}	8,599	6.32×10^{-6}	4.85
13	334.264	367.708	1.098	6.69×10^{13}	6,299	9.41×10^{-9}	4.95
14	337.672	387.682	1.148	2.56×10^{13}	7,836	3.06×10^{-8}	5.26
15	366.619	419.156	1.144	2.79×10^{12}	11,763	4.35×10^{-7}	7.73

Table VII.14

n_1=10, R_2=20,R_3=100, R_4=400, D_1=500, L=4

	Optimal Value	Heuristic Value	Heuristic / Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	345.918	348.623	1.008	2.19×10^{15}	4,549	2.16×10^{-10}	5.32
2	351.650	375.714	1.068	7.73×10^{15}	5,846	7.59×10^{-11}	6.15
3	343.806	361.780	1.052	1.00×10^{16}	5,705	5.70×10^{-11}	6.35
4	390.717	407.720	1.043	6.22×10^{14}	6,258	9.45×10^{-10}	7.13
5	356.968	391.466	1.098	1.24×10^{16}	7,071	5.89×10^{-11}	8.15
6	371.525	394.763	1.061	2.77×10^{15}	9,757	3.61×10^{-10}	8.70
7	349.240	379.053	1.085	1.02×10^{15}	9,512	9.51×10^{-10}	9.68
8	363.952	390.531	1.074	7.30×10^{15}	9,328	1.27×10^{-10}	10.08
9	364.718	387.191	1.063	7.77×10^{14}	9,014	1.16×10^{-9}	10.33
10	363.241	393.710	1.082	3.18×10^{14}	11,430	3.59×10^{-9}	10.37
11	350.339	383.892	1.094	2.72×10^{15}	10,394	3.84×10^{-10}	12.17
12	401.693	428.906	1.067	1.36×10^{14}	12,393	9.11×10^{-9}	14.02
13	397.404	430.304	1.083	1.68×10^{14}	13,385	7.96×10^{-9}	15.80
14	370.005	416.947	1.124	6.76×10^{15}	15,841	2.36×10^{-10}	16.42
15	408.569	453.215	1.110	2.97×10^{14}	22,898	7.70×10^{-9}	24.62

Table VII.15

n_1=12 R_2=20,R_3=100, R_4=400, D_1=500, L=4

	Optimal Value	Heuristic Value	Heuristic / Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	380.479	385.081	1.013	5.86×10^{16}	5,414	9.33×10^{-12}	8.38
2	361.404	371.476	1.027	1.48×10^{17}	5,958	4.02×10^{-12}	9.73
3	388.622	413.709	1.064	2.33×10^{17}	15,742	6.75×10^{-14}	19.72
4	372.866	411.075	1.104	3.51×10^{17}	16,388	4.66×10^{-12}	23.45
5	366.272	387.706	1.057	1.07×10^{17}	14,618	1.36×10^{-11}	23.75
6	398.550	422.747	1.060	4.50×10^{15}	18,714	4.15×10^{-10}	27.30
7	377.894	410.272	1.087	1.25×10^{17}	20,535	1.64×10^{-11}	28.03
8	382.176	403.249	1.054	2.39×10^{17}	16,438	6.87×10^{-12}	28.12
9	426 555	449.254	1.053	3.00×10^{16}	19,624	6.54×10^{-11}	28.90
10	391.421	435.253	1.112	1.04×10^{18}	23,501	2.35×10^{-12}	34.03
11	416.790	449.764	1.079	1.06×10^{17}	38,920	3.67×10^{-11}	49.93
12	433.114	471.666	1.089	9.67×10^{16}	36,815	3.83×10^{-11}	57.43
13	424.071	464.153	1.094	1.01×10^{16}	41,932	4.19×10^{-10}	61.53
14	424.573	456.780	1.075	4.36×10^{17}	36,620	8.39×10^{-12}	63.08
15	435.721	486.798	1.117	4.48×10^{16}	70,287	1.59×10^{-10}	110.97

Table VII.16

n_1=8, R_2=20,R_3=400, R_4=1000, D_1=500, L=4

	Optimal Value	Heuristic Value	Heuristic Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	757.517	813.454	1.073	5.08×10^{13}	3,265	6.42×10^{-9}	2.28
2	894.238	912.694	1.020	1.90×10^{12}	3,210	1.68×10^{-7}	2.33
3	869.138	927.021	1.066	4.18×10^{12}	4,104	9.81×10^{-8}	2.73
4	881.313	900.576	1.021	6.45×10^{12}	3,392	5.25×10^{-8}	2.82
5	799.355	860.278	1.076	7.92×10^{12}	3,614	4.56×10^{-8}	3.08
6	853.445	932.493	1.092	3.92×10^{12}	5,146	1.31×10^{-7}	3.28
7	889.558	938.465	1.055	8.61×10^{15}	4,806	5.58×10^{-8}	3.35
8	847.867	900.555	1.062	1.99×10^{13}	4,785	2.40×10^{-8}	3.62
9	894.169	944.824	1.055	2.23×10^{12}	5,352	2.40×10^{-7}	3.95
10	916.880	975.076	1.064	6.22×10^{9}	6,826	8.30×10^{-5}	4.45
11	888.548	996.951	1.122	3.32×10^{12}	5,742	1.72×10^{-7}	4.60
12	880.476	988.274	1.122	3.32×10^{13}	5,636	1.69×10^{-8}	4.72
13	851.167	948.202	1.060	1.87×10^{13}	6,912	3.69×10^{-8}	4.78
14	943.215	1029.675	1.091	7.11×10^{11}	7,966	1.12×10^{-6}	6.17
15	941.269	1047.905	1.112	3.16×10^{11}	9,705	3.07×10^{-6}	6.26

Table VII.17

n_1=10, R_2=20,R_3=400, R_4=1000, D_1=500, L=4

	Optimal Value	Heuristic Value	Heuristic / Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	896.946	908.079	1.013	6.35×10^{14}	4,740	7.46×10^{-10}	4.73
2	822.530	854.813	1.038	6.41×10^{15}	4,473	6.97×10^{-11}	4.75
3	898.359	900.507	1.002	2.78×10^{15}	4,390	1.57×10^{-10}	5.13
4	945.921	949.617	1.004	2.47×10^{15}	4,165	1.68×10^{-10}	5.32
5	936.463	946.596	1.010	7.19×10^{15}	6,164	8.57×10^{-11}	7.12
6	931.178	979.825	1.051	9.58×10^{15}	6,413	6.69×10^{-11}	7.17
7	903.180	926.745	1.025	3.92×10^{15}	7,500	1.91×10^{-10}	7.97
8	913.791	963.598	1.054	5.90×10^{14}	6,602	1.11×10^{-9}	8.00
9	883.799	945.120	1.070	2.29×10^{16}	7,503	3.27×10^{-11}	9.22
10	943.640	1014.383	1.075	1.53×10^{15}	9,796	6.40×10^{-10}	10.75
11	961.057	1037.344	1.079	1.13×10^{15}	10,292	9.10×10^{-10}	11.37
12	896.946	908.079	1.013	6.35×10^{15}	4,740	7.46×10^{-8}	11.80
13	956.857	1047.080	1.095	1.83×10^{16}	13,990	7.64×10^{-11}	14.78
14	956.191	1040.282	1.087	7.75×10^{14}	17,838	2.30×10^{-9}	17.22
15	966.123	1106.170	1.144	1.16×10^{15}	20,880	1.80×10^{-9}	22.67

Table VII.18

n_1=12, R_2=20,R_3=400, R_4=1000, D_1=500, L=4

	Optimal Value	Heuristic Value	Heuristic Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	977.466	993.586	1.016	3.82×10^{17}	7,342	1.92×10^{-12}	11.13
2	1039.030	1089.175	1.048	2.98×10^{16}	13,695	4.59×10^{-11}	19.43
3	1005.106	1043.019	1.037	9.33×10^{14}	18,331	1.96×10^{-9}	20.50
4	945.463	993.355	1.050	6.08×10^{17}	15,657	2.57×10^{-12}	23.77
5	964.285	1033.859	1.071	5.68×10^{17}	17,113	3.01×10^{-12}	25.30
6	980.974	1043.364	1.064	4.70×10^{17}	14,881	3.16×10^{-12}	25.45
7	1033.540	1063.345	1.029	2.03×10^{17}	17,597	8.66×10^{-12}	28.08
8	1020.890	1077.762	1.055	1.26×10^{18}	16,848	1.33×10^{-12}	28.26
9	1049.007	1132.317	1.079	3.11×10^{15}	21,105	6.78×10^{-10}	28.28
10	970.487	1053.520	1.085	4.12×10^{17}	19,753	4.79×10^{-12}	31.15
11	982.988	1073.760	1.092	1.72×10^{18}	23,286	1.35×10^{-12}	36.28
12	979.981	1070.139	1.092	1.72×10^{18}	22,269	1.29×10^{-12}	37.23
13	953.358	1070.431	1.122	5.21×10^{16}	30,395	5.83×10^{-11}	42.03
14	1054.642	1164.558	1.104	6.32×10^{16}	31,219	4.93×10^{-11}	47.58
15	1003.196	1135.211	1.131	2.31×10^{17}	41,393	1.79×10^{-11}	61.25

Table VII.19

n_1=8, R_2=100, R_3=1000, R_4=5000, D_1=500, L=4

	Optimal Value	Heuristic Value	Heuristic / Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	4089.980	4107.329	1.004	1.17×10^{13}	2,983	2.54×10^{-8}	2.05
2	4144.802	4167.649	1.005	1.66×10^{13}	2,718	1.63×10^{-8}	2.26
3	4592.304	4697.398	1.022	2.33×10^{11}	4,463	1.91×10^{-6}	2.75
4	4223.465	4441.673	1.051	3.33×10^{13}	3,237	9.72×10^{-9}	2.78
5	4158.100	4353.860	1.046	2.08×10^{12}	3,517	1.69×10^{-7}	2.95
6	4129.209	4226.940	1.023	1.69×10^{13}	3,808	2.25×10^{-8}	2.97
7	4319.537	4504.637	1.042	8.47×10^{12}	3,920	4.62×10^{-8}	3.12
8	4472.685	4709.047	1.052	7.43×10^{11}	4,405	5.92×10^{-7}	3.23
9	3942.954	4392.819	1.114	5.59×10^{12}	5,076	9.08×10^{-8}	3.57
10	4376.819	4702.708	1.074	1.03×10^{12}	5,316	5.16×10^{-7}	3.83
11	4560.929	5021.258	1.101	1.42×10^{13}	5,777	4.06×10^{-8}	4.26
12	4566.122	5005.534	1.096	2.73×10^{12}	6,231	2.28×10^{-7}	5.03
13	4594.717	5092.636	1.108	1.06×10^{12}	6,858	6.46×10^{-7}	5.12
14	5107.819	5450.300	1.067	1.85×10^{11}	8,843	4.78×10^{-6}	5.72
15	4870.914	5151.141	1.057	7.11×10^{14}	7,981	1.12×10^{-6}	6.38

Table VII.20							
n_1=10, R_2=100, R_3=1000, R_4=5000, D_1=500, L=4							
	Optimal Value	Heuristic Value	Heuristic Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	4129.074	4331.716	1.048	4.12×10^{15}	4,704	1.14×10^{-10}	4.47
2	4424.114	4657.449	1.052	4.57×10^{15}	7,432	1.62×10^{-10}	7.40
3	4638.519	4794.124	1.033	4.20×10^{14}	7,652	1.82×10^{-9}	7.42
4	4442.643	4728.468	1.064	1.31×10^{15}	7,879	6.01×10^{-10}	7.88
5	4779.744	4930.490	1.031	1.91×10^{15}	8,770	4.59×10^{-10}	9.92
6	4758.377	5066.601	1.064	2.08×10^{15}	9,912	4.76×10^{-10}	10.22
7	4839.948	5336.490	1.102	1.36×10^{14}	10,798	7.93×10^{-9}	11.08
8	4766.963	4981.337	1.045	3.98×10^{14}	12,334	3.09×10^{-9}	11.57
9	4621.322	5261.987	1.138	1.50×10^{15}	13,921	9.28×10^{-10}	13.60
10	4778.297	5081.502	1.063	1.07×10^{14}	14,947	1.39×10^{-8}	15.08
11	4799.155	5423.897	1.130	4.02×10^{14}	15,973	3.97×10^{-9}	15.35
12	4745.410	5308.691	1.118	3.07×10^{14}	16,293	5.30×10^{-9}	15.40
13	4932.025	5482.516	1.111	7.07×10^{14}	20,298	2.87×10^{-9}	20.97
14	4915.247	5590.467	1.137	1.65×10^{14}	21,277	1.28×10^{-8}	23.48
15	4998.003	5537.492	1.107	6.54×10^{13}	32,050	4.90×10^{-8}	32.60

Table VII.21							
n_1=12, R_2=100, R_3=1000, R_4=5000, D_1=500, L=4							
	Optimal Value	Heuristic Value	Heuristic Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	5432.369	5516.605	1.015	2.74×10^{14}	11,738	4.28×10^{-9}	14.92
2	4764.732	4852.381	1.018	1.09×10^{17}	10,678	9.79×10^{-12}	15.25
3	4984.662	5342.729	1.071	5.54×10^{16}	17,390	3.13×10^{-11}	23.83
4	4981.770	5301.261	1.064	1.56×10^{17}	19,665	1.26×10^{-11}	25.13
5	5067.591	5329.898	1.051	7.10×10^{16}	16,704	2.35×10^{-11}	25.25
6	4970.303	5279.220	1.062	2.42×10^{17}	17,572	7.26×10^{-12}	26.67
7	5103.539	5390.946	1.056	1.26×10^{18}	17,091	1.35×10^{-12}	28.45
8	5095.692	5378.182	1.055	3.33×10^{16}	19,724	5.92×10^{-11}	30.03
9	5119.308	5483.806	1.071	5.60×10^{16}	18,711	3.34×10^{-11}	31.67
10	4949.840	5375.874	1.086	8.58×10^{15}	29,397	3.42×10^{-10}	37.17
11	5337.908	5806.580	1.087	2.66×10^{16}	29,963	1.12×10^{-10}	39.77
12	4940.052	5374.705	1.087	4.72×10^{16}	35,078	7.43×10^{-11}	48.52
13	4861.573	5374.531	1.105	3.67×10^{17}	41,524	1.13×10^{-11}	58.43
14	5227.400	5823.801	1.114	1.57×10^{16}	47,250	3.00×10^{-10}	61.03
15	5368.574	6193.022	1.153	2.00×10^{17}	69,981	3.49×10^{-11}	120.60

VII.4 .1 Analysis

Because of the 3x5 factorial design, ANOVA's of the output can, once again, be constructed for analyzing both the time taken to determine optimal solutions and for evaluating the accuracy of the heuristics used for the filters. The ANOVA for the time taken is;

Source	SS	d.f.	MS	F_0
Range	929.85	4	232.46	1.70
Size	36,766.68	2	18,383.34	134.59
Interaction	854.11	8	106.76	.781
Error	28,684.85	210	136.59	
Total	67,235	224		
$F_{.10,4,210}=1.94$		$F_{.01,2,210}=3.32$		$F_{.25,8,210}=1.28$

This ANOVA indicates that the time taken to determine optimal solutions does not significantly depend upon the range of the part requirements. However, the solution time significantly depends upon the size of the problem. No significant interaction effect can be detected between the two factors.

Thus, the outcome detected in the two level experiment recurs in this four level experiment. As this second experiment is significantly larger than the first (considering both more size and range levels and examining 225 replicates as opposed to 60), this conclusion not only validates the conclusion of the first experiment, but considerably strengthens the initial result. The range of the part requirements plays no significant role in the solution time performance of the DP. Solution time depends significantly upon the number of different products.

To eliminate any concern as to the validity of these conclusions, it is once again imperative to analyze the results using a test which makes no assumptions regarding the underlying distributions of any of the data. It might be that the requisite time variance equality assumption is violated. Therefore, a nonparametric test, not susceptible to any variance bias, could serve as a better indicator. A Kruskal-Wallis test can be used for this purpose. Kruskal-Wallis tests are used to compare differences between several random samples from independent populations and require no assumptions regarding the normality of the data or of the equality of their variances (for a description of this test, see Conover (1980)). These tests are similar in operation to the previously introduced Mann-Whitney test, except the calculated test statistic is denoted by H. In order to use this test, several comparisons must be made. Each comparative test involves fixing a level for one factor and comparing the data from all of the levels of the second factor; thus requiring that eight such comparisons be made. Comparative tests can also be performed on pooled levels of the factors. The results of each of these comparisons are summarized below.

1. Fixing the range at $(20,20,20)$ and testing to see if the times differ at this range level depending upon the value of n_1 produces H=37.45. At a level of significance less than 0.01%, the conclusion can be drawn that the times taken for n_1=8, n_1=10 and n_1=12 at this level of the range values are different.
2. Fixing the range at $(20,40,60)$ and testing to see if the times differ at this range level depending upon the value of n_1 produces H=36.57. At a level of significance less than 0.01%, the conclusion can be drawn

that the times taken for n_1=8, n_1=10 and n_1=12 at this level of the range values are different.

3. Fixing the range at $(20,100,400)$ and testing to see if the times differ at this range level depending upon the value of n_1 produces H=35.07. At a level of significance less than 0.01%, the conclusion can be drawn that the times taken for n_1=8, n_1=10 and n_1=12 at this level of the range values are different.
4. Fixing the range at $(20,400,1000)$ and testing to see if the times differ at this range level depending upon the value of n_1 produces H=36.23. At a level of significance less than 0.01%, the conclusion can be drawn that the times taken for n_1=8, n_1=10 and n_1=12 at this level of the range values are different.
5. Fixing the range at $(20,1000,5000)$ and testing to see if the times differ at this range level depending upon the value of n_1 produces H=35.51. At a level of significance less than 0.01%, the conclusion can be drawn that the times taken for n_1=8, n_1=10 and n_1=12 at this level of the range values are different.
6. Pooling all of the times from each range for each level of n_1 and testing to see if the times differ depending upon the value of n_1 produces H=182.12. At a level of significance less than 0.01%, the conclusion can be drawn that the times taken for n_1=8, n_1=10 and n_1=12 over all levels of the combined range values are different.
7. Fixing the size level at n_1=8 and testing to see if the times differ at this level depending upon the level of the range produces H=5.36. This H is significant only at the 25.4% level and the conclusion can be drawn

that the solution times required at each range level do not differ significantly for this value of n_1.

8. Fixing the size level at n_1=10 and testing to see if the times differ at this level depending upon the level of the range produces H=7.79. This H is significant only at the 10.1% level and the conclusion can be drawn that the solution times required at each range level do not differ significantly for this value of n_1.
9. Fixing the size level at n_1=12 and testing to see if the times differ at this level depending upon the level of the range produces H=4.18. This H is significant only at the 38.3% level and the conclusion can be drawn that the solution times required at each range level do not differ significantly for this value of n_1.
10. Pooling all of the times from each size level for each level of the range and testing to see if the times differ depending upon the level of the range produces H=1.56. This is significant only at the 81.6% level, therefore the conclusion can be drawn that the solution times of all the range levels do not significantly differ for the pooled levels of the size.

The conclusions from this series of Kruskal-Wallis tests can be summarized in the following manner. At no fixed level of the size, n_1, do the solution times differ significantly between any of the range levels. For each fixed level of the range, the solution times between the size levels differed significantly. Pooling the data over the various factor levels corroborates these results for the each of the two factors. Therefore, the conclusions can be made that the solution time does not significantly depend upon the range of the part requirements, but that the solution times significantly depend upon the number of different products.

This conclusion substantiates completely the conclusions obtained from the two level experiment. Once again, it has been demonstrated that solution time is not significantly affected by the range levels of the part requirements.

The average solution times for each treatment combination are presented below. It can be easily seen that the times differ substantially between the size levels, but that a comparison of the times between the range levels does not indicate distinct differences.

	n_1=8	n_1=10	n_1=12	
(20,20,20)	3.3 min	9.0 min	26.9 min	13.1 min
(20,40,60)	3.4 min	12.7 min	33.4 min	16.5 min
(20,100,400)	3.8 min	11.0 min	38.2 min	17.7 min
(20,400,1000)	3.8 min	9.8 min	31.0 min	14.9 min
(20,1000,5000)	3.7 min	13.7 min	39.1 min	18.8 min
	3.6 min	11.2 min	33.7 min	16.2 min

It is also apparent from the experimental data that these solution times are not of an unmanageable length for practical implementation. The increase in solution time between the size levels appears to indicate that the growth rate is nonlinear in the size level and that if the times increased in this same manner for problems of a size larger than those considered here, then the solution times

could become too large for practical implementation. This would suggest a need for further testing with regard to the size factor.

The second evaluation criterion is, again, the DP's storage requirement. Limitations to the storage capability are directly correlated to the success of the heuristic in its screening capacity. For the heuristic to be useful, the accuracy of the solutions produced by it must not be dependent upon the different levels of each factor. Thus, a test as to this variability is necessary. An ANOVA for the percentage overestimation of the optimal solution by the heuristic solution appears below.

Source	SS	d.f.	MS	F_0
Range	107.00	4	26.75	2.20
Size	72.40	2	36.20	2.97
Interaction	151.88	8	18.98	1.56
Error	2551.77	210	12.15	
Total	2883.05	224		
$F_{.10,4,210}=1.94$		$F_{.10,2,210}=2.30$		$F_{.25,8,210}=1.28$

This ANOVA table indicates that the heuristic performed equally accurately over all levels of both the size and the range of the problems. No significant differences between the overestimation of the optimal solution by the heuristic could be uncovered for any level of either factor. Furthermore, no interaction between the factors could be detected. The result confirms that the heuristics can be used with a confidence that their accuracy depends upon neither the size nor the range of the problem. As the heuristics are necessary for

screening suboptimal sets, this confidence with respect to their accuracy, regardless of the size or the range of the problems for which they are used, is essential.

In the worst instance, the maximum percentage of the total possible number of feasible sets generated in any of the replicates over all levels of both factors was 8.30×10^{-5}. This implies that the minimum percentage of screened sets eliminated for any problem was 99.9917%. In the worst case, the maximum number of sets which had to be stored in-core for any of the problems never exceeded 600. Therefore, in no case did storage space constrain the functioning of the DP. The screening provided by the heuristics can be considered an essential and integral part of the DP's operation. This filtering mechanism pacifies prior concerns associated with the growth rate of the state space, which had previously been identified as a potential handicap encountered by these types of algorithms, for all of the problems generated. The concurrent implementation of the heuristics with the DP effectively eliminates the storage concerns for problems which can be optimally solved by the DP in reasonable time.

VII.4.2 Supplementary Testing of Four Level Problems

The results from the 3x5 factorial experiment, since they were collected from a wide variety of problems, provide a very good indication of how the DP performs on four level problems. Some supplementary testing would be of interest to judge how the solution time changes with the problem size and what sort of limitation the size has on a problem's solvability. The preceding analysis demonstrated that range is not an important consideration in determining the solution time for problems of any size. Hence, fixing the range settings arbitrarily

at the levels of R_2=20, R_3=40 and R_4=60 should not significantly affect the measurement of the solution time of additional problems. Table VII.22 provides an indication as to how solution time increases if the size factor is set at n_1=16.

Table VII.22

n_1=16,R_2=20,R_3=40, R_4=60, D_1=400, L=4

	Optimal Value	Heuristic Value	Heuristic Optimal	Total Sets Possible	Sets Examined	% of Sets Examined	Time (min.)
1	69.584	73.344	1.05	8.94×10^{16}	59,298	6.63×10^{-11}	138.39
2	66.252	69.699	1.04	3.35×10^{17}	69,615	2.07×10^{-11}	169.43
3	69.878	74.311	1.07	7.14×10^{15}	89,936	1.25×10^{-9}	197.63
4	73.386	78.091	1.06	1.48×10^{16}	106,235	7.17×10^{-10}	221.60
5	73.931	78.816	1.06	2.51×10^{15}	123,180	4.90×10^{-9}	243.85
6	76.136	79.857	1.03	3.13×10^{16}	146,534	4.68×10^{-10}	342.38

The data clearly indicate that for n_1=16 the solution times have become sufficiently large as to prohibit the use of the DP for practicaly scheduling problems with this number of different products. The average solution time of 218 minutes must be considered too long for realistically scheduling an 8-hour production shift. Due to the length of time required to solve these problems, only six replicates were generated. The heuristics work with a remarkable degree of accuracy on these problems, indicating an average overestimation of only 4.8%. Their filtering capabilities are again self-evident, as the storage capacity constraint is never restrictive.

VII.5 Discussion of Experimental Results

After examining more than 325 randomly generated problems, several conclusions can be drawn with respect to the performance of the DP algorithm. Primarily, that the solution time is dependent upon the number of different products, but does not depend significantly upon the range of the part requirements that make up each of these products. The conclusion dealing with the size of the problem is not unexpected. However, the conclusion regarding the range of the part requirements was not foreseen.

The ramifications of the part requirements finding are substantial; particularly with respect to the potential introduction of weighting factors to the minimax problem. Recall that the gamma matrix used by the DP has rows which correspond to each individual part and that each matrix element, γ_{ijl}, within each row represents a measure of the weighted deviations in usage of part i in level j from the proportional usage per unit of product l. The introduction of weighting factors has the effect of altering the weightings of the individual matrix elements, thereby creating a similar effect to that of altering the range of the part requirements. But if the solution time is essentially independent of this range, then the addition of these weighting factors would provide no time benefit (or time changes) over the solution time of an instance of the unweighted case. In fact, based upon the range conclusion, weighting the various levels would have no significant effect on the solution time. Hence, the time results found by experimentation on the unweighted problems could just as easily be applied to those times that would be obtained from testing weighted problems. The only change caused by weighting would be to produce a different sequence of final assembly. Therefore, experimentation on the DP for the time and space

constraints for problems involving weighting factors need not be considered imperative.

Experimentation on the DP was also performed to determine what sized problems could be optimally solved in a reasonable period of time (on the VAX). Results of this testing on two-level problems demonstrated that, for D_1=1000, optimal solutions could be determined in reasonable time for those problems of size n_1=8 and n_1=10. However, when n_1=12, the problems no longer can be solved in an amount of time which could be useful for practical purposes. Extensive additional testing on problems with n_1=10 demonstrated that, while the solution times remained similar to those encountered in the initial experiment and that most of the problems of this size could be solved in reasonable time, the possibility arose that a few problems could potentially require an excessive amount of time. This was indicated by one outlier problem which took more than 5 hours to solve. Other than this outlier, the vast majority of these problems required less than one hour of computational time.

Testing on a wide variety of four-level problems with D_1=500 revealed that problems of size n_1=8, n_1=10 and n_1=12 could all be solved to optimum in reasonable time. When the problem size was increased to n_1=16, the solution time could no longer be considered practical. Examining how the solution times increased in reponse to an increase in n_1 demonstrated that the time appeared to increase at a nonlinear rate with n_1.

One conclusion from the testing of the DP is that the algorithm is constrained by the amount of solution time. That is, not all problems of practical interest can be solved in a reasonable length of time. A faster computer would increase the size of the problem which could be optimally solved. However, due

to the nature of the rate of increase in computation time, those problems which could be solved on a faster machine would also become constrained for some value of n_1.

The issue of whether the storage capacity constraint could be considered binding was addressed by the concurrent implementation of two simple heuristics with the DP. These heuristics significantly reduced the state space of the DP by eliminating more than 99.99% of the potential sets. Due to the magnitude of the state space, even this reduction in the search effort could not ensure a solution in reasonable period of time. However, no solutions could possibly have been determined without the use of the heuristic filter. The heuristics demonstrated that even very simple dominance schemes can be used as highly effective screening devices. The simple heuristics performed with surprising accuracy, with the largest overestimation of the optimal solution value being only 30%. Testing also revealed that the heuristics were equally accurate over all levels of the size and range of the problems examined. The net effect of the heuristics' use is that the space requirements of the DP never became a constraining factor.

VII.6 Summary

This chapter has presented the findings of the experimentation of the DP algorithm. The aim of this testing was to hopefully uncover useful information regarding its performance capabilities, characteristics and limitations. The concept of time reasonableness was introduced as a requirement for the DP's success. Reasonableness of time, for purposes considered here, implied that a solution could be found by the DP in an amount of time sufficient to allow for the output to be used in the scheduling of 500 products in one 8-hour production

shift. The conclusions from this testing can be summarized in the following way. Solution time was a tight constraining factor for the size of the DP: certain problems could be solved in reasonable time while others could not. The simple heuristic screening method, employed concurrently with the DP, worked highly efficiently and permitted the conclusion to be drawn that, for those problems solvable in reasonable time, storage capacity never constrained the size of the problem.

Experiments with two-level problems indicated that those problems with D_1=1000 and n_1=10 could be solved to optimum in reasonable time, while those problems with D_1=1000 and n_1=12 could not. The experiments with four-level problems indicated that problems with D_1=500 and n_1=12 could be optimally solved in reasonable time, while those with D_1=500 and n_1=16 could not. The rate of production for the two-level problems is 125 products/hour, while the rate for the four-level problems is 62.5 products/hour. The total amount of production and the indicated rates of this production could be considered as sufficiently large enough for implementation in certain realistic applications.

Two conclusions may be drawn from both the two-level and the four-level experiments. Firstly, the solution time depends significantly on the number of different products. This result can not be considered as a surprise. The second conclusion, which was not foreseen, states that the solution time does not significantly depend upon the range of the part requirements. This result has repercussions with respect tc the use of weighting factors with the DP; namely, solution times for the weighted DP will behave in exactly the same fashion as those of an unweighted DP of the same size. Hence, extensive testing to

determine the solution times and size limitations of the weighted DP is not necessary.

CHAPTER 8 SUMMARY

VIII.1 Thesis Summary

This thesis has examined the usage problem which occurs in the scheduling of mixed-model assembly processes operating under JIT methods. A minimax objective function has been introduced to control this process. This minimax model has not been considered previously in the literature for use with mixed-model, JIT systems. A general, integer programming model of the problem was developed and the unweighted versions of this problem were of primary interest. The goal of the thesis was to determine optimal sequencing methods for various formulations of this model. Certain analogous properties of the minimax problem to previously existing 'sum of deviations' models were shown.

The single-level, unweighted version of the general model was considered in some detail. It was shown that optimizing this special case could be achieved by solving a series of decision problems; where each decision involved a test to determine if a feasible sequence could be constructed for a hypothesized target value of the objective. This target restricted where, in a sequence, each copy of a product could appear if the target was to remain feasible. The decision problem is equivalent to a single-machine, unit time scheduling problem in which each job has a release time and due date. The scheduling problem can be modelled as a bipartite, convex graph in which the

feasibility of the scheduling problem is equivalent to being able to determine a perfect matching in this graph. The feasibility of this matching could be determined in time which is linear in the number of vertices and, hence, in the total product demand.

Using the graph theoretic representation allowed for the calculation of bounds on the target value. Both upper and lower bounds were shown for the problem. Of particular significance is the upper bound, which demonstrates that a feasible schedule always exists for each copy of every product in which the actual level of production never deviates from the ideal level by more than 1 unit.

Symmetries within the problem were shown to exist which could be exploited so as to substantially reduce the computational effort. Of particular significance was the proof that if a common factor, β, existed for the demand for each product, then optimal solutions constructed by the EDD algorithm would always consist of β repetitions of the optimal sequence for the reduced problem in which the problem demands consisted of the initial demands with the common divisor factored out. This factoring property alone permits substantial reductions in the computational effort.

It was demonstrated that the stopping criterion for the EDD algorithm could be modified so that the optimal solution could be found in time polynomial in D. This was achieved by proving that only a finite number of calls to the perfect matching routine would be required in order to determine the optimal objective value. Extensions of the unweighted algorithm to the weighted version of the problem were made. Thus, weighted, single-level problems and "pegged" multi-level problems could also be solved to optimum using the EDD

algorithm. Many of the computational efficiencies developed for the unweighted problems could also be applied to the weighted versions of the problems.

A DP algorithm was presented for optimizing the general minimax model and therefore for solving multi-level problem formulations. The time and space requirements of this DP were demonstrated. The computational requirements of the DP are considerably lower than those that would be experienced in any explicit enumeration procedure. The DP could also be modified to solve instances of the multi-level, sum functions. This is important to demonstrate since no previous solution procedures have appeared in the literature for optimally solving these problems.

Tests of the DP's performance capabilities, characteristics and limitations were performed. As problems could always be constructed which would require excessive periods of solution time, the arbitrary concept of time "reasonableness" needed to be considered. Of interest was the nature of the problems which could be solved in this reasonable amount of time. If problems of all practical interest could be optimally solved in time sufficient for realistic applications, then the DP could be used for any such implementations. If, however, reasonable time constrained the size of the problem which could be solved, then it was necessary to know what sized problems were so constrained. Also, it was necessary to ascertain whether storage capacity constrained the size of the problems which could be optimized in this reasonable solution time. As the growth of the DP's state space could severely restrict the problem size if all the states were generated, a necessary screening method, employing simple heuristics, was required. The results of the testing indicated that the size of the problems which could be optimally solved was,

indeed, constrained by reasonable solution time. Conversely, the size of the problems generated never became constrained by the storage requirements. In fact, the simple heuristics acted as a highly efficient screening mechanism.

The experimentation performed on the DP uncovered certain, inherent solution characteristics. Not surprisingly, the size or number of different products in the problems significantly affects the amount of solution time required. Problems with more product types require significantly more solution time, than problems with fewer product types. The experimental results indicated that the range of the products' part requirements had no significant effect on the solution time. This result implies that weighting the parts at the various production levels will not alter the solution time required by an unweighted problem of the same size, as weighting would create a similar effect to that of altering the range of the part requirements in an unweighted problem.

The use of weighting factors introduces an element of arbitrariness and a certain degree of bias to the problems. As weighting factors would only have the effect of altering the product assembly sequence and not the solution time for multi-level problems, a case could be presented whereby if weighting factors are necessary, then they should be used with the weighted, single-level algorithm. There are two main reasons for the implementation of weighting factors; either to place more emphasis on the scheduling of particular products, or to make the deviations at each level comparable in magnitude. The usage of all parts and products has been shown to depend explicitly on the sequence of the final assembly of the products. Because of the pull nature of JIT manufacturing, the focus of control is necessarily the highest production level. Smoothing part usage involves separating the products with the highest part requirements in the

assembly sequence. An algorithm for smoothing part usage would try to avoid the consecutive sequencing of two products with high part requirements.

If a weighted objective is to be used, then some method must exist for the calculation of the weights for all parts and products. Calculating weights for products at the highest level can involve no more arbitrariness than would be involved in the calculation of the weights for the parts at the lower levels. If weights could be calculated for the parts, they could just as easily be determined for the products in such a way as to reflect the impact of a particular product sequence on the lower levels (i.e. the usage). As each product's part requirements are known, product weighting factors for minimizing usage could therefore be developed to reflect their part requirements. These weights could then be used in an implementation of the weighted, single level algorithm, which would run far faster than the DP. The separation of products with high part requirements would be achieved by the magnitudes of the weights assigned to them and would ensure that the products with the highest part requirements are kept as far apart in the assembly sequence as possible. This addresses the first reason for using weighting factors.

The second reason for using weighting factors is the concern that some lower level part [perhaps some such raw material as nuts and bolts which would be measured in units of 1000] could dominate the sequencing decision of a far more important product at the final assembly level. If this is of concern, then it would be more beneficial to assign weights to the individual products indicative of their lower level part requirements and use the single-level algorithm than to use a weighted version of the DP. The justification, used above, to satisfy the first reason, can apply equally well for this second reason.

In summary, this research has investigated the optimization of the scheduling of a mixed-model, JIT assembly process. An objective function for this model was introduced which has not been previously studied. A number of versions of this model have been considered. Various algorithms have been proposed for optimally solving these versions of the problem. Where appropriate, testing of these procedures has been performed. The outcome of these investigations formed the basis of this study.

Subsequent to the preparation of this thesis, Kubiak (1992) has provided a reduction for the two-level minimax model (and also for the quadratic version of the two-level sum function) demonstrating that it is an NP-hard problem. This result significantly validates the use of a DP procedure for the optimization of the multi-level problems and further underlines the need for experimental testing such as that performed in this thesis.

VIII.2 Extensions to this Research

There are a number of avenues for future research arising from this study. Some of these possible extensions will be outlined subsequently.

Clearly, there is a distinct need for research to be performed on the calculation of appropriate weighting factors. Nothing has been published on the methods to determine weights which achieve the goals required of them. For the single level minimax case, methods must be developed for determining weights which would achieve the goal of separating specified high demand products in the assembly sequence.

Studies are also required to find product weights which reflect the part usage inherently created by the order of final product assembly. The simplest example of this type of work is the pegged, multi-level case where each

product's weight was shown to explicitly correspond to the resulting lower level part usage. Further research must determine if similar weighting schemes exist which approximate usage in a like manner for the general multi-level case. These weights could then be used in the single-level weighted, usage problem. If weights are needed by the DP for the multi-level case, then approaches for determining values which allow for the magnitudes at each level to be compared are requisite. If weights of this nature are determinable, then they would also have an impact on solutions to multi-level sum functions.

The upper bound for the single-level problem was shown to be 1. This allows for the determination of an upper bound for the single-level sum function. As the maximum deviation for each copy of each product is less than or equal to this bound, an upper bound for the optimal solution of the sum function must necessarily be n_1D. No previous bound has been proved for the sum function. Use of the minimax bound might also allow for a reduction to a bottleneck assignment problem, as described in section III.6, if it could be used in conjunction with some form of interchange argument whereby the sequence dependent nature of the problem is eliminated.

Furthermore, this bound also suggests the possibility of developing a bicriterion procedure using both the minimax and the sum functions concurrently. Kubiak & Sethi (1989,1991) presented an assignment algorithm for the sum function which required the calculation of the penalty of assigning each copy of every product to every feasible time period. For problems of even a moderate size, this could require the calculation of an oppressively large number of such penalties. If, however, each copy's penalties were calculated only until their deviations equalled 1, then a vast number of the penalty

calculations could be eliminated. The optimal assignment using the algorithm of Kubiak & Sethi for the sum function of this reduced problem could then be determined. The sequence generated would ensure that a good solution to the sum function had been found such that no individual copy's deviation exceeded the upper bound of the minimax problem.

Variants of this problem would be to restrict the calculation of sum function penalties only to the edges of the bipartite, convex graph corresponding to some feasible (perhaps optimal) target value. If the optimal minimax value was used, then the sequence generated by using the assignment algorithm would be optimal with respect to the minimax criterion and the best sum function solution of all possible optimal minimax sequences (i.e. Pareto optimal). Conversely, the best minimax solution could be determined to the problem where the feasible edges would correspond to the reduced edge set of some optimal sum function assignment. These types of bicriterion problems have been studied by Berman et al. (1990), where it is shown that the addition of a bottleneck-type measure changes the complexity of the sum problem by a factor which is at most linear in the number of edges. Research of bicriterion measures could, therefore, be performed on problems in which a minimax objective as considered in this thesis, not a bottleneck measure, is used.

As finding optimal solutions to multi-level problems of a fixed size were constrained by the solution time, if practical applications of large problems using the minimax criterion are necessary, then heuristic algorithms for their solution are required. Therefore, research must be performed which produces various heuristics for the general minimax model. Fortunately, as a result of this

thesis, a substantial body of optimally solved, multi-level problems now exists against which the solutions produced by these newly develop heuristics could be compared. Further improvement of heuristic accuracy may allow for better screening methods to be implemented in the DP; thereby allowing for even larger problems than those considered in this thesis to be optimally solveable.

These and other questions can be addressed in subsequent research.

REFERENCES

Aho, A., J. Hopcroft and J. Ullman, The Design and Analysis of Computer Algorithms , Addison-Wesley, Reading, MA, 1974

Allaby, I., 1986, 'Allen-Bradley: New JIT Methods', Information Technology, May, 1986

Bailey, D. and T. Hubert (eds.), Productivity Measurement, Gower, London, 1980

Baker, K., Introduction to Sequencing and Scheduling, John Wiley, New York, 1974

Baker, K. and G. Scudder, 1990, 'Sequencing with Earliness and Tardiness Penalties', **Operations Research**, 38,1, 22-36

Belt, B., 1987, 'MRP and Kanban-A Possible Synergy?', **Production and Inventory Management**, 28, 1, 71-80

Berge, C, 1985, Graphs 2nd ed., North-Holland Mathematical Library, Elsevier Science Publishers B.V., Amsterdam, Netherlands

Berman, O., D. Einav and G. Handler, 1990, 'The Constrained Bottleneck Problem in Networks', **Operations Research**, 38,1, 178-81

Bevis, G., 1976, 'A Management Viewpoint on the Implementation of an MRP System', **Production and Inventory Management**, 17, 1

Billesbach, T., and M. Schneiderjans, 1989, 'Applicability of Just-In-Time Techniques in Administration', **Production and Inventory Management**, 30, 3, 40-45

Bitran, G. and L. Chang, 1987, 'A Mathematical Programming Approach to a Deterministic Kanban System', **Management Science**, 33, 4, 427-441

Bondy, J. and U. Murty, Graph Theory with Applications, North Holland, New York, NY, 1976

Bridgett, R., 1976, 'MRP-Philosophy or Technique', **Production and Inventory Management**, 17, 2

Burton, T., 1988, 'JIT/Repetitive Sourcing Strategies: "Tying the Knot" with your Suppliers', **Production and Inventory Management**, 25, 4, 38-42

Buzacott, J., 1989, 'Queueing Models of Kanban and MRP Controlled Production Systems', **Engineering Costs & Production Economics**, 17, 1, 3-20

Cadley, J., H. Heintz and L. Allocco ,1989, 'Insights from Simulating JIT Manufacturing', **Interfaces**, 19, 2, 88-97

Carlson, J., 1989, 'JIT Applications to Warehousing Operations', **Engineering Costs & Production Economics**, 17, 2, 315-322

Caves, D., L. Christensen and J. Swanson, 1980, 'Productivity in U.S. Railroads 1951-74', **Bell J. of Economics**, 11

Chartrand, G., 1977, Introductory Graph Theory , Dover Publications, Inc., New York, NY

Chartrand, G., and L. Lesniak, 1986, Graphs and Digraphs , Wadsworth & Brooks/ Cole Advanced Books & Software, Monterey, California

Cheng, T., 1988, 'The JIT Production: A Survey of its Development and Perception in the Hong Kong Electronics Industry', **OMEGA**, 16, 1, 25-32

Chyr, F., T. Lin and F-Y. Ho, 1990, 'Comparison Between Just-In-Time and EOQ Systems', **Engineering Costs & Production Economics**, 18, 3, 233-240

Connell, G., 1984, 'Quality at the Source: The First Step in Just-In-Time Production', **Quality Progress**, 11

Conover, W., Practical Nonparametrice Statistics 2nd ed., John Wiley & Sons, New York, NY, 1980

Conway, R., W. Maxwell and L. Miller, Theory of Scheduling, Addison-Wesley, Reading, Mass., 1967

Cosmetatos, G., 1983, 'Increasing Productivity in Exponential Queues by Server Sharing', **OMEGA**, 11, 2

Crawford, M. and J. Cox, 1990, 'Designing Performance Measurement Systems for JIT Operations', **International J. of Production Research**, 28, 11, 2025-2036

Crawford, K. and J. Blackstone, 1988, 'A Study of JIT Implementation and Operating Problems', **International J. of Production Research**, 26, 9, 1561-1568

Crosby, L., 1984, 'The Just-In-Time Manufacturing Process: Control of Quality and Quantity', **Production and Inventory Management**, 4,

Derigs, U., O. Goecke and R. Schrader, 1984, 'Bisimplicial Edges, Gaussian Elimination and Matchings in Bipartite Graphs', in Graph Theoretic Concepts in Computer Science Proceedings West Germany '84, U. Pape (ed.), Linz: Trauner Verlag, p79-87

DeToni, A., M. Caputo and A. Vinelli, 1988, 'Production Management Techniques: Push-Pull Classification and Application Conditions', **International J. of Operations and Production Management**, 8, 2, 35-51

Dilworth, J., Production and Operations Management3rd ed., Random House Inc., New York, 1986

Discenza, R., and F. McFadden, 1988, 'The Integration of MRP II and JIT through Software Unification', **Production and Inventory Management**, 29, 4, 49-53

Ebrahimpour, M., 1985, 'An Examination of Quality Management in Japan: Implications for Management in the United States', **J. of Operations Management**, 5, 4, 419-431

Ebrahimpour, M. and R. Fathi, 1985, 'Dynamic Simulation of a Kanban Production Inventory System', **International J. of Operations and Production Management**, 5, 5

Egbelu, P., and H. Wang, 1989, 'Scheduling for Just-In-Time Manufacturing', **Engineering Costs & Production Economics**, 16, 2, 117-124

Eilon, S., Elements of Production Planning and Control, MacMillan, New York, 1962

Eilon, S., B. Gold and J.Soeson, Applied Productivity Analysis for Industry, Pergamon Press, Oxford, 1976

Esparago, R.,1988, 'Kanban', **Production and Inventory Management**, 29, 1, 6-10

Fallon, D. and J. Brown, 1988, 'Simulating Just-In-Time Systems', **International J. of Operations and Production Management**, 8, 6, 30-45

Fieten, R., 1989, 'Integrating Key Suppliers-Essential Part of a Just-In-Time Concept', **Engineering Costs & Production Economics**, 15, 7, 185-189

Finch, B., 1986, 'Japanese Management Techniques in Small Manufacturing Companies: A Strategy for Implementation', **Production and Inventory Management**, 29, 3, 30-38

Finch, B. and J. Cox, 1986, 'An Examination of JIT Management for the Small Manufacturer: with an Illustration', **International J. of Production Research**, 24, 63-67

Flapper, S., J. Miltenburg and J. Wijngaard, 1991, 'Embedding JIT into MRP', **International J. of Production Research**, 29, 2, 329-341

Forbes. R., D. Jones and S. Marty, 1989, 'Managerial Accounting and Vendor Relations for JIT: A Case Study', **Production and Inventory Management**, 30, 1, 76-81

French, S., Sequencing and Scheduling, Ellis-Horwood Ltd., Chichester, West Sussex, England, 1982

Frederickson, G., 1983, 'Scheduling Unit-Time Tasks with Integer Release Times and Deadlines', **Information Processing Letters**, 16, p171-173

Fukuda, R., Managerial Engineering, Productivity Inc., Stamford, Connecticut, 1983

Gaither, N., Production and Operations Management 3rd ed., Dryden Press, New York, 1987

Gallo, G., 1984, 'An O(nlogn) Algorithm for the Convex Bipartite Matching Problem', **Operations Research Letters**, 16, p31-34

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

Garey, M., D. Johnson, B. Simons and R. Tarjan, 1981, 'Scheduling Unit Time Tasks with Arbitrary Release Times', **SIAM J. on Computing**, 10, 2

Garey, M., R. Tarjan and G. Wilfong, 1988, 'One Processor Scheduling with Symmetric Earliness and Tardiness Penalties', **Mathematics of Operations Research**, 13, 2, 330-348

Glover, F., 1967, 'Maximum Matchings in a Convex Bipartite Graph', **Naval Research Logistics Quarterly**, 4, 3

Goldstein, T. and J. Miltenburg, 1988, 'The Effects of Pegging in the Scheduling of Just-In-Time Production Systems', Working Paper 294, McMaster University

Gilbert, J., 1990, 'The State of JIT Implementation and Development in the U.S.A.', **International J. of Production Research**, 28, 6, 1099-1109

Gravel, M. and W. Price, 1988, 'Using Kanban in a Job-shop Environment', **International J. of Production Research**, 26, 6

Graves, S., 1981, 'A Review of Production Scheduling', **Operations Research**, 29, 2

Groeflin, H., H. Luss, M. Rosenwein and E. Wahls, 1989, 'Final Assembly Sequencing for Just-In-Time Manufacturing', **International J. of Production Research**, 27 ,2, 199-213

Groenevelt, H., and U. Karmarkar, 1988, 'A Dynamic Kanban System Case Study', **Production and Inventory Management**, 29, 2, 46-50

Gross, O., 1959, 'The Bottleneck Assignment Problem', The RAND Corporation, Paper P-1630

Grout, J., and M. Seastrand, 1987, 'Multiple Operation Lot-Sizing in a Just-In-Time environment', **Production and Inventory Management**, 28, 1, 23-27

Gupta, Y. and M. Gupta, 1989, 'A System Dynamics Model of a JIT-Kanban System', **Engineering Costs & Production Economics**, 18, 2, 117-130

Gupta, K. and J. Kyparisis, 1987, 'Single Machine Scheduling Reasearch', **OMEGA**, 15, 2

Hall, N., 1983, Zero Inventories, Dow-Jones-Irwin, Homewood, Ill

Hall, N., W. Kubiak and S. Sethi, 1991, 'Earliness-Tardiness Scheduling Problems, II: Deviation of Completion Times about a Restrictive Common Due Date', **Operations Research**, 39, 5, 847-856

Hall, N. and M. Posner, 1991, 'Earliness-Tardiness Scheduling Problems, 1: Weighted Deviation of Completion Times about a Common Due Date', **Operations Research**, 39, 5, 836-846

Hall, P., 1935, 'On Representatives of Subsets', **Journal of the London Mathematical Society**, 10, 26-30

Hannah, K., 1987, 'Just-In-Time: Meeting the Competitive Challenge', **Production and Inventory Management**, 28, 3,1-3

Heiko, L., 1989, 'A Simple Framework for Understanding JIT', **Production and Inventory Management**, 30, 4, 61-63

Hendrick, T., 1987, 'The pre-JIT/TQC Audit: First Step of the Journey', **Production and Inventory Management**, 28, 2, 132-143

Hendrick, T., 1988, 'The Fake Pull in a Kanban Environment: Acceptance Tradeoff or Violation of Principle?', **Production and Inventory Management**, 29, 2, 6-9

Hill, A., and T. Vollmann, 1986, 'Reducing Vendor Delivery Uncertainties in a JIT Environment', **J. of Operations Management**, 6, 4, 381-392

Huang, P., L. Rees and B. Taylor, 1984, 'The Japanese Just-In-Time Technique for a Multiline, Multistage Production System', **Decision Sciences**, 14, 326-344

Im, J., 1989, 'How does Kanban Work in American Companies?', **Production and Inventory Management**, 30, 4, 22-24

Im, J. and S. Lee, 1989, 'Implementation of Just-In-Time Systems in U.S. Manufacturing Firms', **International J. of Production Research**, 9, 1, 5-14

Im, J. and R. Schonberger, 1988, 'The Pull of Kanban', **Production and Inventory Management**, 29, 4, 54-58

Inman, R. and R. Bulfin, 1991, 'Sequencing JIT Mixed-Model Assembly Lines', **Management Science**, 37, 7, 901-904

Inman, R. and S. Mehra, 1989, 'Potential Union Conflict in JIT Implementation', **Production and Inventory Management**, 30, 4, 19-21

Inman, R. and S. Mehra, 1990, 'The Transferability of JIT Concepts to American Small Business', **Interfaces**, 20, 2, 30-37

Ishikawa, K., What is Total Quality Control? The Japanese Way, Prentice-Hall, Inc., Englewood Cliffs, N.J. 1985

Johnston, S., 1989, 'JIT: Maximizing its Success Potential', **Production and Inventory Management**, 30, 1, 82-86

Justis, R., 1981, 'America Feasts on Japanese Management Delicacies-Quality Circles, JIT and Kanban', Data Management, 19, 10

Kendall, M., Rank Correlation Methods, Hafner Publishing Co., Inc., New York, NY, 1962

Kimura, T and K. Terada, 1981, 'Design and Analysis of a Pull System: A Method of Multi-Stage Control', **International J. of Production Research**, 19

Kochar, S., R. Morris and W. Wong, 1987, 'The Local Search Approach to Flexible Flow Line Scheduling', Proc. of the Second International Conference on Production Systems, p175-86

Koten, J., 1982, 'Auto Makers have trouble with Kanban', The Wall Street Journal, April 7,

Kubiak, W., Private Communication

Kubiak, W. and S. Sethi, 1989, 'Optimal Level Schedules for Flexible Assembly Lines in JIT Production Systems', Working Paper, Faculty of Management, University of Toronto

Kubiak, W. and S. Sethi, 1991, 'A Note on "Level Schedules for Mixed-Model Assembly Lines in Just-In-Time Production Systems" ', **Management Science**, 37, 1, 121-123

Lambrecht, M. and L. Decaluwe, 1988, 'JIT and Constraint Theory: The Issue of Bottleneck Management', **Production and Inventory Management**, 29, 3, 61-66

Laver, R., 1991, 'Scrapping the Assembly Line', Maclean's Magazine, August 12, 28-29

Lawler, E., J. Lenstra and A. Rinnooy Kan, 'Recent Developments in Sequencing and Scheduling: A Survey', In Deterministic and Stochastic Scheduling, ed. (M. Dempster et al) Dordrecht, Holland: Reidel, 1982

Lee, L. ,1987, 'Parametric Appraisal of the JIT System', **International J. of Production Research**, 25, 10, 1415-1429

Lee, S., and M. Ebrahimpour, 1984, 'Just-In-Time Production System's: Some Requirements for Implementation', **International J. of Operations and Production Management**, 4, 3

Li, A. and H. Co, 1991, 'A Dynamic Programming Model for the Kanban Assignment Problem in a Multistage Multi-Period Production System, **International J. of Production Research**, 29, 1, 1-16

Lieberman, M., L. Lau and M. Williams, 1990, 'Firm-Level Productivity and Management Influence: A Comparison of U.S. and Japanese Automobile Producers', **Management Science**, 36, 10, 1193-1215

Lipski, W., and F. Preparata, 1981, 'Efficient Algorithms for Finding Maximum Matchings in Convex Bipartite Graphs and Related Problems', **Acta Informatica**, 15, p329-346

Luss, H., 1989, 'Synchronized Manufacturing at final Assembly and Feeder Shops', **International J. of Production Research**, 27

Luss, H., and M. Rosenwein, 1990, 'A Lot-Sizing Model for Just-In-Time Manufacturing', **J of the Operational Research Society**, 41, 3, 201-209

Maher, H., 1986, 'MRP II wont Schedule the Factory of the Future', **CIM Review**, 3, 1

Malley, J. and R. Ray, 1988, 'Information and Organizational impacts of Implementing a JIT System', **Production and Inventory Management**, 29, 2, 66-70

Marshall, B., 1977, 'Japanese Business Ideology and Labour Policy', **Columbia J. of World Business**, 12,1

McCormick, S., M. Pinedo, S, Shenker and B. Wolf, 1989, 'Sequencing in an Assembly Line with Blocking to Minimize Cycle Time', **Operations Research**, 37, 6, 925-935

Mellor, P., 1966, 'A Review of Job-Shop Scheduling', **Operational Research Quarterly**, 17

Miltenburg, J., 1986, 'Scheduling Mixed-Model Multi-Level Just-In-Time Production Systems', Working Paper 256, Faculty of Business, McMaster University

Miltenburg, J., 1989, 'Level Schedules for Mixed-Model Assembly Lines in Just-In-Time Production Systems', **Management Science**, 32, 2, 192-207

Miltenburg, J. and T. Goldstein, 1990, 'Developing Production Schedules which Balance Part Usage and Smooth Production Loads in Just-In-Time Production Systems', Working Paper 321, Faculty of Business, McMaster University

Miltenburg, J. and G. Sinnamon, 1989, 'Scheduling Mixed-Model, Multi-Level Assembly Lines in Just-In-Time Production Systems', **International J. of Production Research**, 27, 9, 1487-1509

Miltenburg, J., G. Steiner and S. Yeomans, 1990, 'A Dynamic Programming Algorithm for Scheduling Mixed-Model, Just-In-Time Production Systems', **Mathematical and Computer Modelling**, 13, 3, 57-66

Miltenburg, J. and J. Wijngaard, 1991, 'Designing and Planning in Just-In-Time Production Systems, **International J. of Production Research**, 29, 1, 115-131

Miyazaki, S., H. Ohta and N. Nishiyama, 1988, 'The Optimal Operation Planning of Kanban to Minimize the Total Operation Cost', **International J. of Production Research**, 26, 10, 1605-1611

Monden, Y., Toyota Production System, Industrial Engineering and Management Press, Cambridge, MA, 1983

Monden, Y., 1981[a], 'What Makes the Toyota Production System Really Tick', **Industrial Engineering**, 13, 1

Monden, Y., 1981[b], 'Adaptable Kanban System Helps Toyota Maintain JIT Production', **Industrial Engineering**, 13, 5

Monden, Y., 1981[c], 'Smoothed Production Lets Toyota Adapt to Demand Changes and Reduce Inventory', **Industrial Engineering**, 13, 8

Monden, Y., 1981[d], 'How Toyota Shortened Supply Lot Production Time, Waiting Time and Conveyor Time', **Industrial Engineering**, 13, 9

Moore, J., 1968, 'An n-job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs', **Management Science**, 15

Mosteller, F. and R. Rourke, Sturdy Statistics, Addison-Wesley Publishing Co., Reading, Mass., 1973

Nelson, R., 1981, 'Research on Productivity Growth and Productivity Differences: Dead Ends and New Departures', **J. of Economic Litterature**, 19

Newman, R., 1988, 'The Buyer-Supplier Relationship under Just-In-Time', **Production and Inventory Management**, 29, 3, 45-50

Ohno, T., Toyota Production System: Beyond Large Scale Production, Productivity Press, Cambridge, MA, 1988

Okamura, K., and H. Yamashita, 1979, 'A Heuristic Algorithm for the Assembly Line Model-Mix Sequencing Problem to Minimize the Risk of Stopping the Conveyor', **International J. of Production Research**, 17, 3, 233-247

Olhager, J., and B. Ostlund, 1990, 'An Integrated Push-Pull Manufacturing Strategy', **European J. of Operational Research**, 45, 2, 135-142

Oliver, N., 1990, 'Human Factors in the Implementation of Just-In-Time Production', **International J. of Operations and Production Management**, 10, 4, 32-40

Orlicky, J., Material Requirements Planning, McGraw-Hill, New York, 1975

Pan, A. and C. Liao, 1989, 'An Inventory Model under Just-In-Time Purchasing Agreements', **Production and Inventory Management**, 30, 1, 49-52

Parnaby, J., 1988, 'A Systems Approach to the Implementation of JIT Methodologies in Lucas Industries', **International J. of Production Research**, 26, 3, 483-492

Pearl, J., Heuristics: Intelligent Search Strategies for Computer Problem Solving, Addison-Wesley Publishing Co, Inc., Reading, Mass, 1984

Philipoom, P., L. Rees, B. Taylor and P. Huang, 1987, 'An Investigation of the Factors Influencing the Number of Kanbans Required in the Implementation of the JIT Technique with Kanbans', **International J. of Production Research**, 25, 3

Plenert, G., and T. Best, 1986, 'MRP, JIT and OPT: What's "Best"? ', **Production and Inventory Management**, 27, 2, 22-29

Pyke, D., and M. Cohen,1990, 'Push and Pull in Manufacturing and Distribution Systems', **J. of Operations Management**, 9, 1, 24-43

Rao, A., 1989(a), 'Manufacturing Systems-Changing to Support JIT', **Production and Inventory Management**, 30, 2, 18-21

Rao, A., 1989(b), 'A Survey of MRPII Software Suppliers trends in Support of Just-In-Time', **Production and Inventory Management**, 30, 3, 14-17

Rao, A., and D. Scherage, 1988, 'Moving from Manufacturing Resource Planning to Just-In-Time Manufacturing', **Production and Inventory Management**, 29, 1, 44-49

Rege, K., 1988, 'Approximate Analysis of Serial Manufacturing Lines with Buffer Control', **Information & Decision Technologies**, 14, 1

Rinnooy Kan, A. Machine Scheduling Problems: Classification, Complexity and Computations, Martinus Nijhoff, The Hague, Holland, 1976

Sarker, B. and R. Harris, 1988, 'The Effect of Imbalance in a Just-In-Time System: A Simulation Study', **International J. of Production Research**, 26, 1, 1-18

Schonberger, R., Japanese Manufacturing Techniques, The Free Press, New York, 1982

Seidman, A., 1988, 'Regenerative Pull (Kanban) Production Control Policies', **European J. of Operational Research**, 35, 3, 401-413

Sewell, G., 1990, 'Management Information Systems for JIT Production', **OMEGA**, 18, 5, 491-503

Simons, B., 1978, 'A Fast Algorithm for Single Processor Scheduling', IEEE 19th Annual Symposium on Foundations of Computer Science (formerly Annual Symposium on Switching & Automata Theory, Long Beach

Sipper, D., and R. Shapira, 1989, 'JIT vs WIP-A Trade-off Analysis', **International J. of Production Research**, 27, 6, 903-914

Smith, H., K. Mangelsdorf, J. Luna and R. Reid,1989, 'Supplying Ecuador's Health Workers just in Time', **Interfaces**, 19, 3, 1-12

So, K. and S.Pinault, 1988, 'Allocating Buffer Storages in a Pull System', **International J. of Production Research**, 26, 12, 1959-1980

South, J., 1986, 'A Minimum Production Lot-Size Formula for Stockless Production, **Production and Inventory Management**, 27, 2

Spence, A., and E. Porteus, 1987, 'Set-up Reduction and Increased Effective Capacity', **Management Science**, 33

Stevenson, W., Production/Operations Management 3rd ed., Richard D. Irwin Inc., Homewood, Ill., 1990

Sugimori, Y., K. Kusunoki, F. Cho and S. Uchikawa, 1977, 'Toyota Production System and Kanban System- Materialization of Just-In-Time Respect-for Human Systems', **International J. of Production Research**, 15

Suri, R. and S. DeTreville, 1986, 'Getting from "Just-In-Case" to Just-In-Time', **J. of Operations Management**, 6, 3, 295-304

Tarjan, R., Data Structures and Network Algorithms, Society for Industrial and Applied Mathematics, Philadelphia, Penn., 1983

Toomey, J., 1989, 'Establishing Inventory Control Options for Just-In-Time Applications', **Production and Inventory Management**, 30, 4, 13-15

Wang, H., 1990, 'Determining the Number of Kanbans: A Step Toward Non-stock Production', **International J. of Production Research**, 28, 11, 2101-2115

Westbrook, R., 1988, 'Time to Forget Just-In-Time? Observations on a Visit to Japan', **International J. of Operations and Production Management**, 8, 4, 5-21

Wildemann, H., 1988, 'Just-In-Time Production in West Germany', **International J. of Production Research**, 26, 3, 521-538

Willis, T., and C. Huston, 1990, 'Vendor Requirements and Evaluation in a Just-In-Time Environment', **International J. of Operations and Production Management**, 10, 4, 41-50

Willis, T., and W. Suter, 1989, 'The Five M's of manufacturing: A JIT Conversion Life Cycle', **Production and Inventory Management**, 30, 1, 53-57

Wilson, G., 1985, 'Kanban Scheduling: Boon or Bane?', **Production and Inventory Management**, 26, 3

Wittrock, R., 1985, 'Scheduling Algorithms for Flexible Flow Lines', **IBM J. of Research & Development**, 29

Wittrock, R., 1988, 'An Adaptable Scheduling Algorithm for Flexible Flow Lines', **Operations Research**, 36

Wortman, J. and W. Monhemius, 1984, 'Kanban-Its Use as a Final Assembly Scheduling Tool Within MRP', Operational Research '84, ed. J. Brans, Elsevier Science Publishers

APPENDIX 1

1. Introduction

Frederickson (1983) considers the problem of scheduling n, unit-time tasks, $T_1, T_2, \ldots, T_n$ on m identical processors, where each task has an integer release time r_i and an integer deadline d_i such that no task starts prior to its release time or completes beyond its deadline. The problem can be solved by scheduling according to the earliest deadline rule (Garey et al, 1981) and Frederickson demonstrates how this solution technique can be implemented to run in $O(n)$ time. Frederickson, however, does not account for a possible infeasibility condition, described as deficiency case 2 in chapter 4, and consequently the algorithm could potentially construct a schedule which is not feasible. Deficiency case 2 corresponds to the condition of having to schedule too many tasks into the available starting times. However, a test for this case of deficiency can be incorporated into the algorithm to allow either the determination of the best feasible schedule or the conclusion that no feasible schedule exists. This modification requires the addition of only constant time, thereby preserving the linearity of the algorithm.

2. Scheduling Algorithm

Frederickson's algorithm involves three phases; a sorting phase using a table structure; a partitioning phase where the tasks are separated into subsets, each of which can be scheduled independently; and a final phase in which each subset is translated into an off-line minimum problem on integers not exceeding the size of the subset.

In the first phase, tasks are placed into an uninitialized table which contains M locations, where M is the size of the largest deadline. An auxiliary stack of maximum size n is maintained to indicate which table locations possess meaningful information. Meaningful information at table location r' is indicated by having the contents point to location x in the stack, which in turn points back to location r' in the table. Associated with each meaningful location r' in the table will be a list of tasks with release time r'. The lengths of these lists are maintained as the tasks are entered into the table.

The second phase partitions the schedule into *compact sections* where each section starts at some release time, r', and extends until the number of tasks released during the section is less than or equal to the number of time units available within the section. This partitioning makes use of the table and stack structure employed in phase 1 and can be considered as analogous to ensuring that deficiency case 1 is satisfied within each compact section.

In the third phase, each compact section is treated as an individual scheduling problem and the release times and deadlines of the tasks within each section are transformed and reindexed so that the first release date occurs at time 0 and the transformed deadlines, $\hat{d}_i$ are such that $\hat{d}_i \leq \hat{d}_j$ implies that $i \leq j$. A sequence of instructions is then constructed and executed for an off-line minimum problem and the schedule is extracted. An off-line minimum problem is defined as follows: There is initially an empty set S and a sequence of two types of instructions, INSERT(i) and EXTRACT-MIN. The instructions are executed in turn and either insert an integer $i \leq n$ into S or find and delete the minimum element of S.

A sequence of EXTRACT-MIN instructions is constructed whereby the number of these instructions equals the number of tasks within the compact section. INSERT instructions are placed in front of the appropriate EXTRACT-MIN instructions to correspond to when the various tasks are released. This sequence of EXTRACT-MIN and INSERT instructions can then be executed as an off-line minimum problem.

If index i is removed by the j^{th} EXTRACT-MIN instruction then task T_i will be scheduled in time step $r'-1+\lceil j/m \rceil$ on processor $1+(j - 1) \bmod m$. It is possible, however, that even though the number of release times within a compact section is greater than or equal to the number of tasks, that the deadlines of a subset of the jobs in the compact section would be such that the cardinality of this subset exceeds the number of starting times available and that no feasible schedule is possible. This corresponds to deficiency case 2 and can best be demonstrated by a simple example.

Example: Assume that there are five tasks to be scheduled on a single processor, where each task,i, is denoted by the triple (i,r_i,d_i). These tasks are: $(1,1,2)$, $(2,1,3)$, $(3,1,3)$, $(4,2,5)$ and $(5,4,6)$. Using the method of Frederickson, phase 1 and phase 2 will place all of these tasks into the same compact section. The instruction sequence for the applicable off-line minimum problem is; INSERT(1), INSERT(2), INSERT(3), EXTRACT-MIN, INSERT(4), EXTRACT-MIN, EXTRACT-MIN, INSERT(5), EXTRACT-MIN, EXTRACT-MIN which will construct the schedule 1-2-3-4-5. This schedule is not feasible because task 3, which has a deadline of 3, will complete at time 4. The infeasibility occurs because tasks 1, 2 and 3 all have release time 1 and must all

be completed by time 3, thereby requiring that three tasks be scheduled into two starting times.

3. Conclusion

Frederickson's algorithm can be modified in one of two ways. Firstly, if the above infeasibility condition is encountered then the algorithm could stop, declaring that no feasible schedule exists for the set of tasks. Alternatively, if the deficiency case is encountered, the algorithm could be modified to construct a sequence for the maximum number of jobs which can be feasibly scheduled. These modifications can be achieved in the following way.

Define an EXTRACT-MIN instruction to be *feasible* if the task selected by it can be scheduled within its release date and deadline. Modification 1 would stop if an infeasible EXTRACT-MIN were encountered. That is, if j-1 feasible EXTRACT-MIN instructions have been encountered thus far by the algorithm, and the index i selected by the current EXTRACT-MIN is such that $d_i \leq \lceil j/m \rceil$ then task T_i cannot be feasibly scheduled. Modification 2 would state that if index i is removed by the j^{th} feasible EXTRACT-MIN, then, letting $k=\max\{\lceil j/m \rceil, r_i\}$, task T_i will be scheduled in time r'-1+k. Either of these modifications could be performed in constant time, thereby preserving the linear time nature of Frederickson's original algorithm.

APPENDIX 2

A Linear Time Algorithm for Maximum Matchings in Convex Bipartite Graphs

1. Introduction

Let $G=(V_1,V_2,E)$ represent an undirected, bipartite graph. V_1 and V_2 is a partition of the vertices and E is the edge set in which each edge (i,j) is such that $i\varepsilon V_1$ and $j\varepsilon V_2$. A matching is a subset, M, of these edges such that no two edges in M are incident to the same vertex (Berge 1985). M is of maximum cardinality (or simply maximum) if it contains the maximum number of edges.

Glover (1967) describes a special instance of a bipartite matching problem which he refers to as that of determining the maximum matching in a convex, bipartite graph. The graph G is *V_2-convex* if $(i,j)\varepsilon E$ and $(i,k)\varepsilon E$ with $j,k\varepsilon V_2$ and $j<k$ implies that $(i,l)\varepsilon E$ for $j\le l\le k$ (Derigs et al 1984). For a convex graph, the required ordering on V_2 is explicity provided by the values of some parameter associated with each $j\varepsilon V_2$. It can therefore be assumed that a convex graph is given by specifying the ordering on V_2 and by specifying, for every $i\varepsilon V_1$, two values a_i and b_i the smallest and largest elements respectively of the interval of the vertices of V_2 connected to i.

Glover provided an $O(|E|)$ algorithm for the maximum matching problem in convex, bipartite graphs. Let $|V_1|=n$ and $|V_2|=m$. Lipski and Preparata (1981) presented an almost linear time $O(m+nA(n))$ algorithm; where A(n) is a very slowly growing function related to the functional inverse of

Ackerman's function [for a description of the Ackerman function see Tarjan (1983, p24-29)]. In this appendix, an $O(n)$ algorithm is given for the maximum matching problem using data structures applied by Frederickson (1983) to a machine scheduling problem. This algorithm also addresses the issue of matching deficiency case 2, described in chapter 4, for ensuring the feasibility of a matching which is not accounted for by the algorithm of Frederickson (see Appendix 1).

2. Matching Algorithm

The matching algorithm requires the execution of three specific stages. The first stage involves bucketing the vertices of V_1 into an uninitialized table with $T=\max_i\{b_i\}$ locations. An auxiliary stack, of maximum size n, is maintained to indicate which table locations possess meaningful information. This type of table and stack structure is discussed in Aho et al (1974, p71). Meaningful information at table location v is indicated by having the contents point to location x in the stack, which in turn points back to location v in the table. For each vertex $i \varepsilon V_1$, an entry is made into table location a_i. Associated with each meaningful table location v will be a list, L(v), of those vertices having $a_i=v$. The actual length of each of these lists is updated as each vertex is entered into the table. From the description above, each pointer in the stack must correspond to a vertex a_i, $i \varepsilon V_1$. Hence, when this first stage completes, the n pointers in the stack will consist of the entire set of the n vertices a_i, $i \varepsilon V_1$, arranged in the stack by the order in which each vertex i was placed into the table.

Stage two partitions G into distinct components of V_2 by combining some of the lists of vertices created in stage 1. Each component starts at a $v \varepsilon V_2$ and extends until the number of vertices $i \varepsilon V_1$ connected to the component is less than or equal to the number of vertices of V_2 in the component. This stage is anologous to a check for the matching deficiency case 1.

The process of creating these components is to examine the vertices of V_2 to which vertices from V_1 can potentially be matched. For each vertex $v \varepsilon \{a_i\}$ in the stack, if v has not been previously examined then the following procedure is performed:

A V_2 component is constructed by scanning through the table starting from location v. When this scan terminates, a list, L(v), of vertices $i \varepsilon V_1$, along with the number of vertices in the list, will be associated with v. During the scan, each potential value of an a_i say the vertex v', is checked by using the pointer into the stack to determine if it is a meaningful table entry (i.e a valid a). If v' is a valid a_i and has not been examined previously, then the list of V_1 vertices, i, in which a_i=v' (i.e. L(v')) is appended onto the current component's list, L(v), and its count is updated. If v' is a valid a_i that has already been examined, then the component starting at v' (i.e. L(v')) is incorporated into the current component by adding the number of vertices contained in it and appending them onto the list, L(v), of the current component. In this case, the scan continues after bypassing the values of a_i included in the component that started at v'. Hence, previously encountered components can be incorporated into the current component in constant time.

No more than n table entries will be examined in the second stage and no entry can be examined more than twice. Upon the conclusion of this stage,

there is a list of V_2-components of the graph and a list of V_1-vertices available for matching into each of these components.

The third stage requires that each component be processed separately. The following transformation is performed on each component. Let z be the number of vertices in the component and let v be the value of the smallest V_2 vertex in the component. Define the modified a_i and b_i to be,

$$\hat{a}_i = (a_i - v + 1) \qquad \text{and} \qquad \hat{b}_i = \min\{b_i - v + 1, z\}$$

Reindex the vertices so that $\hat{b}_i < \hat{b}_k$ implies that $i<k$. Bucketsort the pairs $(\hat{a}_i, i)$ into lexicographically increasing order.

Now construct and execute a sequence of instructions for an off-line minimum problem (Aho et al 1974) and extract the matching. An off-line minimum problem is such that there is initially an empty set S and a sequence consisting of the instructions, INSERT(i) and EXTRACT-MIN. The instructions are executed sequentially and will either insert an integer i no greater than n into the set S or find and delete the minimum element of S. The problem is off-line as the whole sequence of instructions must be supplied before any instruction can be executed. Construct a sequence of z EXTRACT-MIN instructions. Before the k^{th} EXTRACT-MIN instruction, k=1,2,...,z, place INSERT(i) instructions for all i with $\hat{a}_i = k$ in order of the previously determined lexicographic order. When an EXTRACT-MIN instruction is executed then define it to be the *j^{th} feasible* EXTRACT-MIN, $j \le z$, if either the index i selected by it is such that $\hat{b}_i \ge j$ or if S=o. Otherwise the EXTRACT-MIN instruction is *infeasible*. The matching is determined as follows: If index i is removed by the j^{th} feasible EXTRACT-MIN instruction, then, letting $k = \max\{j, \hat{a}_i\}$, i is matched to the $(v-1+k)^{th}$ vertex of V_2. An infeasible EXTRACT-MIN would correspond to encountering the matching

deficiency case 2. Here the algorithm does not stop since a maximum matching, as opposed to a perfect matching, is sought.

Theorem

The maximum matching problem in the V_2-convex bipartite graph $G=(V_1,V_2,E)$ with $|V_1|=n$ can be solved in $O(f(n))$ time, where f(n) is the time to solve the off-line minimum problem.

Proof

The preceding approach can be used to determine the maximum matching in the graph as it would construct the same matching as the algorithm of Glover (1967). In stage 1, inserting vertices into the table requires constant (i.e. $O(n)$) time. Similarly, in stage 2, at most n entries will be examined in the table and no entry will be examined more than twice. A constant time is expended per table entry examined; thus this stage requires $O(n)$ time. In stage 3, the translation phase requires $O(n)$ time. The solution of a sequence of INSERT and EXTRACT-MIN instructions requires $O(f(z))$ time, where z is the number of V_1 vertices in a component. Hence, matchings in all components will require a total of $O(f(n))$ time. □

Corollary

The maximum matching problem in convex bipartite graphs can be solved in $O(n)$ time, since Tarjan (1983) has shown the off-line minimum problem to be linear. □

APPENDIX 3

SAMPLE MULTI-LEVEL PROBLEM

This appendix provides the information for an L=4 level problem with n_1=8 products and a total demand of D_1=500. The number of parts at each level are as follows; at level 2 there are n_2=24 parts, at level 3 there are n_3=46 parts, and at level 4 there are n_4=58 parts. The level 1 product requirements vector is:

$$\mathbf{D} = [20, 59, 86, 114, 12, 48, 63, 98] .$$

The range of part requirements for each product at every level is R_2=20, R_3=20 and R_4=20. The part requirements for each product are shown; for level 2 in table A.1, for level 3 in table A.2, and for level 4 in table A.3. The rows of each of these tables corresponds to a particular part in that level and the entry under each product in a particular row corresponds to the number of parts of the type in that row which are required in the production of one unit of that product. For example in table A.1 for part 1, product 1 requires t_{121}=19 of these parts, product 2 requires t_{122}=11 of these parts, product 3 requires t_{123}=2 of these parts and so on.

The DP was run using this data with the following results. The optimal, minimum value of the maximum deviation for a production schedule was 17.658. The screening value, corresponding to the best heuristic solution, was 17.879. Using this screening value as a filter, the DP generated only 3,219 sets out of the 50,879,430,201,600 potential number of feasible sets in determining the optimal solution in 2 minutes 59 seconds. The optimal sequence is shown in table A.4, which shows which level 1 product is to be produced in each stage.

Table A.1: LEVEL 2 PART REQUIREMENTS

	Product							
Part	1	2	3	4	5	6	7	8
1	19	11	2	2	12	5	3	16
2	17	3	12	17	2	1	0	7
3	19	19	6	10	6	2	18	14
4	7	18	12	8	5	17	7	6
5	6	0	7	14	18	4	8	1
6	1	12	13	15	5	12	1	7
7	15	14	13	3	12	3	13	18
8	6	16	6	12	10	4	18	19
9	4	3	2	8	16	13	7	12
10	4	5	15	14	12	5	4	5
11	0	10	13	3	2	18	11	10
12	15	19	9	13	6	9	5	16
13	13	8	12	11	2	11	17	13
14	9	9	11	10	9	11	13	0
15	16	1	4	4	9	0	1	9
16	11	19	4	11	14	14	17	2
17	3	9	0	14	16	13	8	6
18	16	9	2	15	16	12	12	15
19	11	9	4	5	12	1	3	10
20	10	1	19	17	6	11	6	13
21	15	4	5	10	5	19	1	13
22	7	3	16	13	10	19	11	6
23	7	2	5	5	15	15	10	10
24	8	9	11	18	10	3	8	13

TABLE A.2: LEVEL 3 PART REQUIREMENTS

	Product							
Part	1	2	3	4	5	6	7	8
1	0	5	19	19	9	9	12	2
2	7	8	0	0	17	0	15	11
3	12	17	10	4	18	7	1	1
4	1	4	12	9	4	19	12	9
5	7	13	1	13	2	0	2	2
6	13	0	7	13	0	13	12	4
7	18	9	8	13	5	11	16	3
8	11	8	12	0	18	17	11	0
9	3	0	0	6	0	10	17	6
10	8	1	19	6	13	3	18	13
11	7	5	9	19	13	11	3	11
12	10	2	4	15	2	4	18	8
13	10	17	5	4	6	11	12	15
14	15	9	14	16	12	9	0	18
15	8	8	9	2	0	9	13	17
16	3	9	7	2	1	4	11	2
17	17	7	8	16	2	4	1	8
18	14	5	2	10	5	18	13	6
19	19	17	6	11	4	18	12	17
20	14	4	15	1	15	0	15	0
21	1	13	3	15	12	5	14	4
22	0	3	9	14	9	4	4	15
23	3	8	2	16	5	3	5	4
24	0	4	12	9	11	2	3	8
25	0	17	8	0	2	0	2	2
26	3	19	11	4	15	19	4	14

27	8	14	1	12	4	4	17	12
28	16	18	0	16	9	18	19	6
29	3	15	6	14	18	9	12	0
30	12	19	1	15	1	0	16	18
31	10	8	12	1	2	9	10	16
32	11	18	11	8	19	1	13	19
33	2	2	2	5	7	3	11	17
34	8	15	3	14	19	14	10	15
35	9	16	8	8	1	2	5	6
36	12	6	12	5	4	4	4	4
37	16	8	4	6	16	4	4	10
38	12	8	16	4	11	12	3	1
39	12	4	3	3	6	17	1	17
40	1	10	3	3	14	5	0	9
41	10	4	5	14	15	10	16	18
42	18	3	11	2	9	7	3	3
43	12	14	16	0	19	1	16	1
44	14	6	18	10	14	2	16	8
45	13	14	19	11	12	7	6	16
46	16	16	15	0	17	12	6	16

TABLE A.3: LEVEL 4 PART REQUIREMENTS

	Product							
Part	1	2	3	4	5	6	7	8
1	6	10	19	7	12	6	10	7
2	16	2	11	17	15	18	8	3
3	3	7	3	0	16	17	3	13
4	6	4	5	10	4	14	7	0
5	17	11	4	16	2	5	15	8
6	10	18	5	10	8	19	3	2
7	10	11	3	9	9	7	15	13
8	3	11	17	14	4	0	10	5
9	1	7	3	12	15	7	8	4
10	5	17	5	7	5	19	1	19
11	14	16	6	13	10	13	9	14
12	3	3	13	5	11	4	10	15
13	8	11	10	4	4	4	1	16
14	5	8	0	17	14	8	14	7
15	8	12	12	4	16	10	19	10
16	7	17	3	10	9	17	3	14
17	8	0	3	1	19	4	4	2
18	2	12	13	16	4	6	8	8
19	5	11	8	17	15	14	3	12
20	7	6	14	9	5	14	12	11
21	5	12	5	12	19	17	8	15
22	14	1	1	14	7	18	17	6
23	14	10	3	7	9	16	15	6
24	18	17	4	10	4	3	12	18
25	3	11	3	9	16	9	2	4
26	7	5	18	11	17	19	4	16

27	1	9	13	12	11	7	10	15
28	6	7	8	14	19	17	6	17
29	1	3	9	4	15	5	17	15
30	14	3	9	18	9	14	4	13
31	10	10	16	8	6	17	3	17
32	3	10	17	11	19	19	8	13
33	16	18	1	8	11	10	3	10
34	14	11	17	15	2	15	19	4
35	1	0	17	13	12	15	9	6
36	6	12	14	5	1	0	1	18
37	3	9	19	16	14	4	9	5
38	17	16	17	17	17	1	3	17
39	12	9	4	6	17	5	2	13
40	1	13	17	9	0	1	18	11
41	13	12	10	3	4	10	3	13
42	15	19	3	6	16	1	19	8
43	19	10	12	8	5	13	16	4
44	3	5	1	6	16	9	1	15
45	15	9	5	4	14	4	15	5
46	10	15	18	17	4	14	18	0
47	13	1	19	0	9	2	4	0
48	10	16	14	11	8	1	2	19
49	14	2	19	15	9	7	13	4
50	12	12	4	16	7	5	2	0
51	4	1	6	3	16	14	19	0
52	11	4	17	10	0	2	3	1
53	14	18	13	14	7	11	4	12
54	17	13	15	19	15	7	4	1
55	18	14	3	0	4	9	14	18
56	1	3	10	1	18	5	17	1
57	15	8	12	19	18	14	17	5
58	7	11	2	13	0	4	5	4

TABLE A.4: OPTIMAL PRODUCTION SEQUENCE

Stage	Product	Stage	Product	Stage	Product	Stage	Product
1	8	26	6	51	4	76	7
2	4	27	3	52	7	77	4
3	3	28	8	53	8	78	8
4	7	29	7	54	2	79	6
5	6	30	4	55	3	80	3
6	2	31	2	56	4	81	2
7	4	32	3	57	6	82	4
8	8	33	8	58	7	83	7
9	3	34	4	59	8	84	3
10	1	35	7	60	4	85	8
11	7	36	1	61	3	86	4
12	4	37	6	62	1	87	1
13	8	38	2	63	2	88	6
14	3	39	3	64	4	89	2
15	2	40	4	65	8	90	3
16	6	41	8	66	5	91	8
17	4	42	4	67	3	92	4
18	8	43	7	68	4	93	7
19	7	44	8	69	6	94	4
20	3	45	3	70	7	95	9
21	4	46	4	71	8	96	3
22	2	47	2	72	4	97	2
23	8	48	6	73	2	98	6
24	5	49	3	74	3	99	7
25	4	50	8	75	8	100	4

TABLE A.4 (cont.): OPTIMAL PRODUCTION SEQUENCE

Stage	Product	Stage	Product	Stage	Product	Stage	Product
101	8	126	3	151	8	176	3
102	3	127	8	152	4	177	8
103	4	128	6	153	6	178	4
104	5	129	4	154	3	179	7
105	8	130	7	155	8	180	8
106	2	131	8	156	4	181	5
107	4	132	3	157	7	182	4
108	7	133	2	158	2	183	2
109	3	134	4	159	6	184	6
110	8	135	1	160	4	185	3
111	6	136	8	161	3	186	8
112	4	137	3	162	8	187	4
113	2	138	7	163	7	188	7
114	3	139	4	164	4	189	3
115	7	140	8	165	2	190	2
116	8	141	6	166	1	191	4
117	4	142	3	167	3	192	8
118	1	143	2	168	8	193	1
119	3	144	4	169	4	194	6
120	8	145	8	170	7	195	7
121	6	146	7	171	8	196	3
122	4	147	5	172	3	197	4
123	2	148	4	173	6	198	8
124	7	149	3	174	2	199	2
125	4	150	2	175	4	200	4

TABLE A.4 (cont.): OPTIMAL PRODUCTION SEQUENCE

Stage	Product	Stage	Product	Stage	Product	Stage	Product
201	3	226	7	251	2	276	6
202	8	227	4	252	4	277	2
203	7	228	8	253	3	278	3
204	4	229	5	254	8	279	8
205	6	230	4	255	6	280	4
206	8	231	3	256	4	281	7
207	3	232	8	257	7	282	3
208	2	233	2	258	8	283	8
209	4	234	4	259	3	284	4
210	7	235	6	260	2	285	2
211	8	236	7	261	4	286	6
212	3	237	3	262	1	287	4
213	4	238	8	263	8	288	1
214	1	239	1	264	3	289	3
215	6	240	4	265	7	290	8
216	2	241	7	266	6	291	7
217	3	242	8	267	4	292	4
218	8	243	3	268	8	293	2
219	4	244	2	269	5	294	3
220	7	245	4	270	2	295	8
221	4	246	6	271	4	296	6
222	8	247	8	272	3	297	4
223	3	248	3	273	8	298	7
224	2	249	4	274	4	299	8
225	6	250	7	275	7	300	3

TABLE A.4 (cont.): OPTIMAL PRODUCTION SEQUENCE

Stage	Product	Stage	Product	Stage	Product	Stage	Product
301	4	326	4	351	5	376	8
302	2	327	2	352	2	377	7
303	8	328	6	353	4	378	2
304	7	329	3	354	3	379	6
305	4	330	8	355	7	380	4
306	3	331	7	356	8	381	8
307	6	332	4	357	4	382	3
308	2	333	2	358	2	383	4
309	4	334	3	359	3	384	2
310	8	335	1	360	6	385	7
311	3	336	8	361	8	386	8
312	5	337	4	362	4	387	3
313	8	338	7	363	7	388	1
314	4	339	6	364	1	389	4
315	7	340	3	365	3	390	8
316	6	341	8	366	8	391	6
317	1	342	4	367	4	392	3
318	2	343	7	368	7	393	7
319	4	344	2	369	8	394	4
320	3	345	4	370	6	395	2
321	8	346	8	371	3	396	8
322	7	347	3	372	2	397	5
323	4	348	6	373	4	398	4
324	8	349	8	374	8	399	3
325	2	350	4	375	3	400	8

TABLE A.4 (cont.): OPTIMAL PRODUCTION SEQUENCE

Stage	Product	Stage	Product	Stage	Product	Stage	Product
401	4	426	8	451	8	476	3
402	7	427	3	452	3	477	4
403	6	428	2	453	6	478	8
404	2	429	4	454	2	479	2
405	3	430	8	455	4	480	4
406	8	431	7	456	3	481	3
407	4	432	6	457	8	482	7
408	7	433	4	458	7	483	8
409	4	434	3	459	4	484	4
410	8	435	2	460	8	485	6
411	3	436	5	461	1	486	2
412	2	437	8	462	3	487	3
413	6	438	4	463	2	488	8
414	1	439	1	464	4	489	4
415	4	440	3	465	6	490	7
416	8	441	4	466	7	491	1
417	3	442	8	467	4	492	3
418	7	443	7	468	8	493	8
419	4	444	6	469	3	494	4
420	2	445	2	470	2	495	2
421	3	446	3	471	7	496	6
422	6	447	4	472	4	497	7
423	8	448	8	473	5	498	3
424	4	449	7	474	8	499	8
425	7	450	4	475	6	500	4