5-1999

The multi-products EMQ model

John Roy Gray

To the Graduate Council:

I am submitting herewith a dissertation written by John Roy Gray entitled "The multi-products EMQ model." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Management Science.

M. M. Srinivasan, Major Professor

We have read this dissertation and recommend its acceptance:

M. R. Bowers, K. C. Gilbert, R. S. Sawhney

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by John Roy Gray entitled "The Multi-Products EMQ Model." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Management Science.

[illegible]

Dr. M. M. Srinivasan, Major Professor

We have read this dissertation
and recommend its acceptance:

[illegible]

Dr. M. R. Bowers

[illegible]

Dr. K. C. Gilbert

[illegible]

Dr. R. S. Sawhney

Accepted for the Council:

[illegible]

Associate Vice Chancellor and
Dean of the Graduate School

THE MULTI-PRODUCTS EMQ MODEL

A Dissertation
Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

John Roy Gray
May 1999

DEDICATION

This dissertation is dedicated to my wife

Brenda Olhausen Gray

whose support, encouragement, and forbearance made this dissertation possible.

ACKNOWLEDGEMENTS

I owe a debt of gratitude to Dr. George H. Worm, my good friend and former roommate. Without George's encouragement, moral support, advice and outright prodding I would not have stayed the course to complete this dissertation. I doubt that I will ever be able to pay this debt.

I am particularly grateful to Ms. Lisa A. Hurst of ***Drafting and More***, Knoxville, Tennessee, for doing the word processing and some of the graphics required to convert my almost unreadable notes into a publishable document. Without Lisa's patience, superb drafting and word processing skills, and outstanding work ethnic this dissertation would not have been completed on time. Throughout this often difficult process Lisa has always remained a competent professional. I have been doubly rewarded because, in addition to receiving her support, I have gained a new friend.

Finally, I am also grateful to Dr. Kenneth C. Gilbert, University of Tennessee, Management Science Program. Dr. Gilbert gave me a second chance. I very much appreciate his efforts on my behalf.

ABSTRACT

Since the introduction of the classical economic manufacturing quantities (EMQ) concept early in the twentieth century, many variants of the single-product EMQ model have been solved. These single-product EMQ models usually suppose the product is produced cyclically every T time units. This dissertation examines the general cyclical model (GCM), a generalization of the single-product EMQ. In the GCM $n \geq 2$ products are produced on a single facility according to cyclical schedules. The GCM, unlike some variants of the multi-products, single-facility problem, permits each product i to be produced every T_i time units where it is not necessary that $T_i = T_j$ if $i \neq j$. However, a schedule of n products must be feasible; that is, only one product may occupy the facility at a time. Thus, the objective of the GCM is to find cycle times $\{T_i\}$ that minimize the inventory and production costs subject to the restriction that the schedule is feasible. To mathematically address this feasibility problem, delay times $\{d_i\}$ are introduced where d_i is the time at which the first use period of product i begins. Then conditions are given that are both necessary and sufficient to assure that a specified schedule $\{T_i, d_i\}$ of n products is feasible. These feasibility conditions remove a major handicap suffered by previous researcher. Then delay-independent necessary and sufficient feasibility conditions are derived for the two-products and the three-products case of the GCM. Also, delay-independent necessary feasibility conditions are derived for the four-products case. Finally, an efficient algorithm is developed that finds feasible optimal schedules for the n-products model.

TABLE OF CONTENTS

LIST OF FIGURES

GLOSSARY AND NOTATIONS

$\boldsymbol{a}$	The fundamental cycle of Bomberger's BPM where $\boldsymbol{a} = \boldsymbol{g} = \boldsymbol{g}_n = \boldsymbol{gcd}(T_1, T_2, \cdots, T_n)$.
$\boldsymbol{a}_b^*$	Optimal value of $\boldsymbol{a}$ when $\{\boldsymbol{b}_i\}$ is fixed at $\boldsymbol{b}_i = \boldsymbol{b}_{ai}^*$ all $\boldsymbol{i}$.
$\boldsymbol{a}_{ij}$	The greatest common divisor of cycle times $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$.
$\boldsymbol{a}_l$	Present lower bound on $\boldsymbol{a}$.
$\boldsymbol{a}_u$	Present upper bound on $\boldsymbol{a}$.
$\boldsymbol{b}_{ai}^*$	Optimal integer multiplier of Bomberger's BPM when $\boldsymbol{a} = \boldsymbol{a}_u$.
$\boldsymbol{b}_{arsi}^*$	Unique real-value optimal multiplier when the ***ith*** product is treated as an unconstrained, single-product model with $\boldsymbol{a} = \boldsymbol{a}_u$.
$\boldsymbol{b}_{asi}^*$	Optimal integer multiplier when the ***ith*** product is treated as a constrained, single-product model with $\boldsymbol{a} = \boldsymbol{a}_u$.
$\boldsymbol{b}_i$	Multiplier for the ith product in the BPM, i.e. in the BPM $\boldsymbol{T}_i = \boldsymbol{ab}_i$.

b_{ij}	$T_i \,/\, a_{ij}$.
b_{li}	Present lower bound on the ***ith*** multiplier of the BPM.
b_{ui}	Present upper bound on the ***ith*** multiplier of the BPM.
BPM	Basic period model.
c_i	Setup cost of the ***ith*** item where $c_i > 0$.
C_h	$\sum_{i=1}^{n} c_i$, the setup cost of Hanssman's CCM.
CCM	Common cycle model.
d_i	Delay time of the ***ith*** item, or the length of time from time zero until the start of the first production interval of the ***ith*** item, $T_i > d_i \geq 0$.
$\tilde{d}$	$\{d_i\}$, a vector of n delay times.
EMQ	Economic manufacturing quantities.
$gcd(L_1, \cdots, L_u)$	Greatest common divisor of the integers $L_1, L_2, \cdots, L_u$ where $u \geq 2$.
g	$g_n = gcd(T_1, T_2, \cdots, T_n)$.

g_k	$gcd(T_1, T_2, \cdots, T_k), k = 2,3,\cdots,n.$
GCM	General cyclical model.
h_i	Inventory holding charge per unit time per unit of the ***ith*** item.
H_h	$\sum_{i=1}^{n} H_i$, the inventory cost of Hanssman's CCM.
H_i	$h_i\rho_i(p_i - r_i) / 2.$
n	Number of products to be scheduled on a single facility.
p_i	Production rate of the ***ith*** item where $p_i > 0.$
q_i	Length of a used period of the ***ith*** item where $0 < q_i = [s_i + \rho_i T_i]^+.$
r_i	Demand rate of the ***ith*** item where $r_i > 0.$
s_i	Setup time of the ***ith*** item where $s_i \geq 0.$
S	Sum $\sum_{1}^{n} s_i$ of the setup times of all items where $S > 0.$
$\tilde{T}$	$\{T_i\}$, a vector of n cycle times.

T^*_{asi}	Optimal integer cycle time when the ***ith*** product is treated as a constrained, single-product model with $a = a_u$, i.e. $T^*_{asi} = a_u b^*_{asi}$.
T_{bi}	Cycle times for the ***ith*** item, $i = 1, 2, \cdots, n$, that corresponds to the present best solution found by an iterative procedure.
T_h	Cycle time in Hanssman's CCM.
T^*_h	Optimal cycle time of Hanssman's CCM.
T_i	Production cycle time of the ***ith*** item.
T_{li}	Cycle time for the ***ith*** item, $i = 1, 2, \cdots, n$, that is the present lower bound on T_i found by an iterative procedure.
T^*_{rsi}	Real valued cycle time that is the unique optimal cycle time when the ***ith*** item is treated as the only item of a single-product model.
T^*_{si}	Cycle time that is the optimal integer cycle time when the ***ith*** item is treated as the only item of a single-product model. It is possible that T^*_{si} can equal either one of two values.

T_{ui}	Cycle time for the ***ith*** item, $i = 1, 2, \cdots, n$, that is the present upper bound on T_i found by an iterative procedure.
$\{v_i\}$	Vector of n parameters or variables, e.g. $\{T_i\}$ is a vector of cycle times for the n products model.
x^+	Smallest integer greater than or equal the real number x.
x^-	Greatest integer less than or equal the real number x.
z^*	Optimal value of $Z(\Omega)$.
z_a^*	Optimal cost of BPM when $a = a_u$.
z_{asi}^*	$Z_i(T_{asi}^*)$, the optimal single product cost of the ***ith*** cost component, when $a = a_u$.
z_b^*	Optimal value of $Z(\Omega)$ when $\{b_i\}$ is fixed at $b_i = b_{ai}^*$, all i.
z_h^*	Optimal value of $Z_h(T_h)$.
z_{mi}	Lower bound on optimal cost of the ***ith*** item.
z_{si}^*	Optimal cost when the ***ith*** item is treated as the only item of a single-product integer model.

$Z_b(a)$	Objective function of modified CCM where $\{b_i\}$ is fixed at $b_i = b^*_{ai}$, all i.
$Z_h(T_h)$	Cost function in Hanssman's common cycle model.
$Z_i(T_i)$	Average total cost per unit time for item i when cycle time is T_i.
Z_l	Present lower bound on cost found by an iterative procedure.
Z_u	Present upper bound on cost found by an iterative procedure. This is also the present best cost.
$Z(\Omega)$	Total average cost per unit time for schedule Ω.
β_{ii}	$T_i / (\beta_{ij}\beta_{ik}g_3)$. In a set of three cycle times, (T_i, T_j, T_k), β_{ii} is the quotient of T_i divided by all factors common just to T_i and T_j by all factors common just to T_i and T_k, and by all factors common to all three.
β_{ij}	$a_{ij} / g_3, i \neq j$. In a set of three cycle times, (T_i, T_j, T_k), β_{ij} is the $gcd(T_i / g_3,\ T_j / g_3)$. Note that β_{ij} and T_k / g_3 are relative prime.
ϕ	Null set.

θ	$\{i \mid T_{li} = T_{ui}\}$.
ρ_i	$r_i \,/\, p_i$
Ω	$(\tilde{T}, \tilde{d})$, a schedule of n items. That is, Ω is a n by 2 matrix where the elements of the ***ith*** row are the cycle time and delay time of the ***ith*** item.

Besides this list, notation will be introduced within the remainder of the dissertation. This additional notation will be easier to understand if introduced within the context where it is used.

CHAPTER 1

INTRODUCTION

PROBLEM DEFINITION

Since the introduction of the classical economic manufacturing quantities (EMQ) concept in the early nineteen hundreds, researchers have found solutions for a variety of single product, single facility EMQ models. But, as explained by Salvenson [76] in 1953, by Maxwell [63] in 1964, and, somewhat more recently, by Elmaghraby [20] in 1978, these basic single product models must be extended if the EMQ concept is to render solutions to practical production-inventory problems. The subject of this paper is a logical extension of the fundamental single-product EMQ models; namely, a multi-product, single facility model which is defined by the following assumptions.

MODEL ASSUMPTIONS

a. All model parameters are known with certainty.

b. A set of n products are produced on a single facility in accordance with a cyclical schedule i.e., the production schedule of each product is repeated every T_i time units where T_i is the cycle time of the ***ith*** product, $i = 1, 2, \cdots, n$.

c. There is a positive demand, $\boldsymbol{r_i} > \boldsymbol{0}$, per unit time for product $\boldsymbol{i}$ which remains constant over an infinite planning horizon.

d. The facility produces product $\boldsymbol{i}$ at a constant rate $\boldsymbol{p_i} > \boldsymbol{r_i}$ during each active production interval.

e. All demand must be satisfied and production of a product begins just when the inventory of that product is exhausted. Thus, no surplus inventory or safety stock is maintained for any of the products.

f. Each changeover from producing product $\boldsymbol{i}$ to producing product $\boldsymbol{k}, \boldsymbol{i} \neq \boldsymbol{k}$, requires a setup time, $\boldsymbol{s_k} \geq \boldsymbol{0}$, where $\boldsymbol{s_k}$ is independent of $\boldsymbol{i}$ and does not vary with time.

g. The facility may be occupied with only one product at a time.

h. The demand and production rates of the $\boldsymbol{n}$ products satisfy

$$\sum_{i=1}^{n} r_i / p_i \leq 1; \tag{1}$$

otherwise, no feasible schedule exists. If there is at least one setup time such that $\boldsymbol{s_i} > \boldsymbol{0}$, then (1) must be a strict inequality.

i. To remove the problems inherent in adjusting the production schedule to reflect beginning inventories, it is assumed that production started at time $\boldsymbol{t} \Rightarrow -\infty$.

j. All parameters, except the cost parameters, are expressed as integers by choosing an appropriate time unit.

k. Each changeover from producing product i to producing product $k, i \neq k,$ causes a set-up cost, $c_k > 0,$ where c_k is independent of i. The only cost components are the setup cost c_i and an inventory holding cost, $h_i > 0,$ per unit of product i in inventory per unit time for each product i. These costs do not vary with time.

GRAPHICAL EXAMPLE

The following graphical look at the model for the two-product case is presented to provide insight into the problem. Under the defining assumptions of the model, the inventory level of each product goes through the classical triangular cycle as depicted by Figures 1 and 3.

Figure 2 shows the Gantt chart representation of the production schedule of products 1 and 2 on the facility. Note that the effect of the setup times, s_1 and s_2, is to shift the beginning of the production cycle backwards in time with respect to the inventory cycle. The $\{d_i\}$ are the delay times; or, in other words, the length of time between time zero and the beginning of the first production interval. The production interval, q_i, is the length of time that the facility is occupied with the ***ith*** product. As shown, by figures 1, 2, and 3 this interval is the sum of the setup time and the active product time. Note that $d_i < T_i$ and $q_i < T_i$. The area of the triangle representing each inventory cycle is the total units of products per cycle.

Figure 1
Inventory Cycle for Product 1

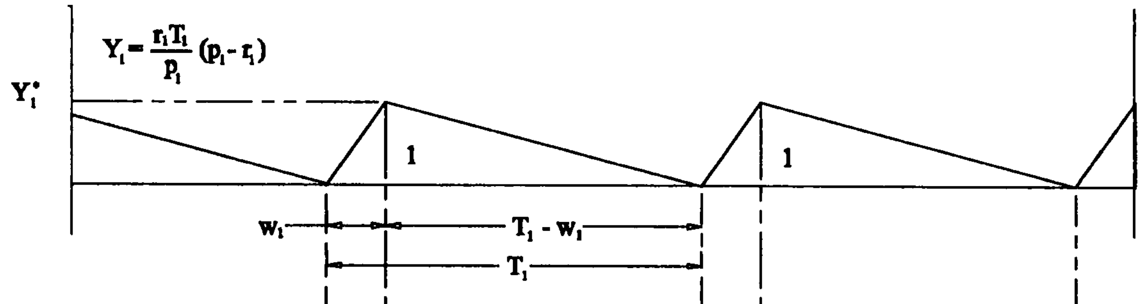


Figure 2
Gantt Chart for Products 1 and 2

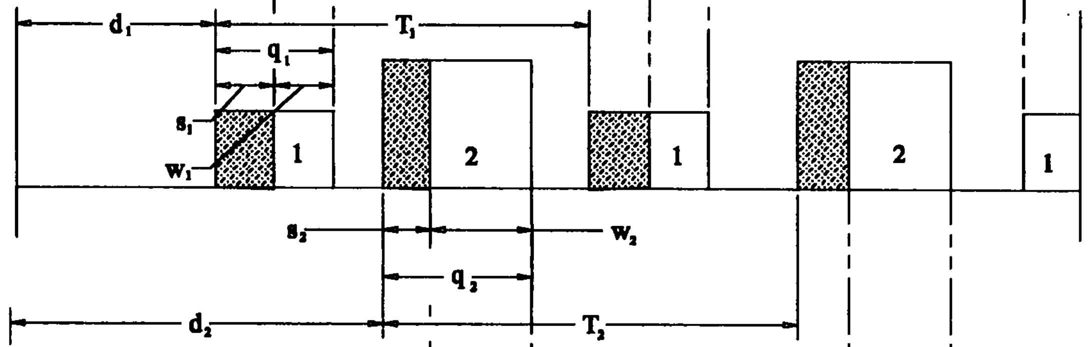
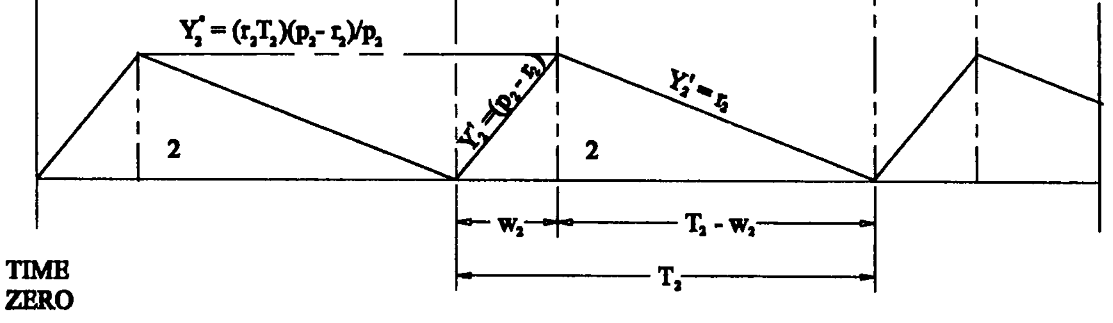


Figure 3
Inventory Cycle for Product 2



$w_1 = r_1T_1/p$, $w_2 = r_2T_2/p_2$, $Y_2' = \frac{dY_2}{d_t}$, Y_i^* = max inventory level for product i.

MODEL DESCRIPTION

The defining assumptions (a) through (k) imply there is an item schedule for each item which is an infinite series of production cycles of fixed length $\boldsymbol{T_i}$. Each cycle of an item schedule is composed of a facility use period, or simply use period, and a non-use period. The use period is further divided into a setup period and a production period. If we let $\boldsymbol{q_i}$ be the length of the use period for item ***i***, it is easily shown that

$$\begin{aligned} q_i &= \left[s_i + \left(r_i / p_i\right)T_i\right]^+ \\ &= s_i + \left[\rho_i T_i\right]^+ \qquad (2) \end{aligned}$$

where

$$[x]^+ = \textit{The smallest integer not less than } x$$

and

$$\rho_i = r_i / p_i.$$

The last form of (2) is valid when s_i, all ***i***, are restricted to integer values.

Because of defining assumption (***i***) we may arbitrarily select time zero. But, once time zero has been selected, the ***ith*** item schedule is fixed with respect to time zero by specifying a delay time; $\boldsymbol{d_i}$, $\boldsymbol{0 \le d_i < T_i}$, where $\boldsymbol{d_i}$ is the length of the time interval between time zero and the start of the first use period of the ***ith*** item. Because of relationship (2), the ***ith*** item schedule is completely determined by specifying values for all model parameters, time zero, and values for the decision variable T_i and d_i.

If the facility produces only one item (i.e., $n = 1$), then the item schedule is also the facility schedule, or simply schedule. But if $n \geq 2$, a schedule is determined by specifying n item schedules and a sequence. Suppose the items are arbitrarily numbered from 1 to n and let λ be a mapping of the indices of the items where λ_i is the ***ith*** item under λ. Then a sequence is a mapping λ such that

$$d_{\lambda 1} \leq d_{\lambda 2} \leq \cdots \leq d_{\lambda n}. \tag{3}$$

Hence, a schedule Ω of n items produced on a single facility is $\Omega = \left(\tilde{T},\ \tilde{d},\ \lambda\right)$ where $\tilde{T},\ \tilde{d}$, are n-element vectors of cycle times and delay times, respectively, and λ is a mapping which determines (3). Actually when values are specified for $\tilde{d}$ a mapping is implicitly determined by the element of $\tilde{d}$. Therefore, without loss of generality a schedule Ω can be specified by $\Omega = \left(\tilde{T},\ \tilde{d}\right)$.

The objective of the model is to find a schedule Ω which minimizes the cost per unit time of producing the n items, $Z(\Omega)$, subject to the restriction that Ω is feasible. As shown by Figures 1 and 3 the graph of the inventory level of each item is an infinite series of triangular curves. Thus, by using simple triangular relationships and remembering the setup costs it is easily shown that:

$$Z(\Omega) = \sum_{i=1}^{n} \left[c_i / T_i + h_i r_i (p_i / r_i) T_i / (2p_i) \right]. \tag{4}$$

A schedule Ω is <u>feasible</u> if

$$0 < d_i \le T_i < \infty; \; i = 1, 2, \cdots, n; \tag{5}$$

and

$$\Omega \text{ is } \underline{\text{non-conflicting}}. \tag{6}$$

A schedule Ω is <u>non-conflicting</u> if there is no time interval, having positive length, in the time domain during which the use periods of two or more items overlap; otherwise, Ω is said to be conflicting. Observe that the use periods of a feasible schedule may coincide at end points. By defining a non-conflicting schedule in this manner, we achieve a more tractable form of later mathematical developments.

Because of assumption ***(k)*** the cost function (4) is a sum of n strictly convex single product cost functions. This objective function is somewhat limiting; however, it is the cost function used by most researchers studying this EMQ model. Using this function permits an easy comparison of the numerical results of this research with that by previous researchers. Actually, most, if not all, of these results are equally applicable given any cost function that is the sum of separable, strictly convex functions of the $\{T_i\}$.

The mathematical concept of the greatest common divisor of integers plays an important role throughout this dissertation. Henceforth, $gcd(x_1, x_2, \cdots, x_k)$ will indicate the greatest common divisor of the integers $(x_1, \cdots, x_k)$.

NECESSARY CONDITIONS

Existence of Feasible Schedules

For a feasible schedule to exist condition (1) must be satisfied. Otherwise, the facility capacity is clearly insufficient to meet the demands for all n items. The question of whether or not there exist a feasible schedule for a specified set of model parameters is an important one. The algorithms of chapters 4 and 9 implicitly presume that at least one solution exists for the problem at hand. If this presumption is incorrect these algorithms may yield meaningless results. Fortunately, in addition to being a necessary feasibility condition, (1) is sufficient to assure that an infinite number of feasible schedules exist. To prove this assertion two cases must be considered: when $s_i = 0$ for all i and when there is at least one i such that $s_i > 0$.

In each case it is presumed that (1) is satisfied and, at first, the integer constraint is relaxed. Then an infinite family of real-valued schedules is constructed. Finally, it is shown that this family of real-valued schedule can be mapped to an infinite family of feasible integer schedules. In both cases, the "trick" of the construction is to find a $\widetilde{T}$ where the T_i are all large enough to accumulate enough slack time to make all required changeovers from producing one product to another.

Suppose $s_i = 0$, all i, and let $T_i = T > 0$ then from (1)

$$T_i \geq \sum_{i=1}^{n} q_i = \sum_{1}^{n} \rho_i T. \tag{7}$$

Thus, each T_i is greater than the sum of the n use periods, so it is easily shown that $\Omega = \left(\tilde{T},\ \tilde{d}\right)$ where

$$d_1 = 0, \text{ and} \tag{8}$$

$$d_k = \sum_{i=1}^{k-1} q_i = \sum_{i=1}^{k-1} \rho_i T \tag{9}$$

is a feasible real-valued schedule. Because $\rho_i = \gamma_i / \rho_i$ setting $T = \prod_{i=1}^{n} p_i$ converts this real-valued schedule to an integer-valued one.

On the other hand, suppose $s_i > 0$ for at least one item i. If (1) is an equality, all of the facility capacity is required just to satisfy the demand for all items. This means there is no time available for setups. Thus, a feasible schedule does not exist if any $s_i > 0$ when (1) is an equality. Now suppose that (1) is a strict inequality. Then it is easily show that $\Omega = \left(\tilde{T},\ \tilde{d}\right)$ is a feasible real-valued schedule when

$$T_i = T \geq \frac{S}{1 - \sum_{1}^{n} \rho_i},\ all\ i, \tag{10}$$

$$d_1 = 0, \tag{11}$$

$$d_k = \sum_{1}^{k-1} q_i = \sum_{1}^{k-1} \left(s_i + \rho_i \cdot T\right), \tag{12}$$

and

$$S = \sum_{i=1}^{n} s_i .$$

Setting $T = m \prod_{i=1}^{n} p_i$ where m is large enough for T to satisfy (10) converts this real-valued schedule to an integer-valued one. Thus, if (1) is satisfied there are an infinite number of integers that satisfy either (7) or (10). Therefore, satisfying (1) is sufficient to assure that an infinite number of feasible integer-valued schedules exist.

Integer Necessary Conditions

When $\tilde{T}$ and $\tilde{d}$ are restricted to integers values it is possible to derive a necessary condition that is somewhat more restrictive than (1). Suppose there are $n \geq 2$ items and let:

$$a = gcd\left(T_1, T_2, \cdots, T_n\right) \tag{13}$$

$$b_i = T_i \,/\, a \tag{14}$$

$$k_i = \prod_{j \neq i}^{n} b_j \tag{15}$$

$$T = \left(\prod_{j=1}^{n} b_j\right) a \tag{16}$$

so that T is a common multiple of the $\left\{T_i\right\}$. The facility schedule during the time interval $\left[0, T\right]$ is identical to the schedule during $\left[mT, (m+1)T\right]$ for any integer $m \geq 1$. Thus, a feasible schedule exists if and only if a feasible schedule exists during the interval $[0, T]$. Note that T is not necessarily the least common multiple (lcm) of the $\{T_i\}$. In general, a period of length T will be a positive multiple of $T_{lcm} = lcm(\{T_i\})$. The facility schedule clearly repeats every T_{lcm} time units. However, this means that, as stated above, the schedule also repeats every T time units. The time period T is used here rather than T_{lcm} because T is more easily defined. Furthermore, the same integer feasibility condition is obtained whether one uses T or T_{lcm}.

For a feasible schedule to exist, T must be greater than the sum of all use periods for all products during the $[0, T]$ interval. It is easily seen from (15) and (16) that there are k_i use periods for item i, all i, during the $[0, T]$ interval. Clearly time period T must be long enough to accommodate all the use periods of all the products that occur during $[0, T]$. These observations lead to the necessary conditions:

$$T \geq \sum_{i=1}^{n} k_i q_i = \sum_{i=1}^{n} k_i (s_i + \rho_i \cdot T_i)^{+}. \tag{17}$$

Inequality (17) immediately leads to an integer necessary condition on a:

$$T = \left(\prod_{j=1}^{n} b_j \right) a \geq \sum_{i=1}^{n} \left[\left(\prod_{j \neq i}^{n} b_j \right) s_i \right] + \sum_{i=1}^{n} \left[\left(\prod_{j \neq 1}^{n} b_j \right) (\rho_i b_i a)^{+} \right] \tag{18}$$

or

$$a \geq \sum_{i=1}^{n} q_l / b_l. \tag{19}$$

HISTORY OF THE PROBLEM AND PREVIOUS RESEARCH

The subject of this research is a restricted version of the general deterministic, multi-products EMQ model. The model being studied differs from the general multi-products model with regard to two characteristics:

1. Once a facility schedule is set, each item is produced according to a cyclical schedule of fixed length, and

2. Production is started only when inventory reaches zero.

Restriction 1 is equivalent to the equal-lot assumption because once the cycle time is fixed the lot size is fixed via (2). In the literature restriction 2 is referred to as the zero-switch rule. For reference purposes the subject of this research is called the general cyclical model (GCM). The GCM label is used to distinguish the model defined by assumptions (a) through (k) from the general multi-products EMQ model and from models with additional constraining assumptions.

Rogers [73] gave the first formal computational statement of the GCM. As shown by the list of references in the last chapter this model and several derivative models have been the subject of much research over the intervening four decades. As Elmaghraby [20] explained in 1978, the approaches of early research on the GCM could be divided into two categories: developing analytical solutions to a restricted version of the GCM, e.g. Bomberger [4], Hanssman [42], and Leachman, Xiong, Gascon, and Park [58], or developing heuristics that yielded good solutions for the GCM, e.g. Davis [10] and [11], Ditt and Kuhn [13], and Geng and Vickson [29]. More recently there are at least two additional categories of research: estimating the complexity of the GCM, e.g. Gallego and Shaw [26] and Hsu [47], and determining feasibility conditions, e.g. Glass [12], Gray [36], Park and Yun [69], and Vermuganti [80].

Elmaghraby [20] and Narro Lopez and Kingsman [67] give good reviews of the GCM research and the approaches used to simplify the model. Bourland and Yano [5] provide a good discussion of the GCM and the effects of restrictions 1 and 2 and other restrictions that are often imposed on the GCM in the introduction to their work on the general multi-products model. Gallego and Shaw [26] show that the GCM and other restricted forms of the general multi-products EMQ model are "NP-hard in the strong sense".

The restricted GCM's most often studied are the common cycle model (CCM) introduced by Hanssman [42] and the basic period model (BPM) introduced by Bomberger [4]. Hanssman's CCM greatly simplifies the GCM by imposing the additional restriction that $\boldsymbol{T_i} = \boldsymbol{T}$, all $\boldsymbol{i}$. He then uses standard techniques of calculus and a single feasibility condition to find an optimal $\boldsymbol{T}$.

To define the BPM Bomberger [4] imposes two additional restrictions on the GCM:

$$\boldsymbol{T_i = a_r \cdot b_i,\ i = 1,\ 2,\ \cdots,\ n} \tag{20}$$

and

$$\left[\sum_{i=1}^{n} a_r \cdot b_i \cdot \rho_i + s_i\right] \le a_r \tag{21}$$

where the $\{b_i\}$ are restricted to integer values but a_r is not. Bomberger called a_r the fundamental cycle of the schedule. He did not restrict a_r to integer values. However, in the BPM algorithm described in chapter 4 the fundamental cycle is not only restricted to integer values it is also restricted to be the greatest common divisor of the cycle times, a. The greatest common divisor restriction does not impose additional constraints on the algorithm beyond that required by the integer restrictions. To see this, suppose there is an integer factor $a_I < a$. Clearly, if a_I satisfies (21) then neither on the other hand, if a does not satisfy (21) then neither does any other common factor of the cycle times. Therefore, a common factor of $\{T_i\}$ satisfies (21) if and only if a does.

Note that Bomberger's multiples b_i are not quite the same as the factors defined by (14). However, one a is restricted to be the greatest common divisor of the cycle times in the BPM algorithm, Bomberger's multiples and the factors defined by (14) become the same factors. Thus, Bomberger's multiples are denoted by $\{b_i\}$ to maintain consistence notation. Hopefully, this will not be confusing to the reader.

Bomberger shows that, if (20) and (21) are satisfied, a feasible schedule exists. It can be shown for both Hanssman's and Bomberger's models that, if $\tilde{T}$ satisfies the conditions of their models and if the elements of $\tilde{d}$ are set to

$$d_1 = 0$$

$$d_j = \sum_{i=1}^{j-1} q_i, \; j = 2, 3, \cdots, n$$

then $\Omega = (\tilde{T}, \tilde{d})$ is a feasible schedule. Hence, in these models the problem of finding a feasible schedule is reduced to finding feasible cycle times.

Bomberger proposes a straight-forward dynamic programming procedure for finding $\{b_i^*\}$ for a fixed a. The state variables of his formulation are restricted to the discrete values $0, \Delta a_r, 2\Delta a_r, \cdots, a_r$ where Δ is arbitrarily chosen. Bomberger suggests that a_r be determined by trial and error. There is a major problem with this trial and error procedure. To describe this problem let z_a^* be the optimal cost over the $\{b_i\}$ found by Bomberger's dynamic programming procedure when a is fixed at some specified value. In general, z_a^* will change if a sufficiently large change is made to a. That is, z_a^* is a function of a. However, Hodgson [43] shows that z_a^* is not an unimodal function of a_r. In fact, for the integer model, used in chapter 4, z_a^* is best described as an erratic function of a. Hence, in general, Bomberger's trial and error method of determining a_r does not guarantee that an optimal or a near-optimal schedule will be found for the BPM.

CHAPTER 2

SUMMARY

The motivation for this research concerning the *n*-product GCM comes from personal experience in large, complex manufacturing plants and a chance reading of Bomberger's [4] article that describes his BPM. Within a manufacturing plant there are usually a few critical facilities. In large measure, these critical facilities determine the schedule and productivity of the entire plant.

My archetype for the GCM is a large 7,500 ton forming and forging press located in a plant where I worked. This press is three stories high, expensive to operate and a huge capital investment. Changing the setup of this press requires from a few hours to more than a day to complete. There are several other critical facilities in this plant; however, most of the plant moves to the rhythm of this large press. An algorithm that can be used to periodically find optimal, or near optimal, schedule for this critical press should significantly reduce the manufacturing costs of the entire plant.

The subject of the research of this dissertation is primarily the general cyclical model of a single-facility producing *n*-products. Many would argue that the GCM is an artificial restriction on the general multi-products, single-facility scheduling problem. On the other hand, by considering the real world settings of most facilities, one can argue just as forcefully that the general single-facility, multi-product problem is, in many cases, an unnatural extension of the GCM. Many, if not most, activities within a manufacturing plant occur on a more or less cyclical basis, e.g. parts are produced in "campaigns", machine maintenance is performed every Monday, and stock is inventoried monthly.

Concisely stated, the goal of this research is to develop an algorithm that will find an optimal cyclical schedule for actual multi-products, single-facility operations. Furthermore, to be of practical value, this algorithm must be efficient. The algorithm should be capable of scheduling a facility that is producing at least ten products with a reasonable expenditure of computer resources.

To achieve this goal one must first develop a set of mathematically necessary and sufficient conditions that identifies feasible schedules. This is a critically important task because without such conditions it is not possible to determine whether or not a given schedule is feasible. Without such conditions it is not possible to create an optimization algorithm. The primary results of chapter 3 fulfilled this need by providing conditions that are both necessary and sufficient to assure that a given schedule is feasible. Bomberger [4] provides a partial solution to the basic period model (BPM). The BPM is a restricted version of the GCM that restricts all cycle times to multiples of a common basic period. Bomberger gives a dynamic programming formulation that solves for optimal multipliers when the common basic cycle is a specified value. However, his algorithm does not find a global optimal for the BPM over all possible values of the basic period. The algorithm developed in chapter 4 finds an optimal BPM schedule over all variables. The results of chapter 4 are quite important to the goal of developing an efficient GCM. As shown by the numerical results of chapter 10, using the optimal cost of the BPM as an initial upper bound for the GCM algorithm will, in most cases, greatly reduce the computational effort required to find an optimal GCM schedule. Furthermore, in some practical situations an optimal BPM will suffice.

Chapters 5, 7, and 8 study the two-products, three-products, and four-products, respectively, cases of the GCM. These chapters provide a simpler feasibility conditions for these limited versions of the GCM. These limited models are academically interesting. For instance, chapter 5 shows that when $n = 2$ the BPM algorithm of chapter 4 will find optimal GCM schedules. Furthermore, each of these limited models provides important mathematical "filters" that allows the GCM algorithm to recognize many infeasible schedules early in the schedule generation process. This greatly improves the efficiency of the GCM algorithm.

The study of the n-products model in chapter 6 provides a number of results that are used in the proofs related to the three-products and four-products models of chapters 7 and 8, respectively. In addition, theorem 5 on the feasibility of partial schedules in chapter 6, in conjunction with the two-products, three-products and four-products results, provides the theoretical foundation for the mathematical "filters" of the GCM. The GCM algorithm presented in chapter 9 is the culmination of the results of the previous chapters. The GCM algorithm is a customized n-stage implicit enumeration scheme. That is, the research in chapter 9 takes advantage of the characteristics of the cost objective functions and the results of the previous chapters to develop an efficient GCM algorithm.

An abbreviated version of the GCM algorithm was coded in a compiled version of the basic programming language. This computer program was used to solve three typical problems taken from the literature. These numerical results are reported in chapter 10.

As expected, the cost of the optimal schedule generated by the GCM algorithm are lower than that provided by Bomberger's BPM. The improvements over Bomberger's results are significant in those cases where facility utilization is moderate to high.

As shown in chapter 10, the upper bounds obtained from the BPM of chapter 4 greatly improves the efficiency of the GCM algorithm. It is expected that, as the number of products increases, the BPM upper bounds will, in most cases, be of even greater benefit to the efficiency of the GCM. This expectation is supported by the result reported in chapter 10. The results for Bomberger's problem 2 reported in chapter 10 is an extreme illustration of the benefits obtained from having the BPM optimal cost as the initial upper bound for the GCM.

The GCM algorithm satisfies the primary goal of this research. As demonstrated by the numerical results reported in chapter 10, the GCM algorithm will generate optimal schedules for practical problems. Furthermore, it does so with a reasonable expenditure of computer resources. Of course, the results of chapter 10 does not mean that the GCM algorithm will efficiently solve every ***n***-products problem. Gallego and Shaw [26] show that the GCM is "NP-hard in the strong sense". Therefore, there will almost certainly be practical ***n***-products problems that require exorbitant amounts of computer resources to solve. Nevertheless, a polished computer program of the GCM algorithm that incorporates all the improvements discussed in chapter 9 should solve many practical instances of the cyclical multi-products, singe-facility scheduling problem.

CHAPTER 3

NECESSARY AND SUFFICIENT FEASIBILITY CONDITIONS

BACKGROUND

As discussed in chapter 1 the GCM has been studied by many authors during the last four decades. For the most part, these authors have directed their research either toward developing heuristic solution procedures or toward finding optimal solutions to models created by further restricting the GCM. The restricted scope of previous research is understandable because the complexity of the inventory problem increases dramatically as one moves from the single product EMQ model to the scheduling of two or more items on a single facility. Furthermore, the lack of conditions which are both necessary and sufficient to assure schedule feasibility has been major handicap suffered by all previous researchers. That handicap is removed by the results of this chapter.

The principle result of this chapter is proof of a general theorem on schedule feasibility. This theorem gives conditions which are both necessary and sufficient to assure that a schedule of n items on a single facility is feasible when $n \geq 2$. The theorem is quite general with respect to the multi-products model defined in the introduction chapter. The hypotheses of the general theorem is the defining assumptions (a) through (j). Assumption k is concerned with costs and does not effect schedule feasibility. In particular, the hypotheses requires that the necessary condition (1) given in chapter 1 be satisfied. As shown in the introduction, if this condition is satisfied then there exists an infinite set of feasible solutions; otherwise, no feasible solution exists.

In spite of the considerable research efforts cited in chapter 1, an algorithm which, in general, yields optimal, feasible schedules for the model defined by assumptions (a) through (k) has not been developed until now. The results of this chapter are an essential step toward that end. Except for chapter 4, the necessary and sufficient conditions for schedule feasibility given here are the foundation for the results of the remaining chapters. These feasibility conditions are in terms of $\left(\tilde{T}, \tilde{d}\right)$. That is, for a given set of model parameters a given schedule $\left(\tilde{T}, \tilde{d}\right)$ is feasible if and only if these conditions are satisfied.

THEOREM 1 – GENERAL THEOREM ON FEASIBILITY

Let $\Omega = \left(\tilde{T},\ \tilde{d}\right)$ be a schedule of n items on a single facility. Further, define a_{ij}, b_{ij} and b_{ji} as the integers that satisfy

$$T_i = a_{ij} b_{ij} \text{ and } T_j = a_{ij} b_{ji} \tag{22}$$

where

$$a_{ij} = a_{ji} = \text{gcd}\left(T_i,\ T_j\right) \tag{23}$$

and $i,\ j = 1,\ 2,\ \cdots,\ n,\ i \neq j.$

Suppose the items are labeled so that $d_i \geq d_j$ and let

$$d_i - d_j = k_{ij}a_{ij} + l_{ij} \tag{24}$$

where

$$0 \le l_{ij} < a_{ij} \tag{25}$$

and $a_{ij}, b_{ji}, b_{ij}, k_{ij}$ *and* l_{ij} are integers. Suppose that the defining assumptions (a) through (j) as given in chapter 1 are valid. Then Ω is feasible if, and only if for each pair of items i and j,

$$l_{ij} \ge q_j \tag{26}$$

and

$$l_{ij} \le a_{ij} - q_i \tag{27}$$

where $i, j = 1, 2, \cdots, n,$ *and* $i \ne j.$

REMARKS CONCERNING THEOREM 1

Equations (22) and (23) imply that b_{ij} and b_{ji} are relative prime or, equivalently, that

$$\gcd\left(b_{ij},\ b_{ji}\right) = 1. \tag{28}$$

Because $d_i - d_j$ is a non-negative integer and $a_{ij} \ge 1$ the division algorithm states that this difference can always be expressed as shown by (24) and (25) with $k_{ij} \ge 0$ furthermore, k_{ij} and l_{ij} are unique for a given d_i, d_j, and a_{ij}. Assumptions (a) through (j) assure that $q_i > 0$ so $l_{ij} > 0$ if (26) is satisfied. The possibility of equality in (26) and (27) allows conflicts of non-zero length to occur. Suppose that q_i, q_j are not required to be integers, but a_{ij} and l_{ij} are integers. If (26) and (27) are satisfied with non-integer q_i then so are

$$l_{ij} \geq q_j^+ \tag{29}$$

and

$$l_{ij} \leq a_{ij} - q_i^+. \tag{30}$$

Therefore, if $\{a_{ij}\}$ and $\{b_{ij}\}$ are required to be integers then requiring the $\{q_i\}$ to be integers as done by (2) does not further constrain the GCM.

PROOF OF GENERAL THEOREM 1

The approach used to prove the general theorem is to first prove an intermediate theorem. Then the proof of two ensuing lemmas quickly leads to the proof of the general theorem. But, first an expression for an arbitrary production interval of any item i is needed. Referring to Figure 2, it is clear that the facility is occupied with the ***ith*** product during all time intervals

$$m_iT_i + d_i \leq t \leq m_iT_i + d_i + q_i, m = 0, 1, 2, \cdots. \tag{31}$$

We also require the following definition.

Definition of $IF(\bullet)$

The function

$$IF(a / b) = a / b - (a / b)^{-}$$

where a and b are any real numbers such that $a \geq 0$ and $b > 0$.

Theorem 2

A schedule $\Omega = (\tilde{T}, \tilde{d})$ for a facility producing $n \geq 2$ items is a feasible cyclical schedule, if and only if, for each pair of items i and j, $i \neq j$,

$$d_i > d_j \text{ or } d_j > d_i \tag{32}$$

and

$$min_{m_i \in M_i} IF\left[\left(m_i T_i + d_i - d_j\right) / T_j\right] T_j \geq q_j \tag{33}$$

and

$$min_{m_j \in M_j} IF\left[\left(m_j T_j + d_j - d_i\right) / T_i\right] T_i \geq q_i \tag{34}$$

where $M_i = \left\{m_i \text{ such that } m_i T_i + d_i - d_j \geq 0\right\}$.

Proof of Theorem 2

As shown by Figure 4, conflicts between products occur if and only if there are products ***i, j*** such that for some $m_i, m_j \geq 0$

$$m_jT_j + d_j \leq m_iT_i + d_i \tag{35}$$

and

$$m_jT_j + d_j + q_j > m_iT_i + d_i. \tag{36}$$

Of course, (35) and (36) could hold for any number of products ***j*** at the same production interval of product ***i***, but we may treat each pair of products separately to show that if (32), (33) and (34) hold for all pairs then no schedule conflicts exist.

Let ***i, j*** be any two of the ***n*** products to be scheduled. Suppose (32) does not hold, then $d_i = d_j$, but $q_i > 0$ and $q_j > 0$ so the first production intervals of the ***ith*** and ***jth*** products conflict. Therefore, (32) must hold in a non-conflicting schedule. Without loss of generality, assume that $d_i > d_j$. Now that we have chosen a particular pair of products, there are two forms of conflicts as shown by Figures 4 and 5. These have been labeled as ***j***-leading edge conflicts and ***i***-leading edge conflicts.

Suppose (32) and (33) are satisfied. Considering the ***i***-leading edge conflicts first. We know $m_iT_i + d_i - d_j \geq 0$ for all $m \geq 0$ because $d_i > d_j$. Thus, $M_i = \left\{m_i \text{ such that } 0 \leq m_i \leq \infty\right\}$, so $IF\left[\left(m_iT_i + d_i - d_j\right)/T_j\right]$ is well defined

i-Leading Edge Conflict

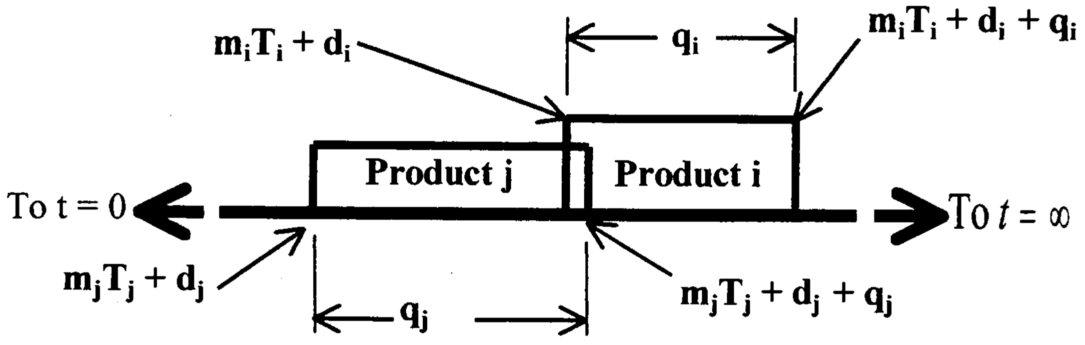


Figure 4

j-Leading Edge Conflict

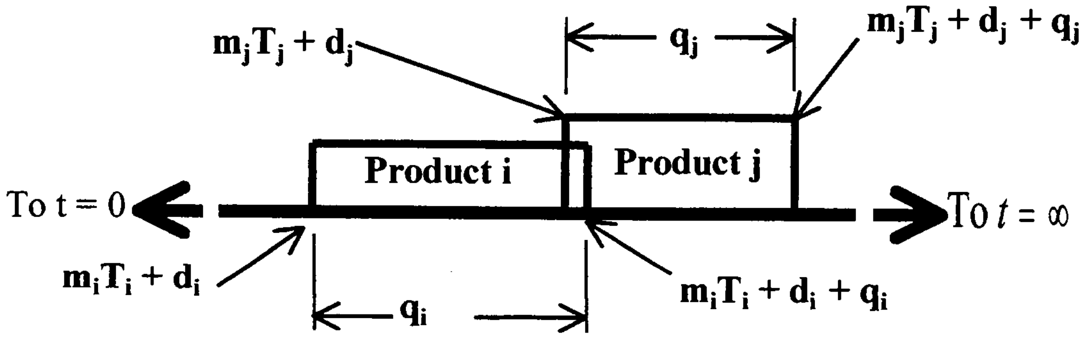


Figure 5

for all m_i, $0 \leq m_i < \infty$ Then let m_i' be any non-negative integer and let m_j' be the maximum m_j such that (35) holds when $m_i = m_i'$.

Clearly $m_j' = \left[\left(m_i'T_i + d_i - d_j\right) / T_j\right]^-$ so consider the difference

$$\begin{aligned}
D_i &= m_i'T_i + d_i - \left(m_j'T_j + d_j\right) \\
&= m_i'T_i + d_i - d_j - \left[\left(m_i'T_i + d_i - d_j\right) / T_j\right]^- T_j \\
&= \left\{\left[\left(m_i'T_i + d_i - d_j\right) / T_j\right] - \left[\left(m_i'T_i + d_i - d_j\right) / T_j\right]^-\right\} T_j \\
&= IF\left[\left(m_i'T_i + d_i - d_j\right) / T_j\right] T_j \\
&\geq min_{m_i \in M_i}\, IF\left[\left(m_iT_i + d_i - d_j\right) / T_j\right] T_j .
\end{aligned}$$

But, because of (33)

$$D_i \geq q_j ,$$

hence,

$$m_i'T_i + d_i \geq m_j'T_j + d_j + q_j > m_jT_j + d_j + q_j \qquad (37)$$

where $0 \leq m_j < m_j'$. Relation (37) shows that the m_i' production interval of the ***ith*** product does not conflict with any production interval of the ***ith*** product which begins prior to $m_i T_i + d_i$. Because m_i' can take on any value $0 \leq m_i < \infty$, this shows that no *i*-leading edge conflicts can occur if (32) and (33) are true.

Now to consider the *j*-leading edge conflict. Suppose that (32) and (34) are satisfied. If $m_j T_j + d_j < d_i$, then $IF\left[\left(m_j T_j + d_j - d_i\right) / T_i\right]$ is ill-defined. But (37) shows that no conflicts occur between the ***ith*** and ***jth*** products in the time interval $0 \leq t \leq d_i$. Therefore, only those production intervals of the ***jth*** product such that

$$m_j T_j + d_j > d_i \tag{38}$$

can be involved in a *j*-leading edge conflict. Noting that

$$M_j = \left\{ m_j \textit{ such that } m_j T_j + d_j > d_i \right\},$$

we can repeat the argument that proved (37) to show that for any m_j' satisfying (38)

$$m_j' T_j + d_j \geq m_i T_i + d_i + q_i \tag{39}$$

for all m_i such that

$$m_j' T_j + d_j \geq m_i T_i + d_i.$$

This shows that no *j*-leading edge conflicts occur if (32) and (34) hold. Hence, the schedules of the ***ith*** and ***jth*** products do not conflict. Because ***i and j*** are any pair of the ***n*** products to be scheduled such that $i \neq j$, the preceding argument shows that (32), (33) and (34) are sufficient conditions for schedule feasibility.

We have shown that (32) is a necessary condition for schedule feasibility, so suppose (33) does not hold for all $\boldsymbol{i, j}$. Then, there are some $\boldsymbol{i, j}$ and $\boldsymbol{m_i' \in M_i}$ such that

$$\boldsymbol{IF\left[\left(m_i'T_i + d_i - d_j\right) / T_j\right]T_j < q_i}. \tag{40}$$

Defining $\boldsymbol{m_i'}$ and $\boldsymbol{D_i}$ in the same manner used to prove (37), we can show that

$$\begin{aligned}\boldsymbol{D_i} &= \boldsymbol{m_i'T_i + d_i - \left(m_j'T_j + d_j\right)} \\ &= \boldsymbol{IF\left[\left(m_i'T_i + d_i - d_j\right) / T_j\right]T_j}.\end{aligned}$$

But by (40)

$$\boldsymbol{D_i < q_i}$$

so

$$\boldsymbol{m_i'T_i + d_i \geq m_j'T_j + d_j}$$

and

$$\boldsymbol{m_i'T_i + d_i < m_j'T_j + d_j + q_j}$$

which is a $\boldsymbol{i}$-leading edge conflict of non-zero length as defined by (35) and (36). Therefore, (33) is a necessary condition for schedule feasibility. A parallel proof shows that (34) is also a necessary condition to avoid j-leading edge conflicts. This completes the proof of Theorem 1.

Integer Constraint on Theorem 1

Note that theorem 2 did not require the T_i, d_i to be integers, but the next two lemmas do. Because these lemmas are required to prove the general theorem, this imposes the integer constraint on general theorem 1.

Lemma 1

If T_i and d_i, $1 \leq i \leq n$, are integers satisfying conditions (32), (33), and (34) of theorem 2, then $gcd\left(T_i, T_j\right) \neq 1$ for any i, j such that $i \neq j$. Furthermore, if $gcd\left(T_i, T_j\right) = 1$ for any T_i, T_j then there are no d_i, d_j such that the three conditions of theorem 2 are satisfied.

Proof of Lemma 1

Let T_i, T_j be any pair of the n items and without loss of generality assume $d_i > d_j$, so $d_i - d_j \geq 1$. Suppose $gcd\left(T_i, T_j\right) = 1$, then a well known result of algebra states that there is a pair of integers x, y such that

$$xT_i + yT_j = gcd\left(T_i, T_j\right) = 1. \tag{41}$$

Let w be any integer, then the addition of (41) to the equation

$$wT_iT_j - WT_iT_j = 0, \tag{42}$$

gives the sum

$$y'T_j = -x'T_i + 1 \tag{43}$$

where $x' = \left(x + wT_j\right)$ and $y' = \left(y - wT_i\right)$. Now $T_i \geq 1,\ T_j \geq 1$ so by the proper choice of w, we have that $x' \leq -1$ and $y' \geq 1$. Now multiplication of both sides of (43) by $\left(d_i - d_j\right)$ gives

$$\left(d_i - d_j\right)y'T_j = -\left(d_i - d_j\right)x'T_i + \left(d_i - d_j\right). \tag{44}$$

Dividing both sides of (44) by T_j gives

$$V = \left(\tilde{m}_i T_i + d_i - d_j\right) / T_j \tag{45}$$

where $V, \tilde{m}_i$ are integers such that $V = \left(d_i - d_j\right)y' \geq 1$ and $\tilde{m}_i = -\left(d_i - d_j\right)x' \geq 1$. This means that T_j divides the right hand side of (44) so

$$min_{m_i \in M_i} IF\left[\left(m_i T_i + d_i - d_j\right) / T_j\right]T_j \leq IF(V)T_j = 0 < q_i.$$

Thus, by contradiction if (32), (33) and (34) hold then $gcd\left(T_i, T_j\right) \neq 1$. Clearly, if $gcd\left(T_i, T_j\right) = 1$ for some T_i, T_j then either $d_i = d_j$ or (33) and (34) are invalid for that T_i, T_j thus, the proof of Lemma 1 is complete.

Observe that if $gcd\left(T_i, T_j\right) \neq 1$ then $gcd\left(T_i, T_j\right) \geq 2$.

Lemma 2

If $T_i, d_i,\ 1 \le i \le n,$ are integers satisfying the conditions of theorem 2 and conditions (22) through (25) of theorem 1 where $a_{ij}, b_{ij}, k_{ji}, k_{ij},$ *and* l_{ij} are integers then for any T_i, T_j and d_i, d_j such that $d_i > d_j$

$$\min_{m_i \in M_i} IF\left[\left(m_i T_i + d_i - d_j\right) / T_j\right] = l_{ij} / T_j \tag{46}$$

and

$$\min_{m_j \in M_j} IF\left[\left(m_j T_j + d_j - d_i\right) / T_i\right] = \left(a_{ij} - l_{ij}\right) / T_i. \tag{47}$$

Proof of Lemma 2

Lemma 1 shows that if conditions (22) and (23) of theorem 1 and the conditions of theorem 2 are valid then $a_{ij} \ge 2.$ Furthermore, equations (22) and (23) imply that $gcd\left(b_{ij}, b_{ji}\right) = 1.$ Considering condition (46) first, we have

$$\begin{aligned}\left(m_i T_i + d_i - d_j\right) / T_j &= \left(m_i a_{ij} b_{ij} + k_{ij} a_{ij} + l_{ij}\right) / a_{ij} b_{ji} \\ &= \left(m_i b_{ij} + k_{ij}\right) / b_{ji} + \left(l_{ij} / a_{ij}\right) / b_{ji}.\end{aligned} \tag{48}$$

Note that $k_{ij} \ge 0$ and $m_i b_{ij} \ge 0$ for $m_i \in M_i$ so $\left(m_i b_{ij} + k_{ij}\right) \ge 0$ for $m_i \in M_i$ Because $\left(m_i b_{ij} + k_{ij}\right) \ge 0$ and $gcd\left(b_{ij}, b_{ji}\right) = 1,$ by using the division algorithm it can be shown that for all $m_i \in M_i$

$$\left(m_i b_{ij} + k_{ij}\right) / b_{ji} = V_i + \left(w_i / b_{ji}\right) \tag{49}$$

where V_i is a non-negative integer and $w_i < b_{ji}$ and $w_i \in Q_i = \left(0, 1, 2, \cdots, b_{ji} - 1\right)$. Furthermore, if $x \in Q_i$ then $w_i = x$ for some $m_i \in M_i$ but if $x \notin Q_i$ then there is no m_i such that $w_i = x$. These last assertions follow quickly once the division algorithm is applied to $\left(m_i b_{ij} + k_{ij}\right) = V b_{ji} + w_i$ and it is remembered that b_{ij}, b_{ji} are relative prime so that there exists an x and y such that $x b_{ij} - y b_{ji} = 1$.

Noting that $0 \le \left(l_{ij} / a_{ij}\right) < 1$, we see from (49) that (48) has the form

$$\left(m_i T_i + d_i - d_j\right) / T_j = V_i + \left[w_i + \left(l_{ij} / a_{ij}\right)\right] / b_{ji}$$

where w_i ranges over the members of Q_i as m_i ranges over M_i. Hence,

$$IF\left[\left(m_i T_i + d_i - d_j\right) / T_j\right] = IF\left\{\left[w_i + \left(l_{ij} / a_{ij}\right)\right] / b_{ji}\right\}.$$

Because w_i and l_{ij} are non-negative and $min\left(w_i\right) = 0$ if follows that

$$min_{m_i \in M_i} IF\left[\left(m_i T_i + d_i - d_j\right) / T_j\right] = \left(l_{ij} / a_{ij}\right) / b_{ji} = l_{ij} / T_j$$

which proves (46).

Considering (47), we have for $m_j \in M_j$

$$\begin{aligned}\left(m_jT_j + d_j - d_i\right) / T_i &= \left[m_jT_j - \left(d_i - d_j\right)\right] / T_i \\ &= \left[m_ja_{ij}b_{ji} - \left(k_{ij}a_{ij} + l_{ij}\right)\right] / a_{ij}b_{ij} \\ &= \left(m_jb_{ji} - k_{ij} - 1\right) / b_{ij} + \left[\left(a_{ij} - l_{ij}\right) / a_{ij}\right] / b_{ij} .\end{aligned} \tag{50}$$

Suppose $\left(m_jb_{ji} - k_{ij} - 1\right) < 0,$ then $\left(m_jb_{ji} - k_{ij} - 1\right) \leq -1,$ but if condition (33) of theorem 2 holds for i then (46) shows that $l_{ij} \geq q_j > 0$ so $l_{ij} \geq 1.$ Then $0 < \left(a_{ij} - l_{ij}\right) / a_{ij} < 1$ so that the right-hand side of (50) is less than zero because $b_{ij} > 0.$ But this implies that $\left(m_jT_j + d_j - d_i\right) < 0$ for $m_j \in M_j,$ so by contradiction $m_ib_{ji} - k_{ij} - 1 \geq 0$ if (33) holds for $i.$ Hence, it can be shown that

$$\left(m_jb_{ji} - k_{ij} - 1\right) / b_{ij} = V_i + \left(w_j / b_{ij}\right)$$

where V_i is a non-negative integer and $w_j < b_{ij}$ and $w_j \in Q_i = \left(0,\ 1,\ 2,\ \cdots, b_{ij} - 1\right).$ Furthermore, if $x \in Q_j$ then $w_j = x$ for some $m_j \in M_j,$ but if $x \notin Q_j$ then there is no m_j such that $w_j = x.$ Hence, as for condition (46),

$$IF\left[\left(m_jT_j + d_j - d_i\right) / T_i\right] = IF\left\{\left[w_j + \left(a_{ij} - l_{ij}\right) / a_{ij}\right] / b_{ij}\right\}$$

and, because $min\left(w_j\right) = 0,$

$$min_{m_j \in M_j} IF\left[\left(m_jT_j + d_j - d_i\right)T_i\right] = \left(a_{ij} - l_{ij}\right) / a_{ij}b_{ij} = \left(a_{ij} - l_{ij}\right) / T_i$$

which proves (47) and completes the proof of Lemma 2.

Observe that it is possible $a_{ij} = a'_{ij}a''_{ij}$ where $1 < a'_{ij}, a''_{ij} < a_{ij}$. Then the effects of factoring T_i and T_j such that $T_i = a'_{ij}a''_{ij}b_{ij}$ and $T_j = a'_{ij}a''_{ij}b_{ji}$ needs consideration. Applying the division algorithm, $d_i - d_j = k'_{ij}a'_{ij} + l'_{ij}$ where $0 \le l'_{ij} < a'_{ij} < a_{ij}$. It appears that this factorization may lead to a smaller value of

$$IF\left[\left(m_iT_i + d_i - d_j\right) / T_j\right] = IF\left[\left(m_iT_i + k'_{ij}a'_{ij} + l'_{ij}\right) / T_j\right].$$

But $k_{ij}a_{ij} + l_{ij} = d_i - d_j = k'_{ij}a'_{ij} + l'_{ij}$ so with either factorization, the $IF(\bullet)$ function yields the same value. Hence, only the factorization $T_i = a_{ij}b_{ij}$ where $a_{ij} = gcd\left(T_i, T_j\right)$ needs to be considered.

By comparing conditions (46) and (47) of Lemma 2 with conditions (33) and (34) of theorem 2, one sees that if $T_i, d_i, 1 \le i \le n$ are integers then Lemmas 1 and 2 prove that the conditions of the general theorem 1 are necessary and sufficient to assure the conditions of theorem 2 are valid. Because theorem 2 conditions are necessary and sufficient to assure schedule feasibility, this completes the proof of the general theorem 1.

NUMERICAL EXAMPLE OF FEASIBILITY CONDITIONS

Consider the two item inventory problem with parameters $r_1 = 8, p_1 = 32, s_i = 8, h_i = 10, c_1 = 122, 880, r_2 = 2, p_2 = 48, s_2 = 4, h_2 = 5, c_2 = 44, 160.$ By the square room EMQ formula the optimal single-product cycle time for each of the two items are:

$$T^*_{rs1} = T^*_{s1} = \left[2c_1 / h_1 r_1 (1 - r_1 / p_1)\right]^{1/2} = (4096)^{1/2} = 64$$

and

$$T^*_{rs2} = T^*_{s2} = \left[2c_2 / h_2 r_2 (1 - r_2 / p_2)\right]^{1/2} = (9216)^{1/2} = 96.$$

Then $T^*_{s1} = 32(2)$ and $T^*_{s2} = 32(3)$ so $a_{12} = a_{21} = 32, b_{12} = 2$ and $b_{21} = 3.$ Set $d_2 = 0$ then $d_1 \geq d_2.$ To check compliance with (26) and (27) of the general theorem 1 observe that

$$l_{12} \geq q_2 = s_2 + (r_2 / p_2)T^*_{s2} = 4 + (2 / 48)96 = 8$$

$$l_{12} \leq a_{12} - q_2 = 32 - s_1 - (r_1 / p_1)T^*_{s1} = 32 - 8 - (8 / 32)64 = 8$$

so l_{12} must equal 8 to satisfy conditions (26) and (27). Then

$$d_1 - d_2 = k_{12}a_{12} + l_{12} = 32k_{12} + 8 \leq T^*_{s1} = 64.$$

Thus if the variables of the model are set to $T^*_{s1} = 64, T^*_{s2} = 96, d_2 = 0,$ with either $d_1 = 8$ or $d_1 = 40$ then all conditions of the general theorem 1 are satisfied. These values also give an optimal schedule because there exist a feasible schedule $\Omega = \{T^*_{si}, d^*_i\}$ when the cycle times are set to the optimal single-product EMQ cycle times. From (4) the optimal cost is

$$
\begin{aligned}
Z^* &= (1/2)h_1 r_1 T_{s1}^*(1 - r_1/p_1) + c_1/T_{s1}^* \\
&\quad + (1/2)h_2 r_2 T_{s2}^*(1 - r_2/p_2) + c_2/T_{s2}^* \\
&= 1{,}920 + 1{,}920 + 460 + 460 \\
&= 4{,}760 \ \ \cos t \ per\, unit \ \ time.
\end{aligned}
$$

A Gantt chart of the first few cycles of this production schedule is shown in Figure 6.

Gantt Chart of Example Problem

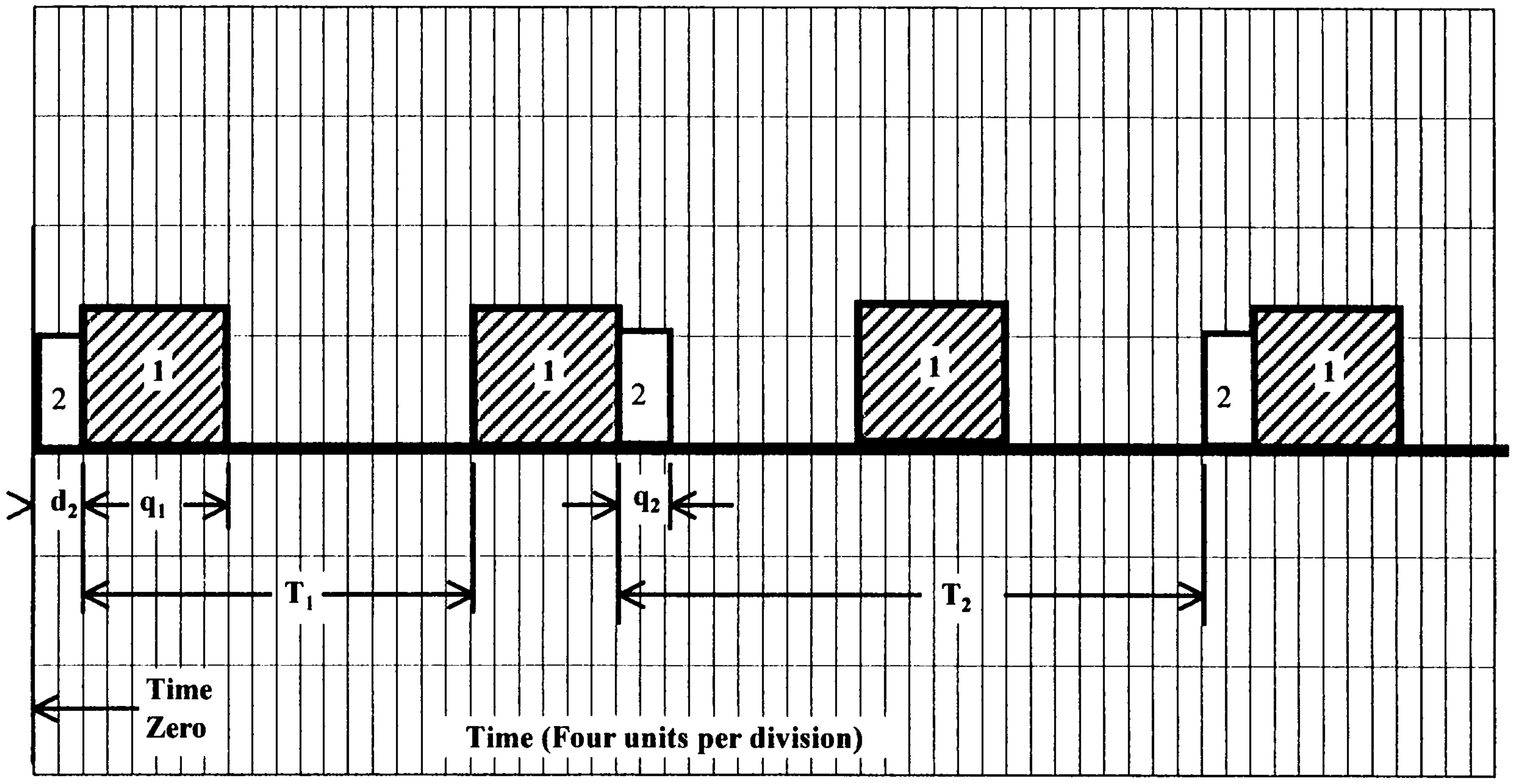


Figure 6

CHAPTER 4

ALGORITHM TO OPTIMIZE THE BASIC PERIOD MODEL

PURPOSE

As described in the history section of the introduction chapter Bomberger defined the BPM by imposing (20) and (21) on the GCM.

The purpose of this chapter is to present an algorithm that determines an optimal schedule, over all decision variables, for Bomberger's restricted BPM. This effort is useful for several reasons; namely,

1. Hodgson [43] shows that the minimum cost with respect to the multipliers (b_i) of the BP model for fixed a is not an unimodal function of a ; hence, Bomberger's trial and error method of determining a_r will not, in general, produce an optimal a_r. The algorithm presented here yields a global optimum for Bomberger's model over all integer variables.

2. The accuracy of Bomberger's procedure is affected by the value of Δ used to produce discrete dynamic programming state variables. In a straightforward dynamic programming formulation, Δ must satisfy two conflicting objectives. If too large a Δ is chosen, the algorithm may yield non-optimal results. On the other hand, it too small Δ is chosen, there will be a large number of feasible values for each state variable. This in turn

increases the computational requirements of the algorithm. In the algorithm presented here, the cycle times are restricted to integer values where the time unit is determined by the required scheduling precision (e.g., lots are scheduled for production to the nearest day or nearest quarter hour.) Then a revised form of Bomberger's dynamic programming algorithm is formulated. The Δ used to produce discrete state variables is then determined by the desired scheduling precision. Selecting Δ is equivalent to selecting a time unit for the scheduling problem. Basing this selection on the desired scheduling precision ranther than on computational considerations as required by Bomberger's procedure is a better defined method that yields more practical schedule.

Unfortunately, there is a downside to this method of selecting Δ. The desired scheduling precision may greatly increase the range of the state variables of the dynamic programming procedure. This, in turn, will significantly increase the computational effort required of the algorithm. To overcome this disadvantage certain properties of the model and an iterative algorithm are used to limit the number of feasible values of the state variables. In most cases, an optimal solution with a specified accuracy is produced by this approach with reasonable computational effort.

3. The procedure does not require intuitive judgement of the problem solver as required by Bomberger's trial and error method for selecting the

fundamental cycle. The algorithm will always produce an optimal, feasible solution for the BPM if model assumptions (a) through (k) and necessary conditions (1) are satisfied. Furthermore, the entire procedure can be programmed so that problems may be solved completely on a computer.

4. Lastly, as shown in the next chapter, when $\boldsymbol{n} = 2$, an optimal solution to Bomberger's model is an optimal solution for the general model. Also, Bomberger shows that the model often produces good solutions when $\boldsymbol{n} > 2$. Hence, the algorithm presented here finds optimal schedules when $\boldsymbol{n} = 2$ and produces a good upper bound on the cost of an optimal schedule when $\boldsymbol{n} > 2$. This upper bound is used in the general optimization algorithm for the $\boldsymbol{n}$-product model presented in chapter 9. Furthermore, many of the results of this chapter are adapted for use in the general algorithm.

In the next section the details of the BPM algorithm are developed. This is followed by a summary of the algorithm for the convenience of the reader and a proof that the algorithm produces an optimal solution for the BPM within a finite number of steps.

BPM ALGORITHM

Model Definition

Formally, the objective is to develop an algorithm which minimizes $Z(\Omega)$ as given by (4) for the model defined by assumptions (a) through (k) subject to Bomberger's BPM constraints (20) and (21). That is, the objective is to find

$$Z^* = \min_{\tilde{T}} Z(\tilde{T}) = \min_{T_i,\ all\ i} \sum_{1}^{n} (c_i / T_i + H_i T_i) \qquad (51)$$

subject to the restrictions of the integer forms of Bomberger's conditions

$$T_i = a \cdot b_i > 0,\ i = 1, 2, \cdots, n \qquad (52)$$

and

$$a \geq \sum_{1}^{n} [\rho_i T_i + s_i]^+ = \sum_{1}^{n} q_i(T_i) \qquad (53)$$

where

$$\rho_i = r_i / p_i,$$

$$H_i = h_i r_i (p_i / r_i) / 2 p_i = h_i \rho_i (p_i - r_i) / 2,$$

and b_i, all i, are integers. It is further assumed that a or in Bomberger's terminology the fundamental cycle, is integer which implies the T_i, all i, are integers. Also, unlike Bomberger's constraint (21), the length of the use periods, q_i all i, are required to be integers by (2). Inequality (52) is used to place initial lower bounds on the integer variables $\boldsymbol{a}, \boldsymbol{b}_i$, and T_i, namely, $\boldsymbol{a}, \boldsymbol{b}_i, T_i \geq 1$, for all $\boldsymbol{i}$.

The objective function (51) is a continuous, strictly convex function of $\{\boldsymbol{b}_i\}$ when the fundamental cycle $\boldsymbol{a}$ is held constant. Likewise, it is a continuous, strictly convex function of $\boldsymbol{a}$ when the $\{\boldsymbol{b}_i\}$ are held constant. However, the optimal values of (51) over $\{\boldsymbol{b}_i\}$, when $\boldsymbol{a}$ is constant, varies erratically as $\boldsymbol{a}$ ranges over its possible values.

To find an optimal schedule for the BPM in the face of this erratic behavior, the algorithm presented here uses the following strategy:

1. Find lower and upper bounds on the optimal of (51) where the upper bound is the cost of a known feasible schedule.

2. Use these bounds on cost and constraints (52) and (53) to place lower and upper bounds on the fundamental cycle $\boldsymbol{a}$.

3. Beginning with the fundamental cycle $\boldsymbol{a}$ set to its upper bound, find the optimal of (51) over the multipliers $\{\boldsymbol{b}_i\}$ for each value of $\boldsymbol{a}$ that lies between its lower and upper bounds. When necessary, Bomberger's dynamic programming routine is used to find the optimal multipliers.

4. At each new value of the fundamental cycle $\boldsymbol{a}$, if a better schedule is found replace the previous best schedule with this one and update the upper bound on (51) to equal the cost of the new schedule.

5. To increase the efficiency of the algorithm:

   a. Eliminate as many values of $\boldsymbol{a}$ as possible as either non-optimal or infeasible,

   b. At each value of $\boldsymbol{a}$ use bounds on the $\{\boldsymbol{b}_i\}$ to reduce their feasible region,

   c. Whenever possible, identify an optimal $\{\boldsymbol{b}_i\}$ without performing the dynamic programming routine, and

   d. At the end of each iteration, find $\boldsymbol{a}_b^*$ the optimal $\boldsymbol{a}$ when $\{\boldsymbol{b}_i\}$ is held fixed at the optimal $\{\boldsymbol{b}_i\}$ for that iteration. It is expected that in many cases will differ from the trial $\boldsymbol{a}$ used by the algorithm to find $\{\boldsymbol{b}_{ai}^*\}$. Thus, this step should usually find a lower cost schedule because

$$Z\left(\left\{\boldsymbol{a}_b^* \bullet \boldsymbol{b}_{ai}^*\right\}\right) \leq Z\left(\left\{\boldsymbol{a} \bullet \boldsymbol{b}_{ai}^*\right\}\right)$$

   for all values of $\boldsymbol{a}$.

It is possible to design an algorithm that begins at the lower bound on $\boldsymbol{a}$ and works upward to its upper bound. However, it is conjectured that the opposite approach will be more efficient for most problems. The rationale that supports this conjecture will be obvious from the step-by-step description of the algorithm.

Several charts similar to Figure 7 will be used to explain the rationale of the algorithm and to facilitate the proofs in each step. These figures require some explanation. The chart in each of these figures is a depiction of the $\left(b_i,\ b_k\right)$ plane for a particular value of the fundamental cycle a. For each value of a there is a related plane for each possible pair $\left(b_i,\ b_k\right)$ where $i \neq k$ and $i,\ k = 1,\ 2,\ \cdots,\ n$. These charts represent the n-dimensioned $\left(b_1,\ b_2,\ \cdots,\ b_n\right)$ hyperplane in n-space.

The two dashed lines radiating from the origin are imaginary in the sense that they do not actually appear in any $\left(b_i,\ b_k\right)$ plane. To construct these lines imagine that the integer constraint on a is relaxed, that a $\left(b_i,\ b_k\right)$ plane is drawn for each possible value of a, and that this infinite set of charts are overlaid, ordered on a. One and only one point on each of these lines will appear in the plane related to a particular value of a. These imaginary lines are constant cost lines and are useful for showing how certain aspects of the $\left(b_i,\ b_k\right)$ plane changes as a is changed.

Except for the imaginary lines each $\left(b_i,\ b_k\right)$ plane is the one-to-one mapping from the $\left(T_i,\ T_k\right)$ plane given by (52) for each constant value of a. There are two major advantages for using the $\left\{b_i\right\}$ hyperplane rather than the $\left(T_i,\ T_k\right)$ hyperplane. Assuming that a is a constant integer, each integer in the $\left\{b_i\right\}$ hyperplane maps to an integer in the $\left\{T_i\right\}$ hyperplane. The reverse is not true. Also, the constraint (53) in the $\left\{T_i\right\}$ hyperplane divides the hyperplane into a "maybe" region and infeasible regions not into feasible and infeasible regions as its image does in the $\left\{b_i\right\}$ hyperplane. More will be said about these hyperplanes as the steps of the algorithm are described.

BPM Algorithm (b_i, b_k) Plane
Medium a

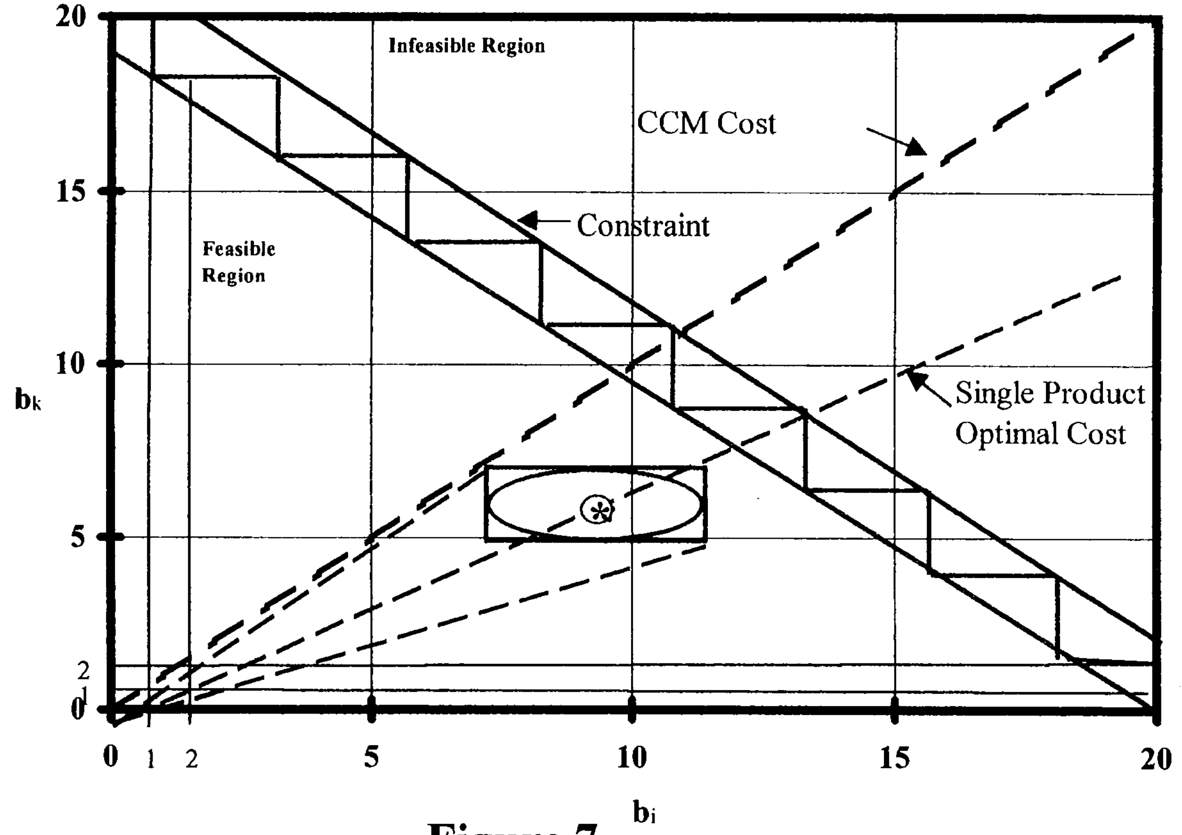


Figure 7

Another curious aspect of Figure 7 is the "sawtooth" curve depicting constraint (53) and the two enclosing parallel lines. The two parallel lines divide the $\left(b_i,\ b_k\right)$ plane into three regions: a feasible region, an infeasible region, and a "maybe" region. This constraint structure is another complication caused by the integer constraint on all model variables. To explain, consider the upper parallel line first and let

$$a_I = \sum_I^n q_i = S + \sum_I^n \left(\rho_i T_i\right)^+ \tag{54}$$

and

$$a_r = \sum_I^n \left(s_i + \rho_i T_i\right) = S + \sum_I^n \rho_i T_i. \tag{55}$$

where a_r is a real number and a_I is an integer.

Applying the division algorithm to T_i, all i, let

$$T_i = m_i / \rho_i - \epsilon_i / \rho_i \tag{56}$$

where $m_i \geq 1$ and $0 \leq \epsilon_i / \rho_i < 1/\rho_i$. Then

$$\begin{aligned} a_r &= S + \sum_I^n \left(m_i - \epsilon_i\right) \\ &= a_I - \sum_I^n \epsilon_i \end{aligned}$$

where $0 \leq \epsilon_i < 1$. Thus, $a_r \leq a_I$. Suppose a vector $\left\{T_i\right\}$ yields an a_r such that $a_r > a$ for a particular value of a then

$$a_I \geq a_r > a.$$

Hence, all vectors $\{T_i\}$ such that $a_r > a$ do not satisfy (53) and are infeasible. Note that $a_r = a_I$ only when $T_i = m_i / \rho_i$, all i.

To consider the lower parallel line, note that (54) is actually an integer function of the vector $\{q_i\}$. Thus, when a_I is fixed in (54) and the value of one q_i changes then there must be off-setting changes in another q_k, $k \neq i$. By examining the plot of a_I in the $(T_i,\ T_k)$ plane shown in Figure 8 one sees that a_I is a continuous, "sawtooth" function of T_i and T_k except at the points $m_i/\rho_i, (m_k - 1)/\rho_k$ or $(m_i - 1)/\rho_i, m_k/\rho_k$ where

$$a_I - S - \sum_{\substack{j \neq i \\ j \neq k}}^{n} \left(\rho_j T_j\right)^+ = m_i + m_k.$$

These discontinuities are removable because

$$\begin{aligned} \lim_{\epsilon \uparrow 1} q_i &= \lim_{\epsilon \uparrow 1} \rho_i T_i \\ &= \lim_{\epsilon \uparrow 1} m_i - \epsilon \qquad (57) \\ &= m_i. \end{aligned}$$

An important observation about a_I in Figure 8 is that over any continuous segment of a_I $T_i = m_i / \rho_i$ for at least one i. Hence, at least one $\epsilon_i = 0$ so

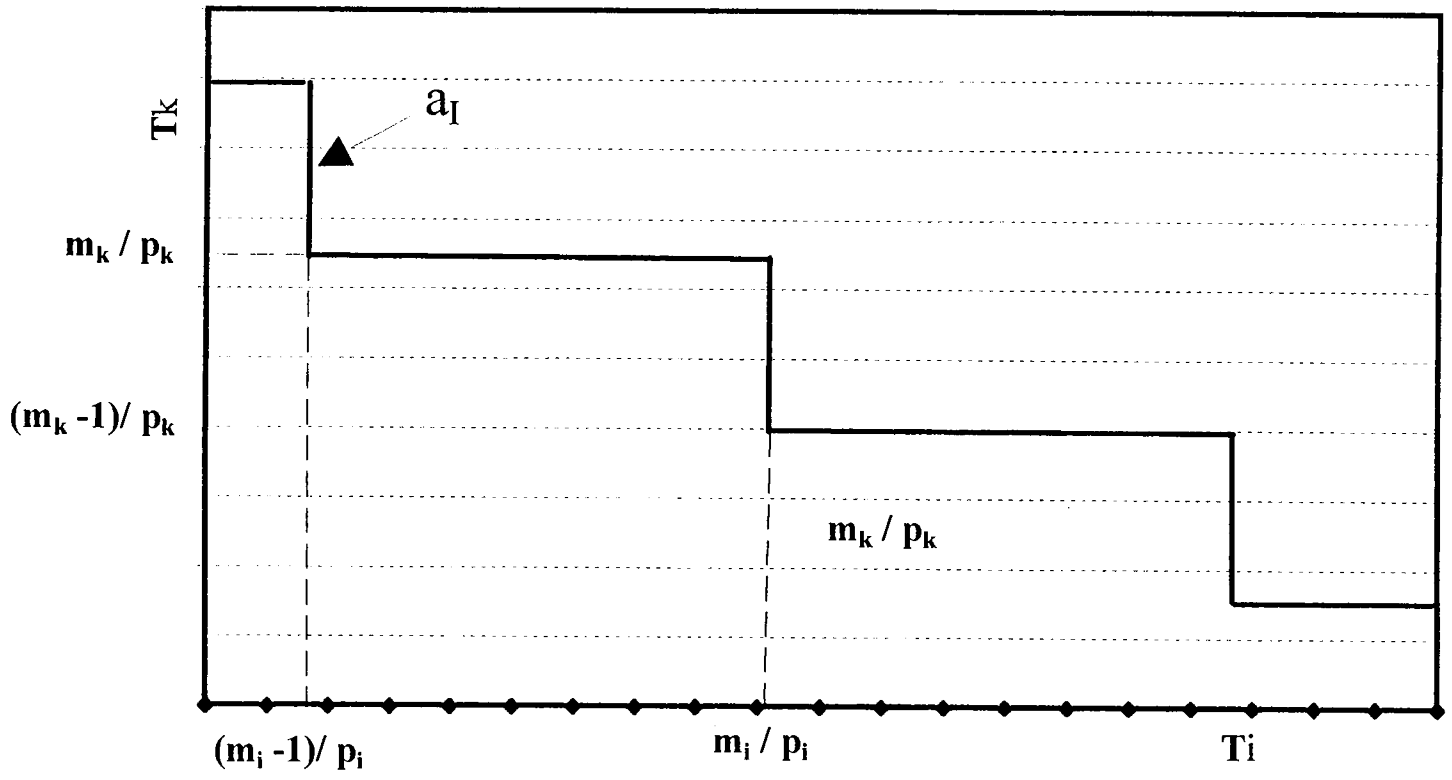


Figure 8

$$\sum_1^n \epsilon_i < (n-1).$$

Now define T_i as shown in (56) and let

$$\begin{aligned} a'_r &= S + \sum_1^n \rho_i T_i + (n-1) \\ &= a_1 + \left[(n-1) - \sum_1^n \epsilon_i\right]. \\ &> a_1. \end{aligned} \tag{58}$$

If a vector $\{T_i\}$ yields an a'_r such that $a'_r \leq a$ for a particular value of a then

$$a \geq a'_r \geq a_1.$$

Hence, all such vectors $\{T_i\}$ satisfy (53) and are feasible. There is equality in (58) only as a limit. This equality occurs when one $T_i = m_i/\rho_i$ and the other $(n-1)$ T_k approach

$$T_k = \lim_{\epsilon \uparrow 1} m_k/\rho_i - \epsilon.$$

The parallel lines defined by (55) and (58) when $a = a_r = a'_r$ are tight upper and lower linear bounds, respectively, on constraint (53). These lines are useful approximate linear constraints for the integer BPM.

To facilitate the description of the algorithm the range of a is divided into five partitions that are labeled very large a, large a, medium a, small a, and very small a. These partitions are defined relative to constraints (52) and (53). The characteristics of the scheduling problem within each of these partitions differs significantly from that within the other partitions.

Several charts similar to Figure 7 will be used to guide the explanation of these differences as the steps of the algorithm are described.

Step 1

The actions of steps 1, 2, and 3 take place in the $\left(T_i, T_k\right)$ plane as shown in Figure 9 where the single-product optimal point T_{si}^*, T_{sk}^* is marked with an asterisk. The constraint and its parallel lines shown in Figure 9 are described above. The other curve and the limit box will be described in steps 3 and 4. When $a > 0$ all of Figure 9 is mapped one-to-one to the $\left(b_i, b_k\right)$ plane shown in Figure 7. These first three steps are performed only once during the execution of the algorithm.

Obtain an initial upper bound on z^* from Hanssman's CCM. Hanssman's [45] results can be quickly extended to obtain an integer solution for his model because his assumption that $T_i = T_h$, all i, converts the n-item model to a single item model where

$$C_h = \sum_1^n c_i \text{ and } H_h = \sum_1^n H_i \tag{59}$$

are the setup and inventory costs, respectively. To obtain this integer solution, let

$$Z_h(T) = C_h / T + H_h T \tag{60}$$

and

$$T_{hr} = \sqrt{C_h / H_h}. \tag{61}$$

BPM Algorithm (T_i, T_k) Plane

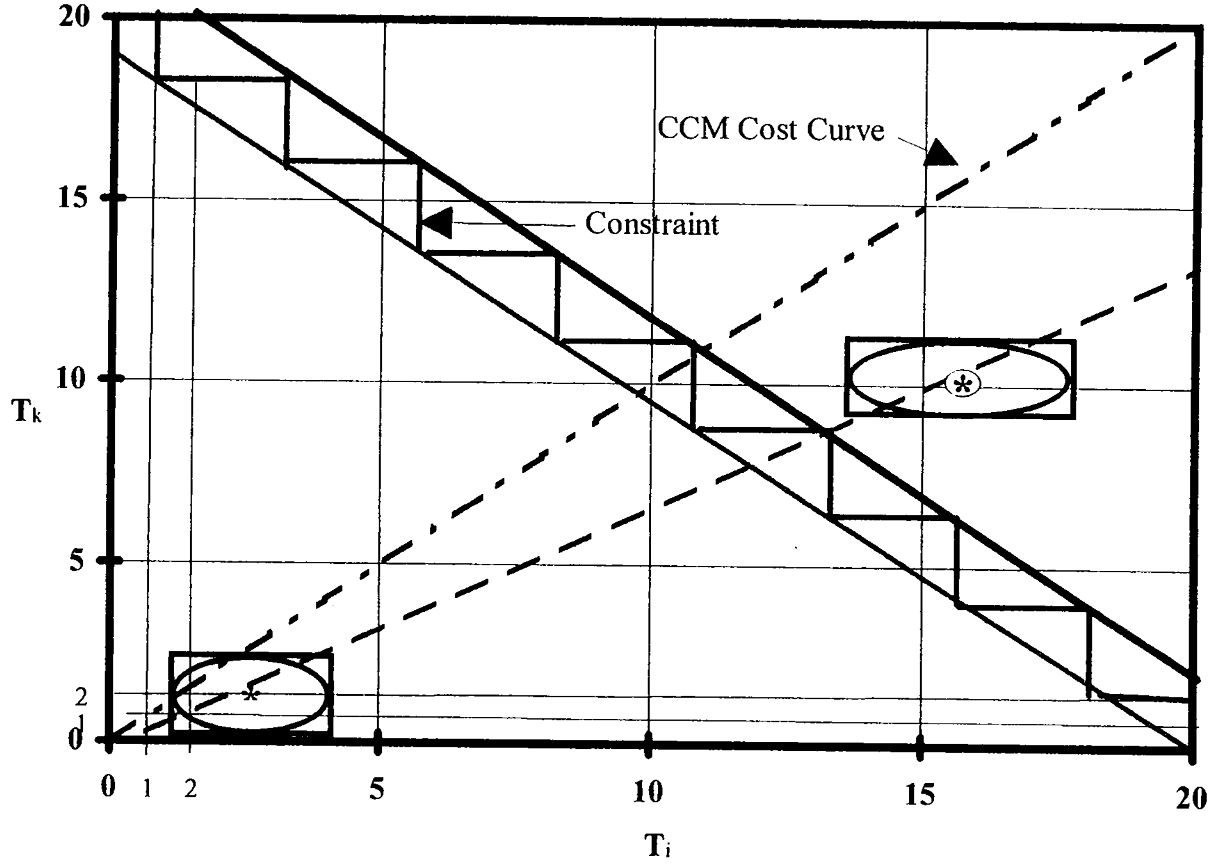


Figure 9

Then the optimal integer T_h is

$$T_h^* = max\left(T_{h1}, T_{h2}\right) \tag{62}$$

where

$$T_{h1} = \begin{cases} \left(T_{hr}^*\right)^+, & \left(T_{hr}^*\right)^- = 0 \\ \left(T_{hr}\right)^-, & Z_h\left[\left(T_{hr}\right)^-\right] < Z_h\left[\left(T_{hr}\right)^+\right] \text{ and } \left(T_{hr}\right)^- > 0 \\ \left(T_{hr}\right)^+, & Z_h\left[\left(T_{hr}\right)^-\right] \geq Z_h\left[\left(T_{hr}\right)^+\right] \text{ and } \left(T_{hr}\right)^- > 0 \end{cases} \tag{63}$$

and

$$T_{h2} = \sum_1^n q_i\left(T_{h2}\right) = S + \sum_1^n \left(\rho_1 T_{h2}\right)^+ > 0 \tag{64}$$

where

$$S = \sum_{i=1}^{n} s_i .$$

A lower bound on T_{h2} is provided by

$$T_{h2} > \left[S / \left(1 - \sum_1^n \rho_i\right)\right]^+ > 0. \tag{65}$$

The cycle time T_{h2} is found by computing this lower bound and then, if necessary, by increasing it by 1 until (64) is satisfied.

Note that if there is a tie in (63) then the largest value for T_{h1} is selected because if $\left(T_{hr}^*\right)^-$ is feasible then so is $\left(T_{hr}^*\right)^+$ but the reverse is not true. The optimum cost for a Hanssman CCM is

$$z_h^* = Z_h\left(T_h^*\right). \tag{66}$$

Hanssman shows that schedule feasibility is assured by requiring $T_h \geq T_{h2}$. By letting $b_i = 1$, all i, and $a = T_h^*$ and observing that (64) is equivalent to (53) when $T_i = T$, all i, we see that $T_i = T_h^*$, all i, is a feasible solution to Bomberger's model, so z_h^* is an initial upper bound on Z^*. Save this initial upper bound solution by setting $Z_u = z_h^*$ and $T_{bi} = T_h^*$, all i, and go to step 2.

Step 2

Find optimal solutions to n single-products EMQ models and check these for schedule feasibility. If the vector of these single-products cycle time are feasible then set $T_i^* = T_{si}^*$, all i, and $z^* = \sum_1^n z_{si}^*$ and terminate the algorithm.

a. Treat each item as if it is the only item of a single item model and let

$$Z_i\left(T_i\right) = c_i \,/\, T_i + H_i T_i; \; i = 1, \cdots, n; \tag{67}$$

suppose that T_i is permitted to assume any positive real value. Then, because assumption (k) requires that $c_i, b_i > 0,$, all i, it is easily shown that $Z_i\left(T_i\right)$ is strictly convex with the unique extreme point

$$T^*_{rsi} = \sqrt{c_i / H_i}, i = 1, \cdots, n. \tag{68}$$

If T^*_{rsi} is not integer, then the optimal integer cycle time of the ***ith*** single-product model T^*_{si} must be one of the two (or both) integers which bracket T^*_{rsi}. Thus, because $z_{si}(T_i)$ is strictly convex, it is easily shown that

$$T_{si1} = \begin{cases} (T^*_{rsi})^+, & (T^*_{rsi})^- = 0 \\ (T^*_{rsi})^-, & Z_i\left[(T^*_{rsi})^-\right] < Z_i\left[(T^*_{rsi})^+\right] \textit{ and } (T^*_{rsi})^- \geq 1 \\ (T^*_{rsi})^+, & Z_i\left[(T^*_{rsi})^-\right] \geq Z_i\left[(T^*_{rsi})^+\right] \end{cases} \tag{69}$$

for $i = 1, 2, \cdots, n$. In case of a tie, both $(T^*_{rsi})^-$ and $(T^*_{rsi})^+$ are used in the feasibility test of step 1.b and in other places in the algorithm.

Note that one or more of the single product optimal cycle times T^*_{si} may not satisfy constraint (53) even when product i is treated as a single-product model. In this case

$$q_i(T^*_{si}) = [s_i + \rho_i T^*_{si}]^+ > T^*_{si} \geq a_s. \tag{70}$$

However, a greater lower bound on T_i than that given by (70) is obtained when product i is treated as an item in the n-products model. This greater bound is derived by observing that $T_i \geq a$ and $T_i \geq 1$, all i, and applying (53) to obtain

$$T_k \geq a \geq S + \sum_{i-1}^{n} (\rho_i T_i) \geq S + \sum_{i \neq k} \rho_i + \rho_k T_k$$

so

$$T_{sk2} \geq \left[\left(S + \sum_{i \neq k} \rho_i \right) / \left(1 - \rho_k \right) \right]^+ . \tag{71}$$

Thus T_{si2} is the minimum feasible integer T_{si} of the modified single-product model. Increasing T_{si} beyond T_{si2} when $T_{si1} < T_{si2}$ increases the cost z_{si} because the cost function is strictly convex. Thus,

$$T_{si}^* = max\left(T_{si1},\ T_{si2} \right) \tag{72}$$

if there are ties in (69) T_{si1} is set to the maximum possible value. Of course, if $T_{si}^* = T_{si2}$ there are no ties in T_{si}^*. This will simplify the remaining steps somewhat. To derive (71) T_i, $i \neq k$, were all set to one. This is the extreme lower bound on T_i. As shown in step 4, a greater lower bound on T_{sk2} is obtained when the lower bound on any T_i, $i \neq k$. increases.

Compute

$$z_{si}^* = c_i \ / \ T_{si}^* + H_i T_{si}^* ; i = 1, \cdots, n \tag{73}$$

and save it for later use. Clearly, $\sum_1^n z_{si}^*$ is a lower bound on z^* over all possible integer solutions because setting any T_i to a feasible integer value other than T_{si}^*, except for possible ties, causes a strict increase over $\sum_1^n z_{si}^*$.

b. Determine $a_s = gcd\left(T_{si}^*, all\ i\right)$. An elementary result from number theory states that, if there is an integer a' such that $T_{si}^* = a'b_i'$, all i, where the b_i' are integers, then $a_s \geq a'$. Therefore, $\left(T_{si}^*\right)$ satisfies (52) and (53), if, and only if, $b_i^* \geq 1$ and

$$a_s \geq \sum_{i=1}^n q_i\left(T_{si}^*\right) \tag{74}$$

If there are two possible values for one or more of the T_{si}^* this step is repeated for every possible value of $\left\{T_{si}^*\right\}$.

If (74) is satisfied by any of the possible $\left\{T_{si}^*\right\}$ terminate the algorithm with

$$z^* = \sum_1^n z_{si} \tag{75}$$

and

$$T_i^* = T_{si}^*, \tag{76}$$

all i where $\left\{T_{si}^*\right\}$ is the vector that satisfies (74). If the algorithm terminates here, clearly z^* is the optimal cost because it equals the lower bound on cost given by (73) while $\left\{T_i^*\right\}$ given by (72) is feasible. Proceed to step 3 if the algorithm is not terminated.

Step 3

Obtain an initial upper bound on $\boldsymbol{a}$. Suppose that $\boldsymbol{T_i = T}$, all $\boldsymbol{i}$, and that $\boldsymbol{T}$ is set to the maximum of the optimum single item cycle times $\left(\boldsymbol{i.e., T = max_i\left(T_{si}^*\right)}\right)$ and consider Figure 10 which illustrates the situation when $\boldsymbol{n = 2}$. (For clarity, it is assumed that $\boldsymbol{T_{si}^* = T_{sri}^*}$ in the illustration.)

Because the $\boldsymbol{Z_i\left(T_i\right)}$ are strictly convex $\boldsymbol{T_i = max_i\left(T_{si}^*\right)}$, all $\boldsymbol{i}$, has lower cost than any other solution where $\boldsymbol{max_i\left(T_{si}^*\right) \leq T_i}$, all $\boldsymbol{i}$. Thus, if this solution is feasible, $\boldsymbol{max_i\left(T_{si}^*\right)}$ is an upper bound on the minimum cycle time of an optimum solution $\left(\boldsymbol{i.e., min_i\left(T_i^*\right) \leq max_i\left(T_{si}^*\right)}\right)$. If this solution is not feasible, then $\boldsymbol{max_i\left(T_{si}^*\right) < T_{h2}}$ as illustrated in Figure 10 where $\boldsymbol{T_{h2}}$ is given by (64). In this case, $\boldsymbol{T_{h2} \geq min_i\left(T_i^*\right)}$. By noting that $\boldsymbol{T_i \geq a}$, all $\boldsymbol{i}$, these observations lead to an initial upper bound on $\boldsymbol{a^*}$; namely,

$$\boldsymbol{a_u = max\left(T_{h2}, max_i\left(T_{si}^*\right)\right) \geq min_i\left(T_i^*\right) \geq a^*}.$$

Save this initial value of $\boldsymbol{a_u}$ and proceed to Step 4.

Step 4

Use the present upper bound on z^* and the lower bound on single item costs, z_{mi}, to place bounds on the optimal cycle times or to terminate the algorithm.

Upper Bound on Fundamental Cycle a

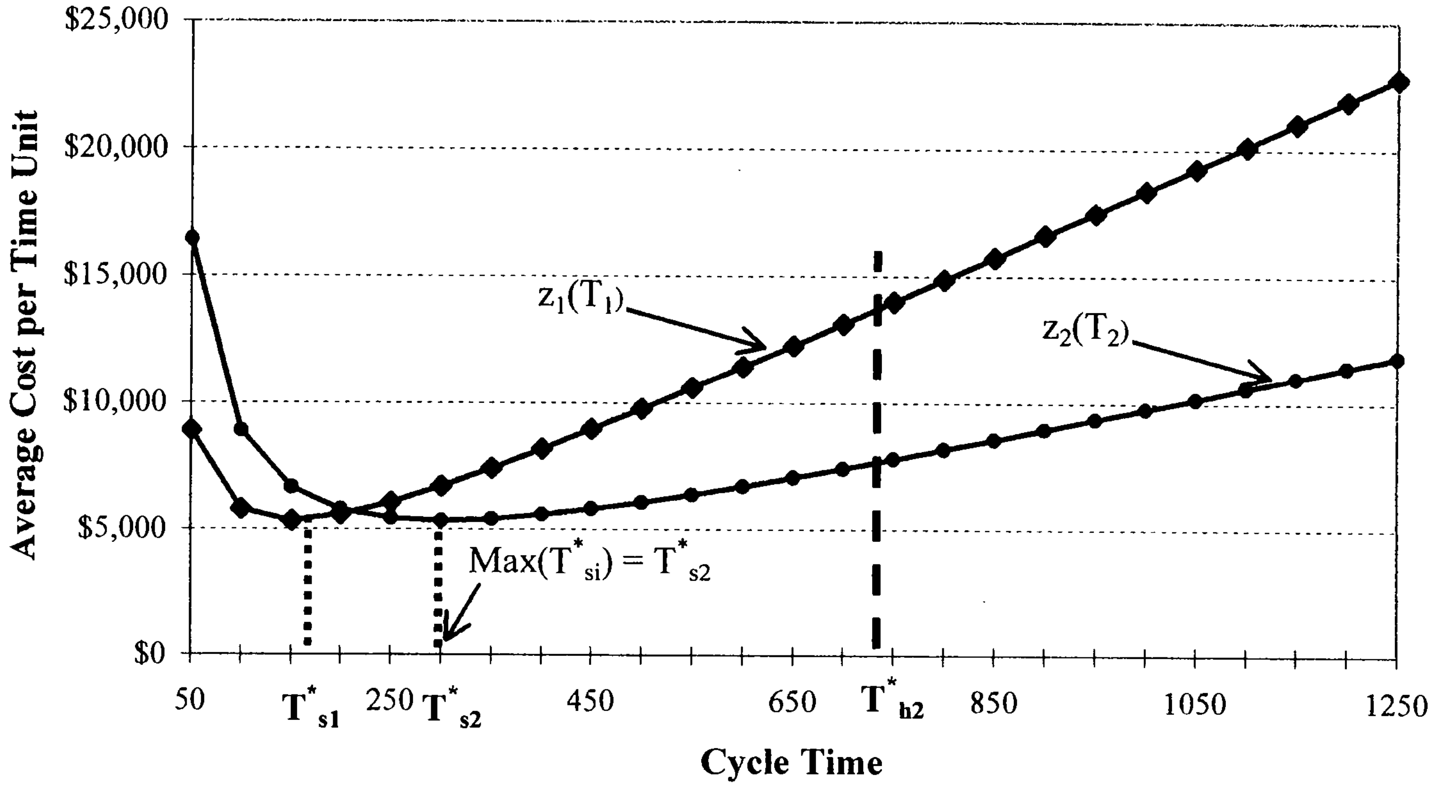


Figure 10

a. If this is the first execution of Step 4 let $T'_{li} = 1$ and $T'_{ui} = M$ where M is a very large integer. Otherwise, set $T'_{li} = T_{li}$ and $T'_{ui} = T_{ui}$ where T_{li} and T_{ui} are the present bounds on T_i. Let

$$z_{mi} = \begin{cases} Z_i(T'_{ui}), & T'_{ui} < T^*_{si} \\ z^*_{si}, & T'_{li} \le T^*_{si} \le T'_{ui} \\ Z_i(T'_{li}), & T'_{li} > T^*_{si} \end{cases} \tag{77}$$

Because $Z_i(T_i)$ is a strictly convex function of T_i the quantity z_{mi} is a lower bound on $Z_i(T_i)$ over all T_i such that $T'_{li} \le T_i \le T'_{ui}$. Thus,

$$Z_l = \sum_{i=1}^{n} z_{mi} \tag{78}$$

is a lower bound on Z^*. Clearly, if

$$\Pi = Z_u - Z_l \le 0, \tag{79}$$

the vector of cycle times $\{T_{bi}\}$ which yields Z_u is an optimal solution. Hence, if (79) is satisfied, terminate the algorithm with $Z^* = Z_u$ and $T_i^* = T_{bi}$, all i; otherwise, proceed to Step 4.b.

b. If the algorithm is not terminated in Step 4.a, then new upper and lower bounds can be placed on the optimal value of each T_i because

$$
\begin{aligned}
Z_u &> z^* \\
&= \sum_1^n \left(c_i / T_i^* + H_i T_i^* \right) \\
&= \sum_1^n Z_i \left(T_i^* \right).
\end{aligned}
$$

Each item i must contribute at least z_{mi} to z^* so the maximum cost contribution of any item j is

$$
\begin{aligned}
c_j T_j^* + H_j T_j^* = Z_j\left(T_j^*\right) & \\
&\le Z^* - \sum_{i \neq j} z_{mi} \\
&< Z_u - \sum_{i \neq j} z_{mi} \\
&= \Pi + z_{mj}
\end{aligned}
\tag{80}
$$

The points at which the two sides of (80) are equal can be found by solving a quadratic in T_j. As illustrated in Figure 11, with $\Pi > 0$ this quadratic always has two real roots $\left(y_1, y_2\right)$ which bound T_j^*. Solving for these bounds, we have

$$
T_{lj}'' \le T_j^* \le T_{uj};\, j = 1,\, 2, \cdots, n; \tag{81}
$$

where

$$
T_{lj}'' = \left\{ \left[\Pi + z_{mj} - \sqrt{\left(\Pi + z_{mj}\right)^2 - 4H_j c_j} \right] / \, 2H_j \right\}^+ \tag{82}
$$

and

$$
T_{uj} = max\left(T_{uj}', \left\{ \left[\Pi + z_{mj} + \sqrt{\left(\Pi + z_{mj}\right)^2 - 4H_j c_j} \right] / \, 2H_j \right\} \right) \tag{83}
$$

Lower and Upper Cost Bounds on Cycle Times

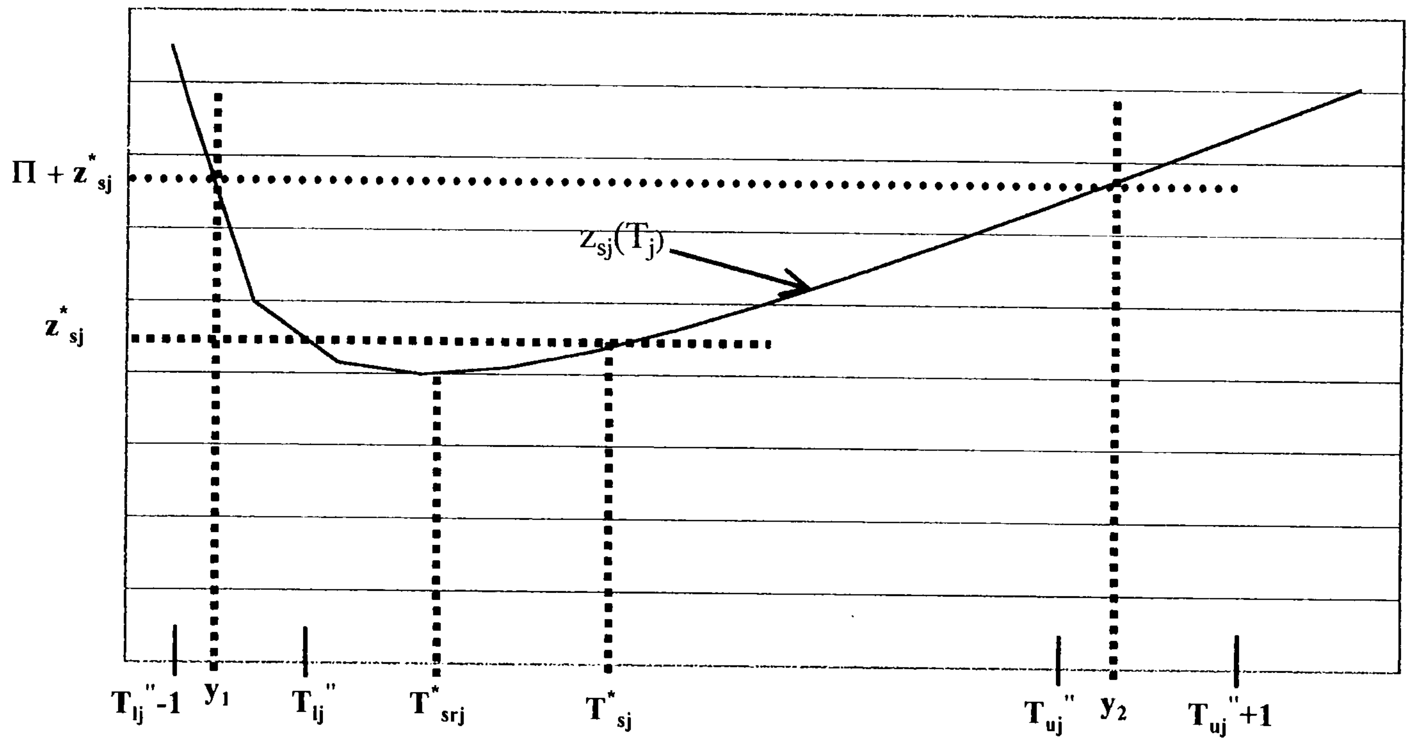


Figure 11

Solve for these bounds on each T_j^* and go to Step 5.

The remaining attribute of Figures 7 and 9 can be explained now. The box bordered by dashed lines is defined by the limits $\left(T_{lj}, T_{uj}\right)$, all j. All potentially optimal $\{T_i\}$ where $Z_u > Z\left(\{T_i\}\right)$ are contained within this n-dimensional box. The approximate ellipsoid contained within the limit box is the constant cost curve where $Z\left(\{T_i\}\right) = Z_u$. Interior points of this approximate ellipsoid have cost lower than the present upper bound on cost Z_u. As shown in Figure 9 it is possible that part of this ellipsoid may be outside the limit box. The algorithm is terminated if all of the ellipsoid is outside the limit box in the $\left(T_i,\ T_k\right)$ plane.

Two constraints, each with its two enclosing parallel lines, appear in the $\left(T_i,\ T_k\right)$ plane shown in Figure 9. The upper constraint is fixed by a_u while the lower constraint is fixed by a_l. Only one constraint appears in the $\left(b_i,\ b_k\right)$ plane of Figure 7. This constraint is related to a fixed a where $a_l \leq a \leq a_u$.

Step 5

a. Use Bomberger's condition and the lower bounds on the cycle times to determine a lower bound for a.

From (53) and (81), we find that a lower bound on a is given by

$$\begin{aligned} a^* &\geq \sum_1^n q_i T_i^* \\ &\geq \sum_1^n q_i (T_{li}) \\ &= a_l. \end{aligned} \tag{84}$$

b. Find a new lower bound on T_i^*, all i. If the bounds on the cycle times T_i are inconsistent so that no feasible solutions $\{T_i\}$ with cost $Z(\{T_i\}) < Z_u$ exist then terminate the algorithm. The lower bound a_l on a is also a lower bound on all T_i^* that is

$$T_i^* \geq min(T_i^*) = min(a^* b_i^*) \geq a^* \geq a_l \tag{85}$$

so set

$$T_{li} = max(a_l, T_{li}', T_{li}'') \tag{86}$$

where T_{li}' and T_{li}'' are defined in Step 4. If $T_{li} > T_{ui}$ for any i then all potentially optimum schedule have been eliminated so set $T_i^* = T_{bi}$ and $Z^* = Z_u$ and terminate the algorithm. This situation is shown in Figure 12 and Figure 13.

If $T_{li} = T_{ui}$ for all i go to step 5.c. Otherwise, $T_{li} \leq T_{ui}$ for all i and $T_{li} < T_{ui}$ for at least one product i. Determine if T_{li}, for any i, increased during the previous iteration of Steps 4 and 5, i.e. check if $T_{li} > T_{li}'$ If so repeat Steps 4 and 5; otherwise, go to Step 6.

c. When $T_{li} = T_{ui}$ for all i find the final fundamental cycle a_f used by the algorithm; that is, find

$$a_f = gcd\left(T_{l1},\ T_{l2},\ \cdots,\ T_{ln}\right).$$

If a_f and $\left\{T_{li}\right\}$ satisfy (53) and $Z\left(\left\{T_{li}\right\}\right) < Z_u$ set $T_i^* = T_{li}$, all i, and $z^* = Z\left(\left\{T_{li}\right\}\right)$; otherwise, set $T_i^* = T_{bi}$, all i, and $z^* = Z_u$. In either case, terminate the algorithm.

Step 6

Set $a_u' = a_u$ the present upper bound on the fundamental cycle, and use the upper bounds on the cycle times to determine a new upper bound on a.

Two additional upper bounds can now be placed on a. The first is simply

$$a_{u1} = min_i\left(T_{ui}\right) \geq min_i\left(T_i\right) \geq a^*. \tag{87}$$

The second bound is not as obvious. Remember that (52) and the integer constraint requires that $b_i \geq 1$, all i. Suppose that there is a solution with $Z^* < Z_u$ and with all $b_i^* = 1$. Then $T_i^* = a^*$, all i, which means our supposed solution is a solution to Hanssman's CCM. But the optimal solution to the Hanssman's CCM is the initial upper bound on Z^* so $z_h^* \geq Z_u$. This contradiction shows that if $Z^* < Z_u$ then there must be at least one $b_i^* \geq 2$, which implies that

BPM Algorithm (T_i, T_k) Plane

$T_{ui} < T_{li}$

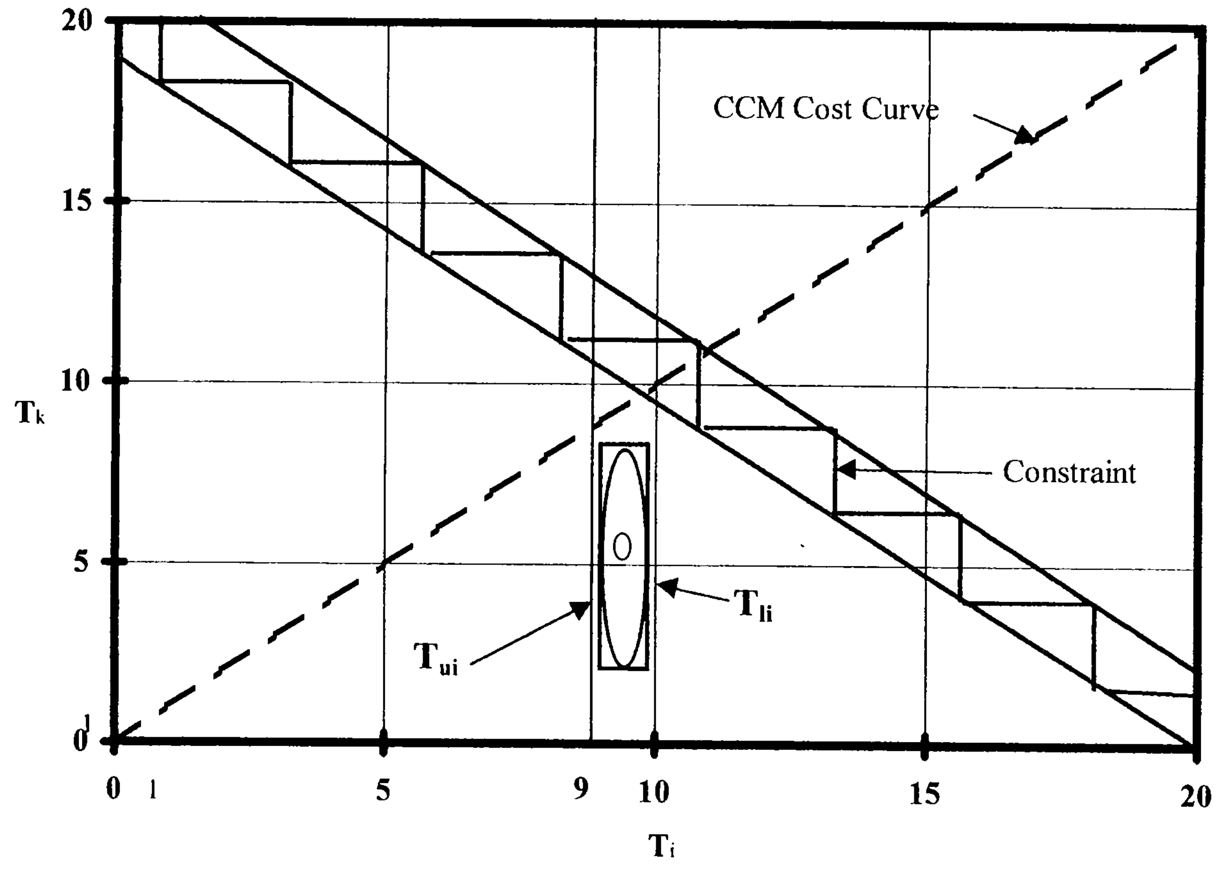


Figure 12

BPM Algorithm (T_i, T_k) Plane

$T_{uk} < T_{lk}$

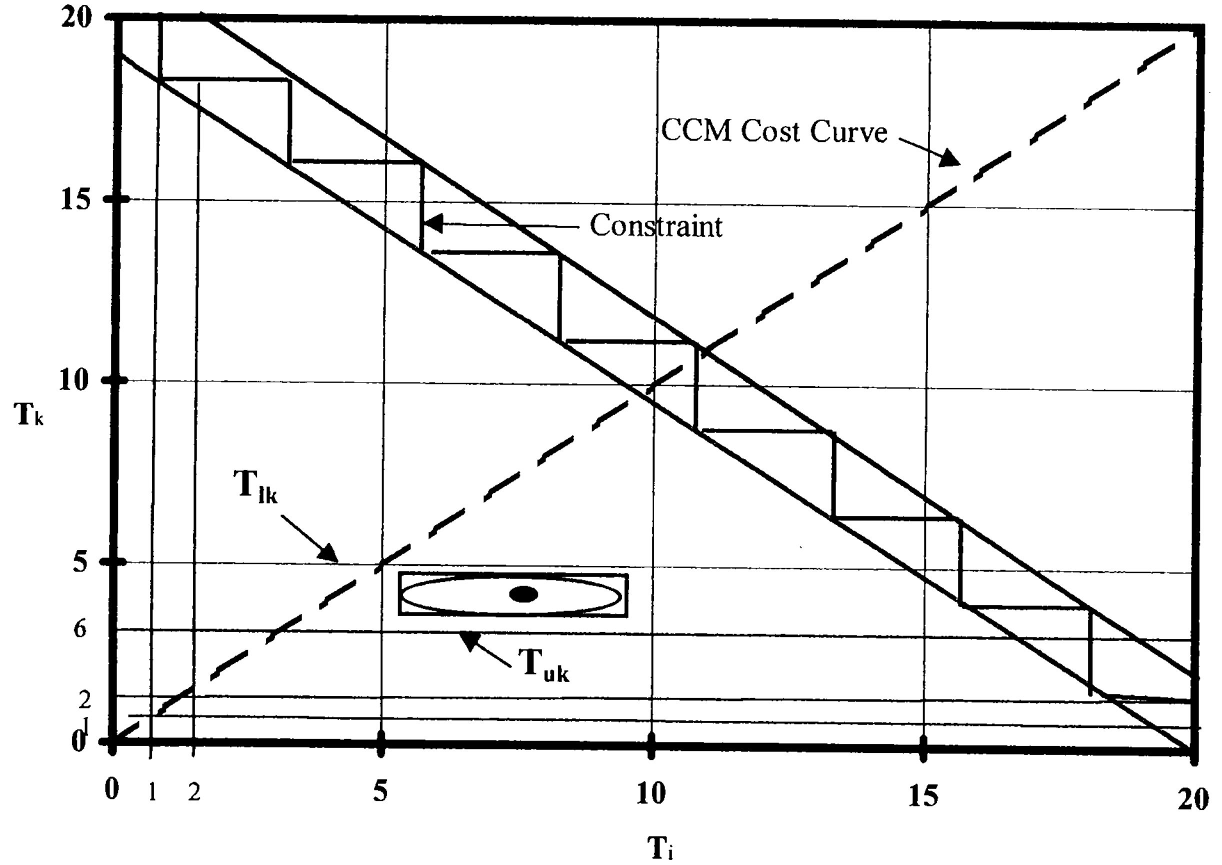


Figure 13

$$\begin{aligned} a^* &\le \left[T_i^* / 2\right]^- \\ &\le \left[max_i(T_i) / 2\right]^- \\ &\le \left[max_i(T_{ui}) / 2\right]^- \\ &\le a_{u2}. \end{aligned} \tag{88}$$

where $b_i^* \ge 2$ for item i.

Finally, recall that a_u' is the previous best upper bound on a^*, the new upper bound is

$$a_u = \min\left(a_u', a_{u1}, a_{u2}\right). \tag{89}$$

The situation where all $b_{iu} < 1$ is shown in Figure 13. This is representative of $\left(b_i,\ b_k\right)$ plane when the fundamental cycle a is *very large*. Values of a in the *very large partition* are clearly infeasible because such values require that $b_i < 1$, all i.

Because of (87) the lower bound on all b_{ui} is 1, that is,

$$1 \le \min_i\left(T_{ui}\right) / a^* = \min_i\left(b_{ui}\right) \le T_{uk} / a^*$$

for $k = 1,\ 2,\ \cdots,\ n$. Thus, when the integer constraint is relaxed and the potentially optimal region of the $\left(T_i,\ T_k\right)$ plane is mapped to the $\left(b_i,\ b_k\right)$ plane at most one $b_{ui} = 1$ unless $T_{ui} = T_{uk}$. This mapping is shown in Figure 14 where $b_{u2} > 2$. However, it is possible that with the integer constraint $b_{ui} = 1$, all i. This situation is shown in Figure 15.

BPM Algorithm (b_i, b_k) Plane

Large a, $b_{ui} = 1$, $b_{uk} > 2$

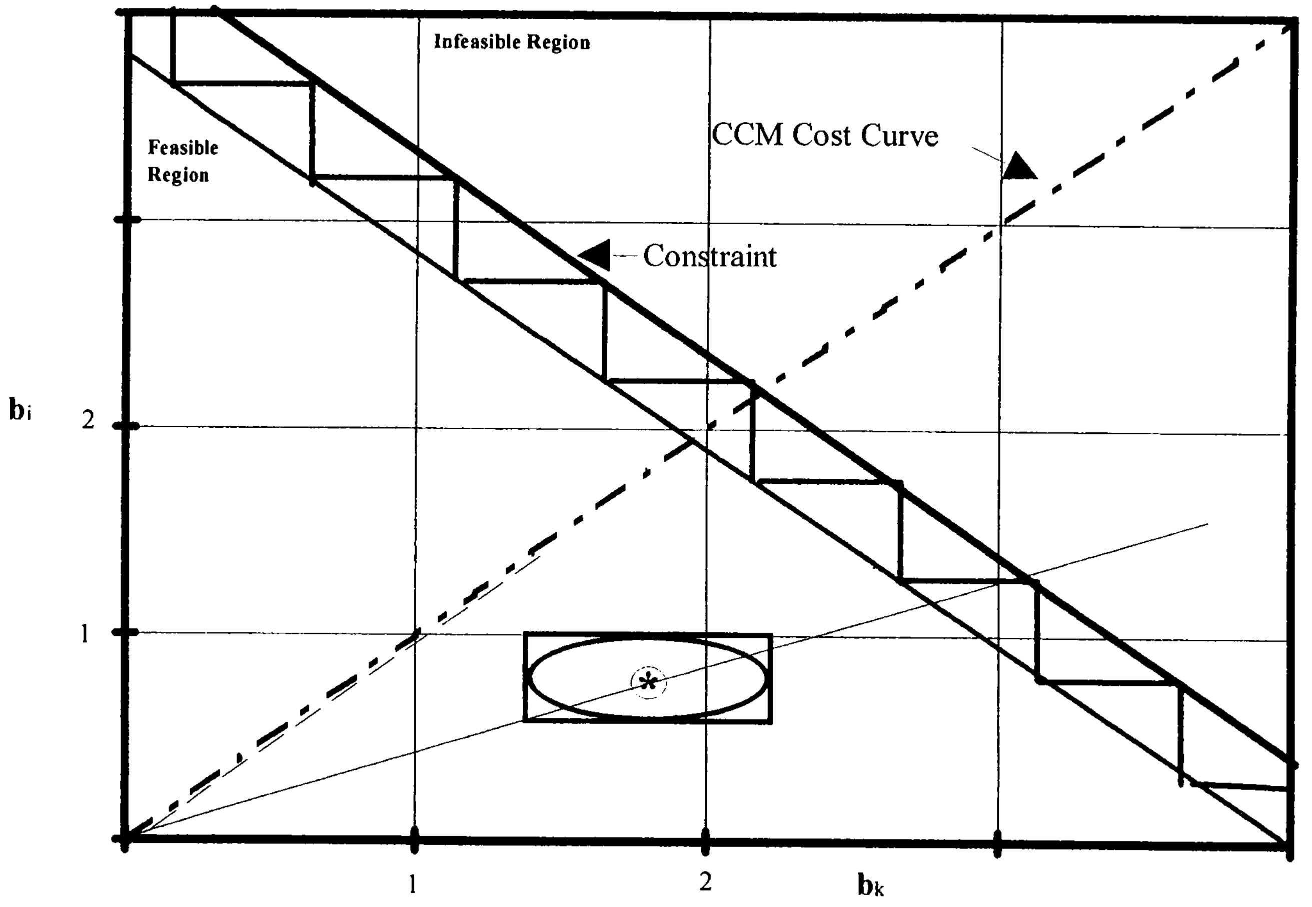


Figure 14

BPM Algorithm (b_i, b_k) Plane
Large a, $b_{ui} = 1$, $b_{uk} = 1$

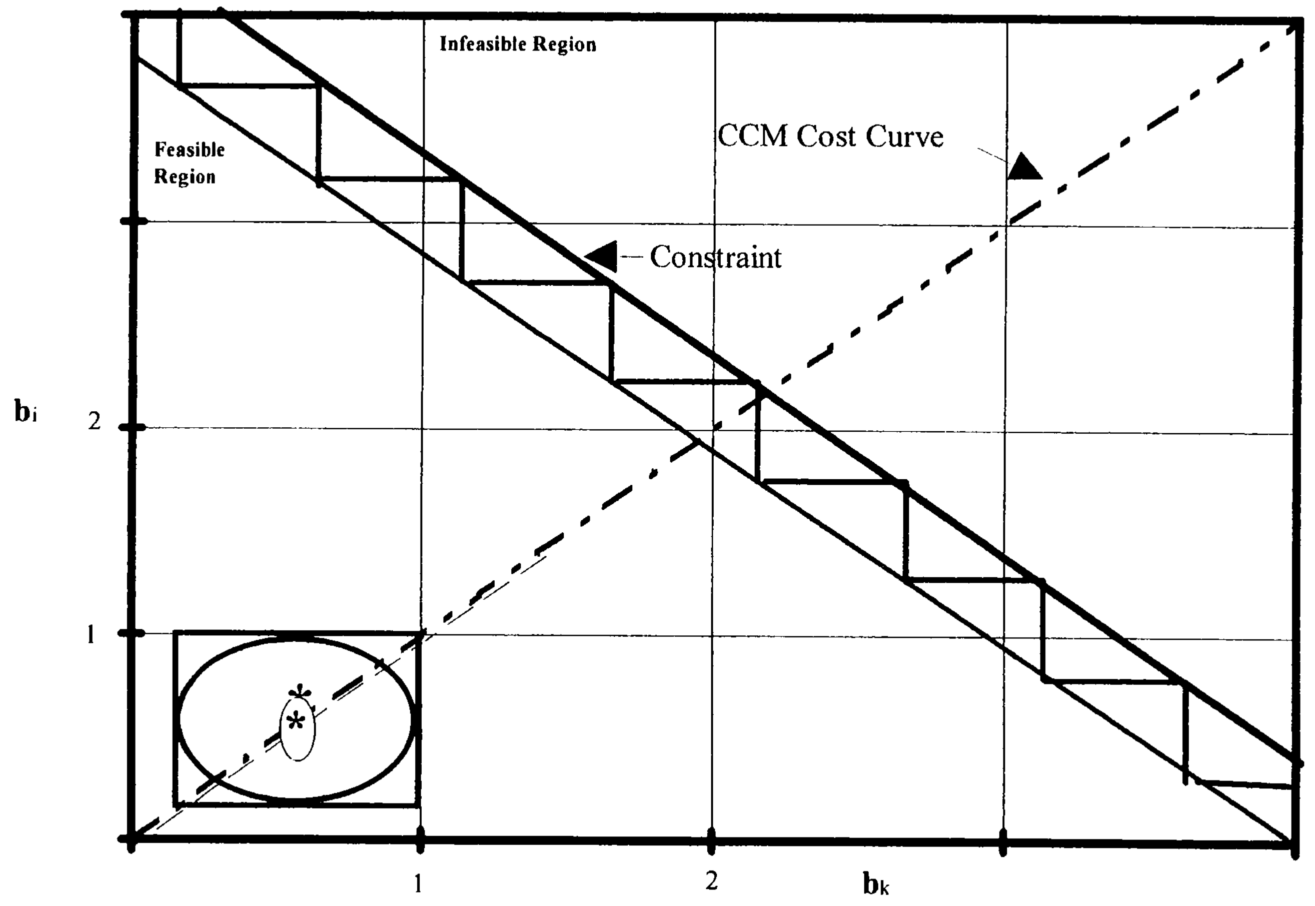


Figure 15

Both Figures 14 and 15 illustrate the $\left(b_l,\ b_k\right)$ planes related to large values of $\boldsymbol{a}$. The fundamental cycle $\boldsymbol{a}$ is considered large if there is at least one $\boldsymbol{i}$ such that $b_{ui} = 1.$

Step 7

Compare the bounds on $\boldsymbol{a}$ and either select a new candidate value for the fundamental cycle or terminate the algorithm.

a. If $a_u \geq a_l,$ then every a satisfying $a_l \leq a \leq a_u$ is potentially an optimum fundamental cycle. To assure that the algorithm produces an optimal solution, each $a, a_l \leq a \leq a_u$, must be either eliminated by using the bounds generated from the structure of the GCM, or used as the fixed fundamental cycle in a procedure that determines an optimum set of $\left\{b_i\right\}$ for a given $\boldsymbol{a}$.

If $a_u < a_l$ then all potentially optimum fundamental cycles have been eliminated which implies that $T_i^* = T_{bi}$ and $Z^* = Z_u$, so terminate the algorithm. Note that it is possible $a_u < 1;$ however, a_l is always greater than 1. Thus, the algorithm will terminate whenever $a_u < 1.$ Otherwise, let

$$\theta = \left\{ i \;\middle|\; T_{li} = T_{ui} \right\} \tag{90}$$

If $\theta = \phi.$ the null set, select a_u as the next candidate fundamental cycle and proceed to step 8. It is conjectured that selecting a_u as the next candidate fundamental cycle reduces the computational requirements of the algorithm. If $\theta \neq \phi$ the dimensionality of the BPM is reduced by the cardinality of θ because for the remainder of the algorithm, the cycle times $T_i^* = T_{li} = T_{ui}$ for all $i \in \theta$. Thus, the optimizations procedures in steps 11 and 12 are performed only with respect to products $i \notin \theta$. If $\theta \neq \phi$ go to step 7.b.

b. For each $i \in \theta$ the lower limit T_{li} is the only possible value for T_i in a schedule $\tilde{T}$ where $Z(\tilde{T}) < Z_u$. This means that

$$T_i^* = T_{li} = b_i^* a^*. \tag{91}$$

Hence, a^* must divide all T_i such that $i \in \theta$. Let

$$\psi = \left[\textit{integer } v \,\middle|\, v \textit{ that divides each } T_i \textit{ where } i \in \theta \textit{ and } a_l \leq v \leq a_u \right].$$

Suppose there are k cycle times T_i where $T_i^* = T_{li}$ then the set ψ is found by first finding

$$g_k = gcd\left(T_i \,\middle|\, i \in \theta\right).$$

A theorem from linear algebra states that v divides all T_i, $i \in \theta$, if and only if v divides g_k. Thus, if $g_k < a_l$ then $\psi = \phi$. If $g_k \geq a_l$ then

$$\psi = \left[\textit{integer } v \,\middle|\, v \textit{ divides } g_k \textit{ and } a_l \leq v \leq a_u \right].$$

If $\psi = \phi$ then all potentially optimum fundamental cycles have been eliminated so set $T_i^* = T_{bi}$ and $Z^* = Z_u$ and terminate the algorithm. Otherwise, set

$$a_u = max_{v\varepsilon\psi}(v) \tag{92}$$

and go to step 8.

Step 8

Find upper and lower limits for the $\{b_i\}$ when $a = a_u$.

a. Because of (52), the limits on T_i found in step 4.a also limit b_i^* when a is fixed. That is, $T_i = a_u b_i$ when $a = a_u$ so

$$T_{li} \leq a_u b_i$$

or

$$b_{li} = \left[T_{li} / a_u\right]^+ \leq b_i^* \tag{93}$$

and

$$b'_{ui} = \left[T_{ui} \; / \; a_u\right]^- \geq b_i^* \tag{94}$$

for $i = 1, 2, \cdots, n$.

b. When $a = a_u$, the lower bounds on the $\{b_i\}$ and (53) provide an additional upper bound on each b_i. Rearranging (53) gives

$$a_u \geq \sum_1^n [\rho_i T_i + s_i]^+ \geq S + \sum_{i=1}^n \rho_i T_i$$

$$1 - S / a_u \geq \sum_1^n \rho_i b_i$$

$$\geq \rho_j b_j + \sum_{i \neq j} \rho_i b_{li}$$

or

$$b''_{uj} = \left\{ \left[1 - S / a_u - \sum_{i \neq j}^n \rho_i b_{li} \right] / \rho_j \right\}^- \geq b_j^* \tag{95}$$

for $i = 1, 2, \cdots, n$. Let $b_{ui} = min\left(b'_{ui}, b''_{ui}\right)$.

Step 9

Compare b_{li} with b''_{ui}. If there is an i such that $b_{li} > b''_{ui}$ when $a = a_u$ then a feasible schedule $\widetilde{T}$ with $Z\left(\widetilde{T}\right) < Z_u$ does not exist when $a = a_u$. This situation is illustrated in Figure 16.

If $b_{li} > b_{ui}$ for any i suppose $a'_u \neq a_u$ is a replacement candidate fundamental cycle. To satisfy $a_l \leq a'_u \leq a_u$ with $a'_u \neq a_u$ clearly a'_u must be less than a_u. However, an inspection of (93) and (95) reveals that b_{li} monotonically increases while b''_{ui} monotonically decreases as a_u is decreased.

BPM Algorithm (b_i, b_k) Plane
Small a, b_{li} Outside Constraints

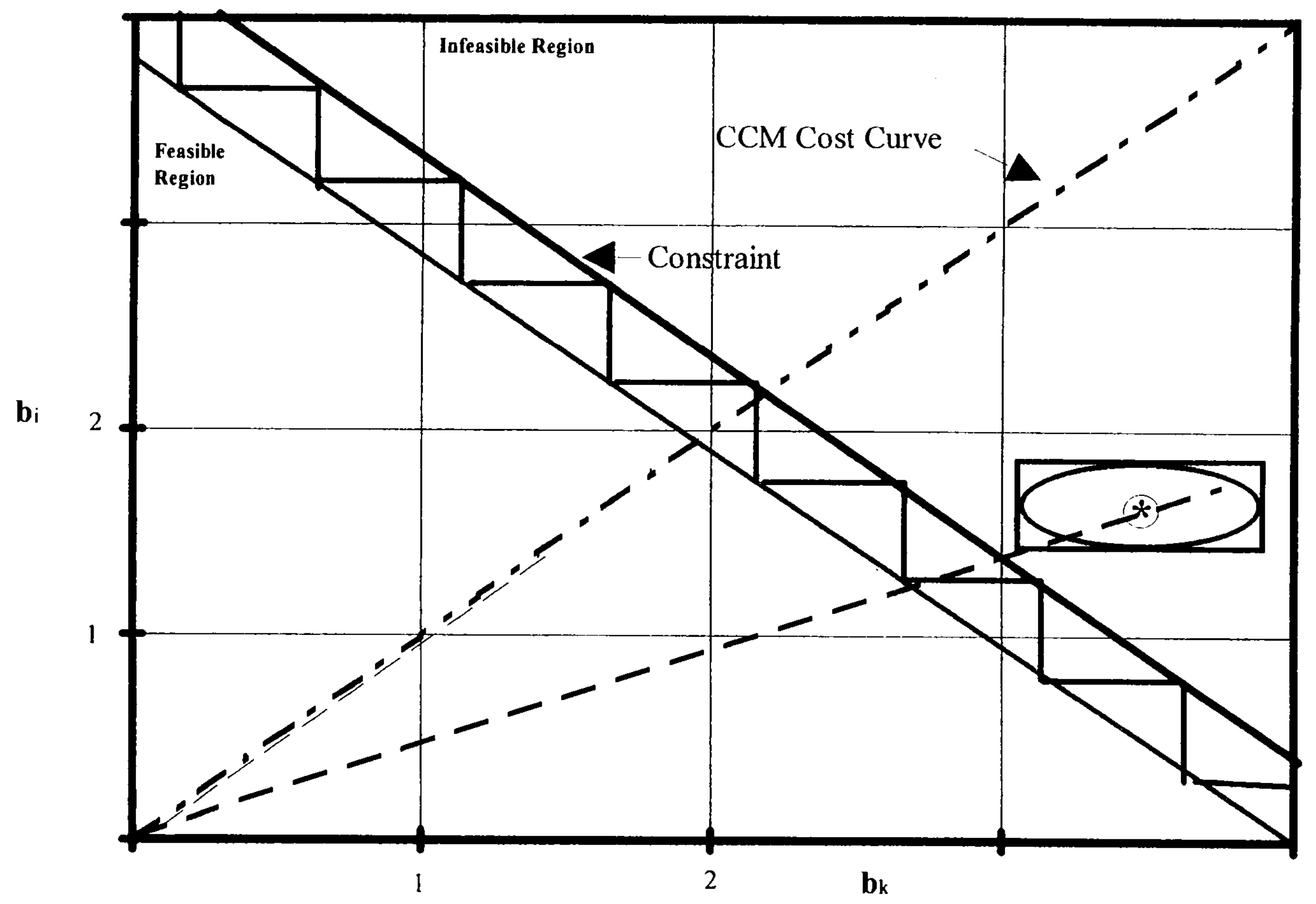


Figure 16

Hence, if $b_{li} > b''_{ui}$ then all candidate fundamental cycles have been eliminated, so the algorithm can be terminated. In this case, set $T_i^* = T_{bi}$ and $Z^* = Z_u$ and terminate the algorithm. Otherwise, go to step 10. Note that it is possible that $b''_{ui} < 1$; however, b_{li} is always greater than one. Thus, the algorithm will proceed to step 10 if and only if $1 \leq b_{li} \leq b''_{ui}$.

Step 10

Compare b_{li} from (93) with b'_{ui} from (94) and let

$$\gamma = \left\{ i \mid b_{li} > b'_{ui}, i = 1, 2, \cdots, n \right\}. \tag{96}$$

If $\gamma = \phi$ go to step 11; otherwise, revise the upper bound on a to

$$a_u = \min_{i \varepsilon \gamma} \left[T_{ui} / b_{li} \right]^- \tag{97}$$

and return to step 7.

If $\gamma \neq \phi$ relationship (97) provides a strictly smaller bound on the optimal fundamental cycle. However, if $\gamma = \phi$ then bounds of this form will not be less than the present value of a_u. To show this, assume that $b_{li} > b'_{ui}$ for some i. This situation is shown in Figure 17.

Use the division algorithm to find

$$T_{li} = w_{li} a_u + v_{li}$$

$$T_{ui} = w_{ui} a_u + v_{ui}$$

BPM Algorithm (b_i, b_k) Plane

$b'_{ui} < b_{li}$

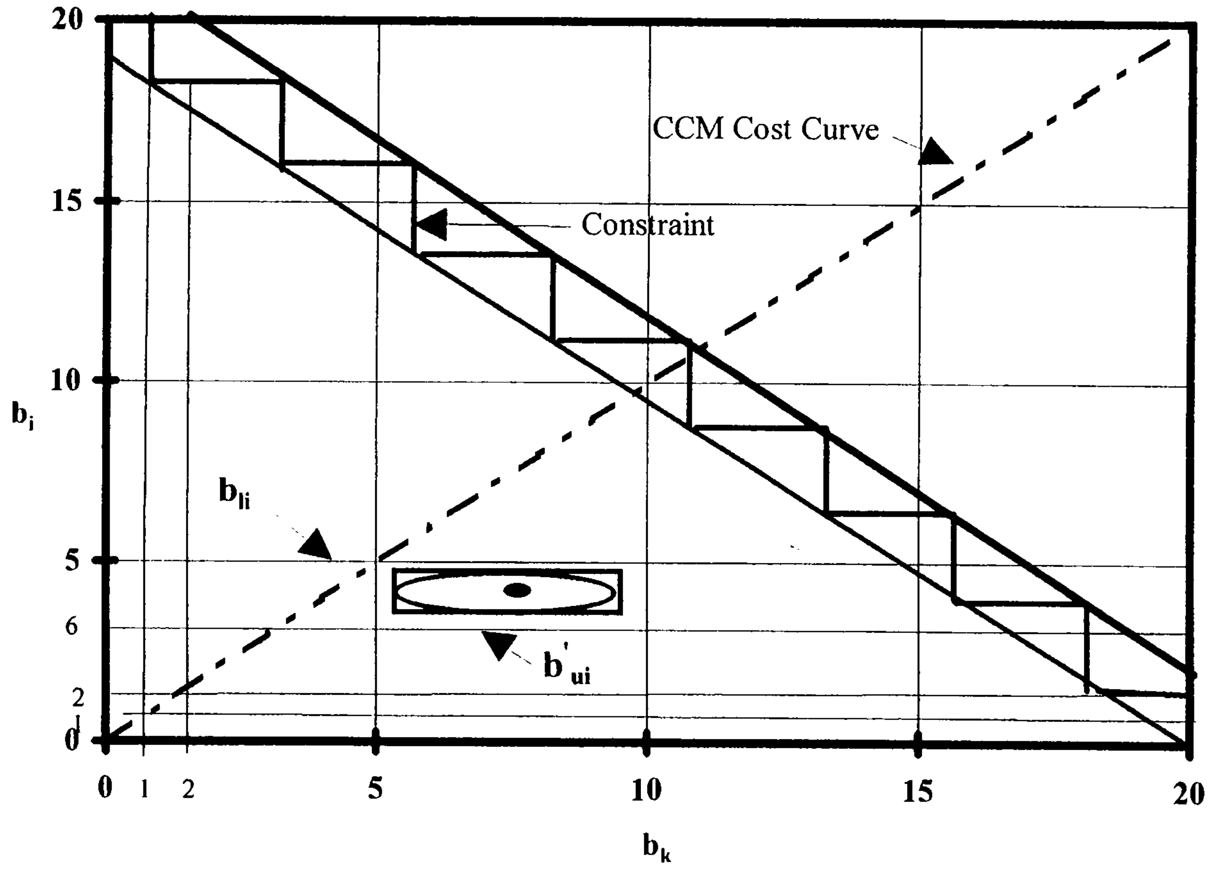


Figure 17

where $w_{li}, w_{ui} \geq 0$ and $0 \leq v_{li}, v_{ui} < a_u$. Suppose that $w_{li} > w_{ui}$ then

$$\begin{aligned} T_{li} &= w_{li}a_u + v_{li} \\ &\geq (w_{ui} + 1)a_u + v_{li} \\ &> w_{ui}a_u + v_{ui} + v_{li} \\ &> T_{ui} \end{aligned}$$

step 5 assumes that $T_{li} \leq T_{ui}$, all i, when step 10 is executed; thus by contradiction $w_{li} \leq w_{ui}$. Now suppose that $w_{li} < w_{ui}$ then

$$b_{li} = [T_{li} / a_u]^+ = [w_{li} + v_{li} / a_u]^+ = w_{li} + 1 \leq w_{ui} = b'_{ui}$$

which contradicts $b_{li} > b'_{ui}$; hence, $w_{li} = w_{ui}$. Therefore $b_{li} > b'_{ui}$ if and only if

$$b_{li} = w_{ui} + 1 = b'_{ui} + 1.$$

Now let a_{ui} be the bound from the ***ith*** item used in (97). Then

$$\begin{aligned} a_{ui} &= [T_{ui} / b_{li}]^- \\ &= [T_{ui}a_u / [(w_{ui} + 1)a_u]]^- \\ &= [T_{ui}a_u / (T_{ui} + a_u - v_{ui})]^- \\ &< a_u. \end{aligned} \tag{98}$$

The strict inequality holds because $\boldsymbol{a_u} > \boldsymbol{v_{ui}}$. Thus, each bound in (97) is smaller than the previous upper bound on $\boldsymbol{a}$.

On the other hand, if $\boldsymbol{b_{li}} \leq \boldsymbol{b'_{ui}}$ then $\boldsymbol{b_{li}} \leq \boldsymbol{w_{ui}}$ with equality only if $\boldsymbol{v_{li}} = \boldsymbol{0}$; thus,

$$
\begin{aligned}
a_{ui} &= \left[T_{ui} / b_{li}\right]^{-} \\
&\geq \left[T_{ui}a_u / w_{ui}a_u\right]^{-} \\
&\geq \left[T_{ui}a_u / \left(w_{ui}a_u + v_{ui}\right)\right]^{-} \\
&= a_u .
\end{aligned}
$$

Therefore, bounds of the form found in (97) will not reduce the present upper bound on $\boldsymbol{a}$ unless $\boldsymbol{b_{li}} > \boldsymbol{b'_{ui}}$ for some for $\boldsymbol{i = 1, 2, \cdots, n}$.

Step 11

The advantages of beginning the algorithm with large values for the fundamental cycle $\boldsymbol{a}$ are obtained primarily in step 11 and 12. There are four parts to step 11. The actions of these parts will be described first then these advantages will be explained. The actions of step 10 assures that $\boldsymbol{b_{li} \leq b_{ui}}$, all $\boldsymbol{i}$. This is necessary condition for actions of step 11 to be consistent.

a. Relax the integer constraint on the $\left\{\boldsymbol{b_i}\right\}$ and let $\boldsymbol{b^*_{arsi}}$ be the unique, real valued multiplier that minimizes $\boldsymbol{Z_i\left(a_u b_i\right)}$ when the ***ith*** product is treated as an unconstrained single-product model with $\boldsymbol{a} = \boldsymbol{a_u}$. It is easily shown that

$$b^*_{arsi} = T^*_{rsi} \;/\; a_u \tag{99}$$

where T^*_{rsi} is given by (68) in step 2.

Let

$$b'_{asi} = \begin{cases} \left(b^*_{arsi}\right)^+, \; \left(b^*_{arsi}\right)^- = 0 \text{ or } Z_i\left[a_u\left(b^*_{arsi}\right)^-\right] > Z_i\left[a_u\left(b^*_{arsi}\right)^+\right] \\ \\ \left(b^*_{arsi}\right)^-, \; \left(b_{arsi}\right)^- \geq 1 \text{ and } Z_i\left[a_u\left(b^*_{arsi}\right)^-\right] \leq Z_i\left[a_u\left(b^*_{arsi}\right)^+\right]. \end{cases} \tag{100}$$

Observe that in case of ties the smaller value is selected for b'_{asi} because the smaller value may be feasible when the larger value is not. Now that b'_{asi} has been defined a useful lemma can be stated.

<u>**Lemma 3**</u>

For any value of a such that $a_l \leq a \leq a_u$

$$b^*_{ai} \leq max\left(b_{li}, \; b'_{asi}\right) \tag{101}$$

for $i = 1, 2, \cdots, n$ where b^*_{ai} is the optimal multiplier for product i when $a = a_u$.

Proof of Lemma 3

One of the advantages of using the $\{b_i\}$ hyperplane is that the left-hand side of constraint (53) is fixed when $a = a_u$. This is not true in the $\{T_i\}$ hyperplane. Clearly the right-hand side of (53)

$$\sum_{i=1}^{n} q_i(T_i) = \sum_{i=1}^{n} q_i(a_u b_i) = S + \sum_{i=1}^{n} (\rho_i a_u b_i)^{+} \tag{102}$$

decreases monotonically as b_i, any i, is decreased while a is held fixed at a_u. Let $\{\tilde{b}_i\}$ be a fixed vector of multipliers such that (53) is satisfied. Suppose that b_k is reduced so that $b_k = \alpha < \tilde{b}_k$ Because of the monotonic behavior of (102)

$$\sum_{1}^{n} q_i(a_u \tilde{b}_i) \geq \sum_{i \neq k}^{n} q_i(a_u \tilde{b}_i) + q_k(a_u \alpha)$$

Thus, if $\tilde{b}_k$ is a component of a feasible vector then so is α where $b_{lk} \leq \alpha \leq \tilde{b}_k \leq b_{uk}$.

The cost function $Z_i(a_u b_i)$ is a strictly convex function of b_i when $a = a_u$. Hence, if there is a vector $\{b_i\}$ that satisfies (53) where $\tilde{b}_k > max(b_{li},\ b'_{asi})$, any k, then $z_a = \sum_{i=1}^{n} Z_i(a_u \tilde{b}_i)$ can be reduced by setting $\tilde{b}_k = max(b_{li},\ b'_{asi})$ and the revised vector will be feasible if $\{\tilde{b}_i\}$ is. Therefore, as stated by lemma 3, $b^*_{ai} \leq max(b_{li},\ b'_{asi})$.

One of the implications of lemma 3 is an enhancement to one of the advantages derived from beginning the algorithm with large values of $\boldsymbol{a}$. This becomes clear in part b.

b. As stated above, step 10 assures that $\boldsymbol{b}_{li} \le \boldsymbol{b}_{ui}$, all $\boldsymbol{i}$. Clearly if there are one or more products such that $\boldsymbol{b}_{li} = \boldsymbol{b}_{ui}$ then the optimal multiplier $\boldsymbol{b}^*_{ai} = \boldsymbol{b}_{li} = \boldsymbol{b}_{ui}$. The dimensionality of the optimization problem is reduced by one for each such product $\boldsymbol{i}$ during the remaining steps of the algorithm. This simplification is enhanced by lemma 3. If there are products $\boldsymbol{i}$ such that $\boldsymbol{b}_{li} \ge \boldsymbol{b}'_{asi}$ then $\boldsymbol{b}^*_{ai} \le \boldsymbol{b}_{li}$. However, $\boldsymbol{b}_{li}$ is a lower bound on $\boldsymbol{b}^*_{ai}$ so that $\boldsymbol{b}^*_{ai} = \boldsymbol{b}_{li}$.

To formalize these observations let

$$\lambda_a = \left\{ i \,\middle|\, b_{li} = b_{ui} \ or\, b_{li} \ge b'_{asi},\ i = m + 1,\ \cdots,\ n \right\} \tag{103}$$

where it is assumed the products are labeled so that the indices of the first m products do not appear in λ_a. Then the optimal multipliers for $i \in \lambda_a$ are

$$b^*_{ai} = b_{li} \tag{104}$$

where $\boldsymbol{i = m + 1,\ \cdots,\ n}$ and $\boldsymbol{0 \le m \le n}$. Note that $\boldsymbol{\theta}$ defined by (90) in step 7 is a subset of λ_a.

The actions of step 10 assure that $\boldsymbol{b}^*_{ai} = \boldsymbol{b}_{li}$, all $\boldsymbol{i}$, is a feasible solution. Hence, if the optimal values of all $\boldsymbol{n}$ multipliers are determined by (102) and (103) set

$$z_a^* = \sum_1^n Z_i(a_u b_{li})$$

and $b_{ai}^* = b_{li}$, all i, and go to step 13. In such cases the dynamic programming procedure of step 12 is not needed. If all but one index is contained in λ_a go to 10.c; otherwise, go to 10.d.

c. When this sub-step is executed there is only one product k such that $b_{lk} < b_{uk}$ and $b_{lk} < b'_{ask}$. If b'_{ask} and b_{li}, $i \neq k$, satisfies (53); that is, if

$$a_u \geq S + \sum_{i \neq k} (\rho_i a_u b_{li})^+ + (\rho_k a_u b'_{ask})^+ \tag{105}$$

then $b_{ak}^* = b'_{ask}$. Otherwise, b_{ak}^* is the largest value of b_k that satisfies (105) because $Z_k(a_u b_k)$ is a strictly convex function of b_k and all candidate values for b_k are strictly less than the integer, single-product Let $b''_{ask} = w$ where the division algorithm is used to obtain

$$\begin{aligned}[\rho_i a_u b_k]^+ &\leq a_u - S - \sum_{i \neq k}^n (\rho_i a_u b_{li})^+ \\ &= w \rho_i a_u + v\end{aligned} \tag{106}$$

where $w \geq 1$ and $0 \leq v < \rho_i a_u$. From (106) it follows that

$$b''_{ask} = \left\{\left[a_u - S - \sum_{i \neq k}^n (\rho_i a_u b_{li})^+\right] / (\rho_k a_u)\right\}^- \tag{107}$$

is the maximum integer that satisfies (105). As stated above, the actions of previous steps assure that $\{b_{li}\}$ satisfies (53) so b''_{ask} in (107) is well-defined. To summarize set

$$\begin{aligned} b^*_{ak} &= min\left(b'_{ask},\ b''_{ask}\right), \\ b^*_{ai} &= b_{li},\ i \neq k \\ z^*_a &= \sum_1^n Z_i\left(a_u b^*_{ai}\right) \end{aligned} \tag{108}$$

and go to set 13. Again, the dynamic programming procedure of step 12 is not required if the indices of all but one product is contained in λ_a.

d. When this sub-step is executed there are $m \geq 2$ products such that $b_{lk} < b_{uk}$ and $b_{lk} < b'_{ask}$. If $b_{uk} < b'_{ask}$ for $k \notin \lambda_a$ then clearly the optimal integer, single-product multiplier for product k subject to the constraint $b_{lk} \leq b_{ak} \leq b_{uk} < b'_{ask}$ is b_{uk}. Thus, set

$$b^*_{asi} = min\left(b'_{asi},\ b_{ui}\right) \tag{109}$$

for $i \notin \lambda_a$.

The medium region of values of a is defined as those values of a where

$$a_u \geq \sum_{i \in \lambda_a} q_i\left(a_u b_{li}\right) + \sum_{i \notin \lambda_a} q_i\left(a_u b^*_{asi}\right). \tag{110}$$

This region is shown in Figure 18. Note that the regions of a may overlap.

BPM Algorithm (b_i, b_k) Plane
Medium a

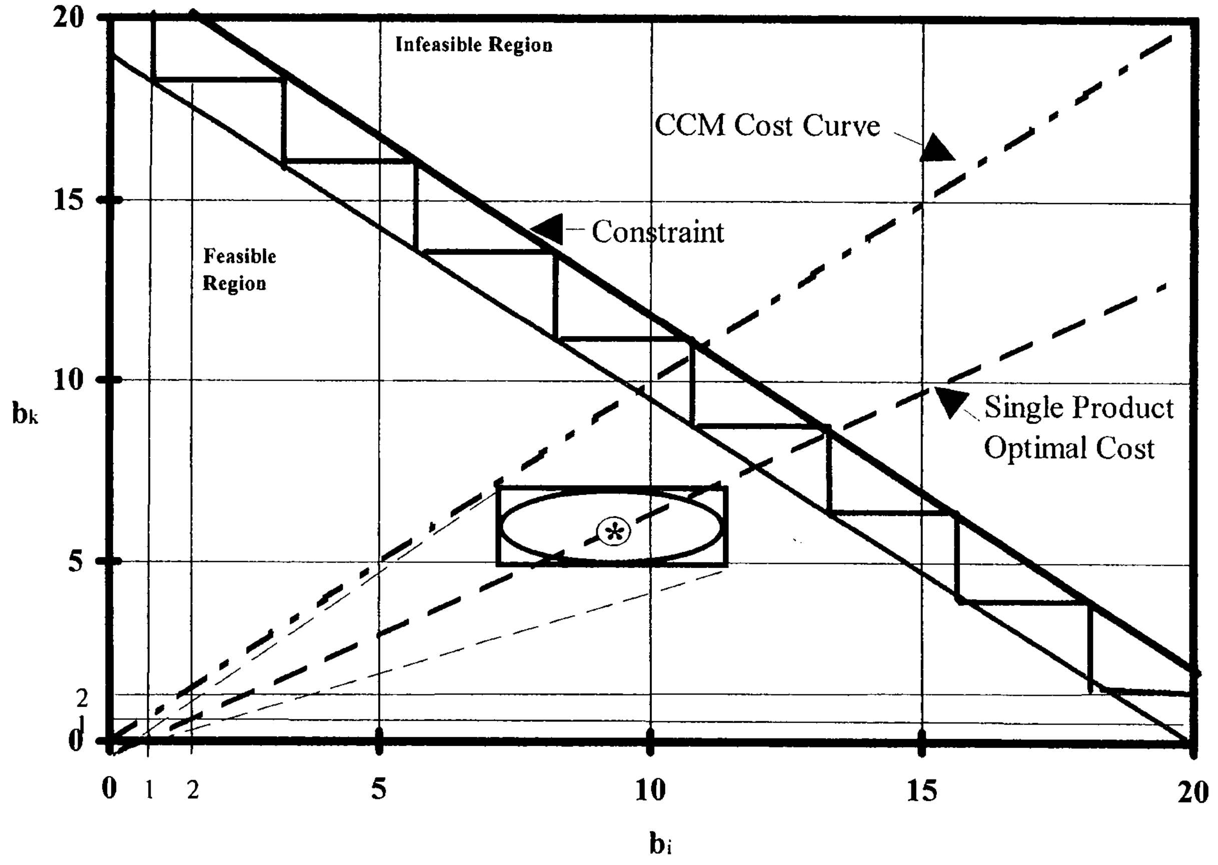


Figure 18

Clearly,

$$z'_a = \sum_{i \in \lambda_u} Z_i(a_u b_{li}) + \sum_{i \notin \lambda_u} Z_i(a_u b^*_{asi}) \tag{111}$$

is a lower bound on the optimal cost z^*_a where $b_{li} \le b_i \le b_{ui}$, all i. Thus, if a_u is in the medium region of a values; that is, if (110) is true, then $z^*_a = z'_a$ where z'_a is given by (111). Thus, if (110) is satisfied set $z^*_a = z'_a$, $b^*_{ai} = b_{li}$, $i \in \lambda_a$, and $b^*_{ai} = b^*_{asi}$, $i \notin \lambda_a$, and go to step 13. Again, the dynamic programming procedure is not required when a_u is in the medium region of a values.

The optimal multipliers b^*_{ai}, $i \in \lambda_a$, are determined by (104); however, if (110) is not valid then the optimal set of multipliers b^*_i for $i \notin \lambda_a$, must be determined by the dynamic programming procedure of step 12. The procedure of step 12 can be simplified by applying lemma 3 to each b_i where $i \notin \lambda_a$. That is, the results of lemma 3 mean that if b'_{ui} is first set to the present value of b_{ui}, $i \notin \lambda_a$, then the revised upper bounds

$$b_{ui} = min(b'_{ui}, b'_{asi}) \tag{112}$$

are improved upper bounds on the multipliers b_i, $i \notin \lambda_a$. Hence, if (110) is not valid then set b_{ui}, $i \notin \lambda_a$ according to (112) and execute step 12 to solve the $m \ge 2$ dynamic programming problem to find b^*_{ai} for $i \notin \lambda_a$.

Advantages of Beginning with Large a_u

The advantages of beginning the algorithm with a equal to its maximum value a_u are manifested in steps 11 and 12 over the regions of large a and medium a. These advantages are:

1. If a is set to its maximum possibly optimal value then the range of potentially optimal b_i, all i, is at its smallest for a given cost upper bound Z_u. This effect is shown by Figure 19 where two limit boxes are shown – one for large a and one for medium a. Hence, during the first iterations of the algorithm the complexity of the dynamic programming problem of step 12 is minimized by starting the algorithm with the maximum value of a. Also, the smaller ranges of b_i, all i means that the optimal $\left\{b_{ai}^*\right\}$ will be found more often by the simple decision rules of step 11.

2. The dimensionality of the optimization problem is more likely to be less than n when a is large or medium value. To explain this second advantage, note that the actions of the previous steps assure that $b_{li} \leq b_{ui}$, all i, before step 11 is executed. It is possible that $b_{li} = b_{ui}$ for one or more i. In fact, as shown by Figure 14, during the first iteration of the algorithm there is at least one i such that $b_{li} = b_{ui}$. In such cases the optimal multipliers are set to

BPM Algorithm (b_i, b_k) Plane
Advantages of Large a

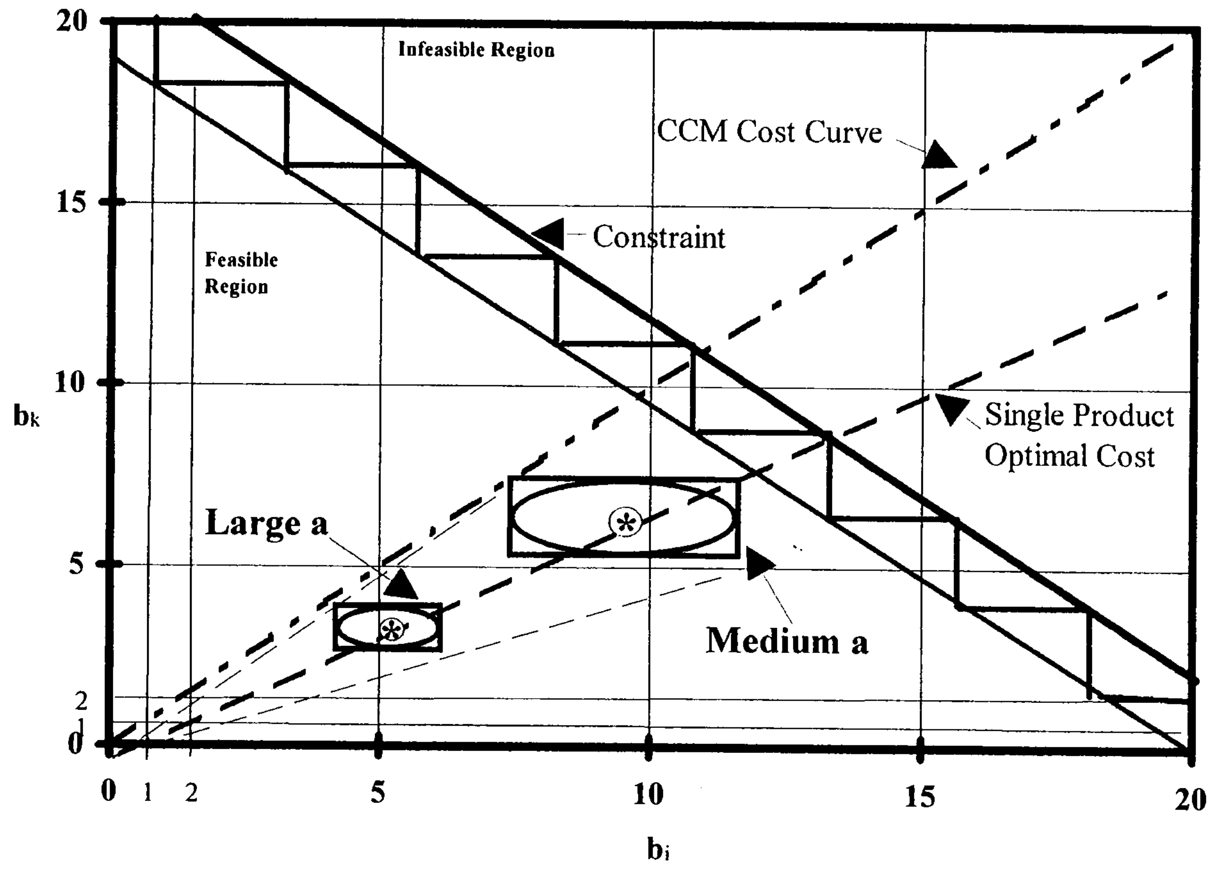


Figure 19

$$b^*_{ai} = b_{li} = b_{ui}$$

during the optimization procedures of steps 11, 12, and 13. Of course, the dimensionality of the optimization problem is reduced by one for each product i such that $b_{li} = b_{ui}$. This advantage is enhanced by the results of lemma 3, because the dynamic programming procedure of step 12 is must only find b^*_{ai} for $i \notin \lambda_a$.

3. In the large and medium ranges of a where there is at most one i such that $i \notin \lambda_a$ an optimal set of multipliers is quickly determined is step 11. Thus, for such cases, the dynamic problem procedure of step 12 does not have to be executed.

Step 12

This step is executed only if there are $m \geq 2$ products indices i such that $i \notin \lambda_a$. Use a dynamic programming algorithm to find an optimal vector $\left\{b^*_{ai}\right\} = \left(b^*_{a1}, b^*_{a2}, \cdots, b^*_{am}\right)$, for $i \notin \lambda_a$, which minimizes (51) subject to (52), (53), $a = a_u$, $b_{li} \leq b^*_{ai} \leq b_{ui}$, and $b^*_{ai} = b_{li}$ for $i \in \lambda_a$ where b^*_{ai} all i are integers. Then go to step 13.

The actions of the previous steps provide the prerequisites for a coherent dynamic programming procedure. That is, when step 12 is executed the previous steps give assurance that

1. $b^*_{asi} \geq b_{ui} > b_{li} \geq 1,\ i \notin \lambda_a,$

2. $\{b_{li}\},\ i = 1,\ 2,\ \cdots,\ n,$ satisfies constraints (52) and (53),

3. $b_{ai} = b^*_{asi},\ i \notin \lambda_a,$ and $b_{ai} = b_{li},\ i \in \lambda_a,$ is not a feasible vector of multipliers, and

4. $n \geq m \geq 2$

where $(n - m)$ is the cardinality of λ_a. Because of the first assurance there are two or more feasible values for each $b_i,\ i \notin \lambda_a$. The second assurance means there is at least one feasible solution to the dynamic programming problem. The third assurance, in conjunction with the first, means that the optimal solution to the dynamic programming problem will be "tightly" bound. Tightly bound means that if the integer constraints on $b_i,\ i \notin \lambda_a,$ and q_i are relaxed then the optimal multipliers b^*_{ai} will lie in the linear hyperplane defined by

$$a_u = S + \sum_{i \in \lambda_u} \rho_i a_u b_{li} + \sum_{i \notin \lambda_u} \rho_i a_u b_{ai}. \tag{113}$$

This hyperplane for $m = 2$ is shown in Figure 7 as the upper limit to the integer version of constraint (53). The last assurance means that there will be at least two stages to the dynamic programming procedure.

The dynamic programming formulation of the problem when $a = a_u$ is to find

$$Z^*_a = min_{b_i} Z\left(\{a_u b_i\}\right) = min_{b_i} \sum_{i \notin \lambda_a} \left[c_i / (a_u b_i) + H_i(a_u b_i)\right] \tag{114}$$

subject to

$$a_u \geq \sum_{i=1}^{n} q_i = S + \sum_{i=1}^{n} \left[\rho_i a_u b_i\right]^{+} \tag{115}$$

Let

$$Y_a = a_u - S - \sum_{i \in \lambda_a} \rho_i a_u b_{li} \tag{116}$$

then constraint (115) becomes

$$Y_a \geq \sum_{i \notin \lambda_a} \left[\rho_i a_u b_i\right]^{+} \tag{117}$$

where

$$1 \leq b_{li} \leq b_i \leq b_{ui} < \infty \tag{118}$$

for $i \notin \lambda_a$.

The dynamic programming formulation is completed by defining the state variables and return functions in the usual way. The state variables are

$$\begin{aligned}
Y_a &\leq X_m \\
x_{m-1} &= x_m - \left[\rho_u a_u b_u\right]^{+} \\
&\vdots \\
x_1 &= x_2 - \left[\rho_2 a_u b_2\right]^{+} \\
x_0 &= x_1 - \left[\rho_1 a_u b_1\right]^{+} \\
x_0 &\geq 0.
\end{aligned} \tag{119}$$

where $\boldsymbol{y_a}$ is given by (116).

The return functions are

$$\boldsymbol{f_1}(\boldsymbol{x_1}) = \mathbf{min}_{\boldsymbol{b_1}}\left[(\boldsymbol{c_1} / \boldsymbol{a_u}) / \boldsymbol{b_1} + (\boldsymbol{H_1 a_u})\boldsymbol{b_1}\right] \tag{120}$$

and

$$\boldsymbol{f_i}(\boldsymbol{x_i}) = \mathbf{min}_{\boldsymbol{b_i}}\left\{\left[(\boldsymbol{c_i} / \boldsymbol{a_u}) / \boldsymbol{b_i} + (\boldsymbol{H_i a_u})\boldsymbol{b_i}\right] + \boldsymbol{f_{i-1}}(\boldsymbol{x_i} - \boldsymbol{\rho_i a_u b_i})\right\} \tag{121}$$

subject to

$$\boldsymbol{b_i} \leq \left[\boldsymbol{x_i} / \boldsymbol{\rho_i a_u}\right]^+ \tag{122}$$

and

$$\boldsymbol{b_{li}} \leq \boldsymbol{b_i} \leq \boldsymbol{b_{ui}}$$

for $\boldsymbol{i = 1, 2, \cdots, m}$.

When the dynamic programming procedure has determined the optimal $\boldsymbol{b^*_{ai}}$ for $\boldsymbol{i} \notin \boldsymbol{\lambda_a}$ set $\boldsymbol{b^*_{ai}} = \boldsymbol{b_{li}}$ for $\boldsymbol{i} \in \boldsymbol{\lambda_a}$ and $\boldsymbol{z^*_a} = \sum_1^n \boldsymbol{Z_i}(\boldsymbol{a_u b^*_{ai}})$ and go to step 13.

Step 13

Reduce $\boldsymbol{a_u}$, the upper limit on $\boldsymbol{a}$, by one. Then with $\boldsymbol{b_i}$ fixed at $\boldsymbol{b^*_{ai}}$, all $\boldsymbol{i}$, minimize the objective function with respect to $\boldsymbol{a}$. If this produces an improved upper bound on $\boldsymbol{Z^*}$, replace the old best solution with this one and go to step 4. Otherwise, discard $\{b_{ai}^*\}$ and go to step 7.

The algorithm is finite because the integer fundamental cycle a_u is reduced by one as the first part of step 13. In the worst case, step 13 will be performed $(a_u - a_l + 1)$ times where a_l and a_u are the initial limits on the fundamental cycle a.

With $b_i = b_{ai}^*$, all i, the minimum of (123) with respect to a is found by applying the procedure of step 1 to the modified Hanssman model where

$$z_b^* = min_a\, Z_b(a) = min_a(c_b \,/\, a + H_b a) \tag{123}$$

subject to

$$a \geq S + \sum_{i=1}^{n} [\rho_i b_i a]^+ \tag{124}$$

and

$$c_b = \sum_1^n c_i \,/\, b_{ai}^* \text{ and } H_b = \sum_1^n H_i b_{ai}^*. \tag{124}$$

Let a_b^* be the optimum value of a found by these procedures. Then if $z_b^* < Z_u$ set $Z_u = z_b^*, T_{bi} = a_b^* b_{ai}^*,\ \ i = 1, 2, \cdots, n$, and go to step 4. Otherwise discard a_b^* and $\{b_{ai}^*\}$ and go to step 7. Note that $z_a^* \geq z_b^*$ because both are values of the convex function $Z_b(a)$ with b_i fixed at b_{ai}^*, all i.

SUMMARY OF ALGORITHM

The BPM algorithm has many details; however the overall concept is straightforward. The primary tenet of the algorithm is a process of elimination based on the fundamental cycle $\boldsymbol{a} = \boldsymbol{gcd}\left(\left\{T_i\right\}\right)$.

The starting point of the algorithm is to find lower and upper bounds of the optimal cost z^*. These cost bounds are then used to determine lower and upper bounds on each cycle time $\boldsymbol{T_i}$, on the fundamental cycle $\boldsymbol{a}$, and on each multiplier $\boldsymbol{b_i}$. At each iteration of the algorithm these bounds on the model variables must be satisfied by any schedule $\left\{T_i\right\}$ that has a strictly lower cost than the present upper bound $\boldsymbol{Z_u}$ on z^*. Hence, if any of the these bounds are inconsistent the algorithm is terminated because there is no feasible schedule with cost lower than $\boldsymbol{Z_u}$.

Once the initial upper bound is computed in step 1 the upper bound on cost $\boldsymbol{Z_u}$ is the cost of a feasible schedule $\left\{\boldsymbol{T_{bi}}\right\}$ throughout the algorithm. This best schedule $\left\{\boldsymbol{T_{bi}}\right\}$ is saved by the algorithm until another schedule is found with strictly lower cost. If an improved schedule is found $\left\{\boldsymbol{T_{bi}}\right\}$ is set to the improved schedule and the upper bound on the cost $\boldsymbol{Z_u}$ is reduced to the cost of the improved schedule.

The driving force is the process of elimination on the fundamental cycle mentioned above. During the first iteration of the algorithm initial lower and upper bounds $\boldsymbol{a_l}$ and $\boldsymbol{a_u}$ are placed on the fundamental cycle $\boldsymbol{a}$. The algorithm begins with $\boldsymbol{a} = \boldsymbol{a_u}$ the greatest possible value of $\boldsymbol{a}$ and works downward. At each major iteration of the algorithm an optimal set of multiplier $\{b_i\}$ is found by fixing $a = a_u$ where a_u is the candidate fundamental cycle of that iteration. If this results in an improved schedule the best schedule is updated. In any event, the present value of a_u can be eliminated so the upper bound a_u is reduced by 1 before beginning another iteration. This process continues until each value in the interval $[a_l, a_u]$ is either eliminated by the algorithm as non-optimal or used as a candidate optimal fundamental cycle. To provide a more detail summary of the BPM algorithm the steps of the algorithm are divided into the following four stages.

Stage 1

The first stage of the algorithm is to:

Step 1

Use Hanssman's CCM to obtain an initial upper bound, Z_u, on z^*. Save the CCM optimal schedule as the initial best schedule $\{T_{bi}\}$.

Step 2

Use the single-product EMQ model to find an initial lower bound on z^*, the optimal average cost per unit time. If these optimal single-product cycle times are feasible the algorithm is terminated with $T_i^* = T_{si}^*$, all i.

Step 3

Use the results of steps 1 and 2 to find an initial upper bound on the fundamental cycles $\boldsymbol{a}$ of the BPM.

The CCM optimal schedule is an initial feasible schedule for the GCM as well as the BPM. Throughout the remainder of the algorithm Z_u is changed only when another feasible schedule $\{T_i\}$ is found such that $Z(\{T_i\}) < Z_u = Z(\{T_{bi}\})$. Hence, at each step of the algorithm after step 2 Z_u is the total average cost of the best feasible schedule, $\{T_{bi}\}$, found so far. The algorithm terminates when it is shown that the present best schedule is optimal.

The computations required for the first two steps are straightforward calculus problems that are easily solved. Step 3 merely requires comparisons of previously computed values.

The stage 1 steps are executed only once for a given BPM problem. The remaining steps are performed iteratively until there are no feasible schedules $\{T_i\}$ such that $Z_u > Z(\{T_i\})$. When the algorithm terminates the optimal cost $z^* = Z_u$ and the optimal schedule is $\{T_i^*\} = \{T_{bi}\}$.

STAGE 2

This stage of the algorithm places upper and lower bounds on certain model variables and attributes. The steps of this stage are:

Step 4

Use the present lower bounds $\{T_{li}\}$ on $\{T_i\}$ to compute, Z_l, a new lower bound on z^*. This new lower bound on z^* is used to either terminate the algorithm or to compute new lower and upper bounds on the optimal cycle times $\{T_i^*\}$.

Step 5

Use the lower bounds on the cycle times found in step 4 to find a lower bound on the length of the fundamental cycle a. This lower bound is used to find a new lower bound on the cycle times. If the bounds found in steps 4 and 5 are inconsistent the algorithm is terminated because the present best schedule is optimal. Otherwise, the stage 2 steps are repeated until the algorithm terminates or there has been no increase in the lower bound of any $T_i, i = 1, 2, \cdots, n$. This iteration in stage 2 is repeated each time there is a decrease in the upper bound on z^*.

Stage 3

This stage finds a revised upper bound on the length of the fundamental cycle a and upper and lower bounds on the optimal multipliers $\{b_i\}$. The bounds on the $\{b_i\}$ are functions of a. The steps of stage 3 are steps 6 through 10.

Step 6

Uses the present upper bound on the cycle time $\{T_i\}$ to find two new upper bounds on the fundamental cycle $\boldsymbol{a}$. If these new bounds are less than the present value of $\boldsymbol{a_u}$ then $\boldsymbol{a_u}$ is set to the minimum of these new bounds.

Step 7

Check whether the bounds on $\boldsymbol{a}$ are inconsistent. If so the algorithm is terminated because the present best solution is optimal.

If the algorithm is not terminated the algorithm checks whether there are cycle times $\boldsymbol{T_i}$ that are restricted to just one possible value because $\boldsymbol{T_{li}} = \boldsymbol{T_{ui}}$. Clearly the fundamental cycle of any schedule must be an integer factor $\boldsymbol{v}$ of all cycle times $\boldsymbol{T_i}$ of that schedule. Hence, if there is a set of cycle times where $\boldsymbol{T_{li}} = \boldsymbol{T_{ui}}$ then $\boldsymbol{a}$ must divide all such cycle times. These factors must also satisfy $\boldsymbol{a_l} \leq \boldsymbol{v} \leq \boldsymbol{a_u}$. If there are one or more $\boldsymbol{T_i}$ restricted to one value and none of their common factors are between the lower and upper bounds on $\boldsymbol{a}$ then the algorithm is terminated. Otherwise, the upper bound on $\boldsymbol{a}$ is revised by setting $\boldsymbol{a_u}$ to the maximum over the possible common factors $\boldsymbol{v}$. When $\boldsymbol{T_{li}} = \boldsymbol{T_{ui}}$ for some $\boldsymbol{i}$ this action of the algorithm greatly reduces the number of possible values of $\boldsymbol{a}$ that must be examined by the algorithm.

If the algorithm is not terminated by either of the two tests of this step the $\boldsymbol{a_u}$ is selected as the next candidate fundamental cycle and the algorithm proceeds to the next step.

Step 8

Compute a lower bound and two potential upper bounds on the multipliers $\{b_i\}$ given that $a = a_u$.

Step 9

Compare the lower bound b_{li} on each b_i with the second upper bound b''_{ui} found in step 8. If these bounds are inconsistent for any i the algorithm is terminated because the present best solution is optimal.

Step 10

Compare the lower bound b_{li} with the first upper bound b'_{ui} found in step 8. If these bounds are inconsistent for any i the upper bound on the fundamental cycle a is revised and the algorithm returns to step 7. If these bounds on all b_i are consistent then the upper bounds on $\{b_i\}$ are revised and the algorithm proceeds to the next step.

The actions of steps 7 through 10 are repeated until either the algorithm is terminated or the bounds on $\{b_i\}$ found in step 8 are consistent.

Stage 4

In this final stage of the algorithm a set of multipliers $\{b^*_{ai}\}$ is found that is optimal when $a = a_u$. Then the multipliers are held fixed at $\{b_i\} = \{b^*_{ai}\}$ while an optimal value of a is found for that set of multipliers. Then a_u is reduced by 1 to prepare for the next iteration.

If these steps yield an improved schedule the present best solution is updated to equal this improved schedule and the algorithm returns to step 4. Otherwise, the algorithm returns to step 7. The steps of stage 4 are:

Step 11

For each $\boldsymbol{i}$ find an optimal integer multiplier $\boldsymbol{b}^*_{asi}$ for the constrained, single-product EMQ model when the fundamental cycle is held fixed at $\boldsymbol{a}_u$. If these $\left\{\boldsymbol{b}^*_{asi}\right\}$ are feasible in the $\boldsymbol{n}$-product BPM then it is not necessary to perform the dynamic programming routine of step 12. In this case the algorithm skips directly to step 13. Otherwise, the upper bounds on the multipliers are updated by setting $\boldsymbol{b}_{ui} = \boldsymbol{b}^*_{asi}$ before proceeding to step 12.

Step 12

Use a standard dynamic programming algorithm to find an optimal set of integer multipliers $\left\{\boldsymbol{b}^*_{ai}\right\}$ for the BPM when $\boldsymbol{a} = \boldsymbol{a}_u$.

Step 13

First reduce $\boldsymbol{a}_u$ by one. Then with $\left\{\boldsymbol{b}_i\right\}$ fixed at $\left\{\boldsymbol{b}^*_{ai}\right\}$ find the optimal $\boldsymbol{a}$ for the resulting constrained CCM. If this results in a lower cost schedule replace the old best solution with this one and go to step 4. Otherwise, return to step 7.

The steps of stage 4 are the core of the algorithm. Stage 4 must be repeated for every possible value of the fundamental cycle $\boldsymbol{a}$. Suppose that $\left(\boldsymbol{a}_l, \boldsymbol{a}_u\right)$ are the bounds on $\boldsymbol{a}$ after steps 1 through 5 are performed for the first time. The worst case then is that the stage 4 steps will have to be performed a maximum of $\left(a_u - a_l + 1\right)$times.

PROOF OF OPTIMALITY

The assertion which must be proved is that the algorithm produces an optimal integer BPM schedule. That is, that the algorithm finds an optimal schedule of integer cycle times, over all schedules with integer cycle times, for the model defined by assumptions (a) through (k) of chapter 1 where the optimization is subject to Bomberger's BPM constraints (52) and (53) and b_i, all i, are restricted to integer values. Furthermore, the algorithm only requires a finite number of numerical operations. The body of the proof is given under each step of the algorithm so only a skeleton proof which fits the pieces together will be presented here.

Bomberger's formulation does not restrict the cycle times or the fundamental cycle to integer values. In most cases, the optimum solution over integer cycle times will be non-optimum for the general model of non-integer cycle times. But, from a practical point of view, there is a limit to the precision with which items can be scheduled. Also, the model parameters can be easily adjusted so that time is measured in any desired unit. Thus, it is not a practical restriction to require integer cycle times. (Bomberger resorted to integer cycles when he solved his example problems).

It may seem that restricting the fundamental cycle to integer values will unnecessarily restrict the BPM beyond that imposed by requiring $\left\{T_i\right\}$ and $\left\{b_i\right\}$ to be integers. If so this could cause the algorithm to sometimes miss an optimum solution of the BPM. But, suppose there is a solution with $T_i = ab_i$ where a is non-integer and T_i, b_i are integers, all i. An irrational a requires T_i or b_i to be irrational but both these variables are integers. Thus, without loss of generality, assume a is rational so that $a = x / y$ where x, y are relative prime integers. If $T_i = (x / y)b_i$, all i, are integers, then y must divide each b_i so $T_i = xb_i'$, all i, where each $b_i' = (b_i / y)$ and x are integers. Hence, requiring an integer fundamental cycle does not further restrict the model defined in the assertion.

If the algorithm terminates at step 3, then clearly an optimal solution is produced within a finite number of operations. For the remainder of the algorithm, optimality follows from the observations that:

a. The cost upper bound, Z_u, always corresponds to a known feasible schedule,

b. The bounds on the cycle times, the fundamental cycle, and b_i, all i, are designed never to exclude a schedule with a lower cost than the current upper bound on cost,

c. No value of the fundamental cycle a is eliminated unless it is shown that any schedule where $T_i = ab_i$, all i, does not have smaller cost than the current upper bound cost, and

d. Either the solution to the constrained, single-product models or the dynamic programming algorithm produces an optimum set of $\{b_i\}$ for a fixed fundamental cycle.

Thus, if the initial solution is not optimum and the algorithm is finite, eventually an optimum vector $\{\boldsymbol{b}_i^*\}$ must be found at stage 4 of the algorithm. Once these optimal vectors are found the corresponding optimal $\boldsymbol{a}$ is found in the last step of the algorithm.

To prove that the algorithm is finite, observe that

a. The initial upper and lower bounds on the optimal cost and optimal cycle times are finite,

b. The sequence of lower bounds on $\boldsymbol{a}^*$ is monotonically increasing,

c. The sequence of upper bounds on $\boldsymbol{a}^*$ is strictly monotonically decreasing,

d. There is at most a finite number of steps between each reduction of the upper bound on $\boldsymbol{a}$, and

e. Each step of the algorithm requires only a finite number of operations.

Thus, the algorithm will terminate after a finite number of operations.

CHAPTER 5

TWO-PRODUCTS MODEL

PURPOSE

The general GCM theorem 1 on feasibility leads to several important secondary results. For instance, this chapter will provide the proof for a useful corollary to the general theorem for the special two-products case. This two-products corollary shows that when $\boldsymbol{n = 2}$ an optimal solution to Bomberger's [4] more restricted BP model is an optimal solution to the GCM. More importantly, the two product corollary provides necessary and sufficient conditions for schedule feasibility that are independent of the delay times. That is, these conditions are stated solely in terms of the cycle times $\{\boldsymbol{T_i}\}$. This means that when $\boldsymbol{n = 2}$ whether or not a given set of $\{\boldsymbol{T_i}\}$ is feasible can be determined independent of the delay times $\{\boldsymbol{d_i}\}$. In chapter 7 and 8 the general theorem will be used to extend these two-products delay independent feasibility conditions to delay independent conditions for models where $\boldsymbol{n = 3}$ and $\boldsymbol{n = 4}$.

As stated above the two-products corollary shows that the algorithm for the BPM given in chapter 4 will find optimal schedules when two-products are produced on a common facility. The two-products model is rather limited in scope; however, the two-products corollary will play an important role in the general optimization algorithm.

That is, the BPM algorithm will be applied to all possible two-products models to establish lower bounds on cost in the general optimization algorithm for the $\boldsymbol{n}$-products case of the GCM.

TWO-PRODUCT DELAY INDEPENDENT CONDITION

This section will present theorem 3 that gives delay-independent necessary and sufficient feasibility conditions for the two-products case of the GCM. Theorem 3 is actually a corollary to the general theorem 1 discussed in chapter 3.

Theorem 3 on Conditions for Two-Products GCM

If $n = 2$ let $\tilde{T}' = (T_1, T_2)$ be a vector of given cycle times for the two-products where $\tilde{T}'$ is the transpose of $\tilde{T}$. If the hypothesis of theorem 1 is valid then a feasible schedule $\Omega = (\tilde{T}, \tilde{d})$ exists if and only if

$$a \geq q_1 + q_2 \tag{126}$$

where

$$a = gcd(T_1, T_2), \tag{127}$$

$$T_1 = ab_1,\ T_2 = ab_2, \tag{128}$$

$b_1, b_2 \geq 1$, and a, b_1, b_2 are integers. Furthermore, if (126) is satisfied then

$$d_1 = 0, \; d_2 = q_1 \tag{129}$$

is a feasible schedule.

Note that by definition q_1, q_2 are integers; thus, the results of (129) will be integer delay times.

Proof of Theorem 3

There is only one pair of products in the two-products case so conditions (22), (23), (26), and (27) of the general theorem 1 can be translated to (127), (128), and

$$l \geq q_1 \tag{130}$$

and

$$l \leq a - q_2 \tag{131}$$

where

$$d_2 - d_1 = ka + l \tag{132}$$

$k \geq 0,$ and $0 \leq l < a.$

Because of assumption (*i*) of the GCM definition, time zero can be set at any point relative to a schedule Ω. In particular, time zero can be set to the start of any use period of either product without loss of generality. Hence, to simplify the problem somewhat, time zero is set to the start of a use period of product 1. This means that $d_1 = 0$ so (132) becomes

$$d_2 = ka + l. \tag{133}$$

Whether conditions of theorem 3 are sufficient to assure schedule feasibility will be examined first. Suppose (126) is satisfied and let

$$d_2 = q_1 < q_1 + q_2 \leq a. \tag{134}$$

this inequality shows that $k = 0$ in (133) so that

$$d_2 = l.$$

Thus, $l = q_1$ and

$$l + q_2 = q_1 + q_2 \leq a.$$

This shows that if the conditions (126), (127), and (128) of theorem 3 are satisfied then (130) and (131) are satisfied when $d_1 = 0$ and $d_2 = q_1$. Therefore, by theorem 1, this schedule is feasible. Thus, the conditions of theorem 3 are sufficient to assure that a feasible schedule exists. That these conditions are necessary follows immediately from lemma 4.

Lemma 4 on Minimum Separation

Suppose that products 1 and 2 are any two products of a set of $n \geq 2$ products. Furthermore, suppose that (T_1, T_2) are given cycle times for these two products. Without loss of generality assume that $d_1 \leq d_2$ and let

$$D = m_2 T_2 + d_2 - (m_1 T_1 + d_1) \tag{135}$$

where $m_1, m_2 \geq 0$ and $m_1 \in M_1(m_2)$ where

$$M_1(m_2') = \{m_1 \mid D \geq 0 \text{ and } m_2 = m_2'\}. \tag{136}$$

Then

$$\min(D) = l \tag{137}$$

where $m_2 \geq 0$ and $m_1 \varepsilon M_1(m_2)$ where l is given by (132).

Proof of Lemma 4

If a is the $gcd(T_1, T_2)$ then b_1, b_2 are relative prime so there are integers v_1, v_2 such that

$$v_1 b_1 - v_2 b_2 = 1. \tag{138}$$

If $k > 0$ in (132) then multiplying both sides of (138) by $-ka$ yields

$$v_2 kab_2 - v_1 kab_1 = -ka$$

or

$$m_2'T_2 - m_1'T_1 = -ka$$

where $m_i' = v_i k,\ i = 1, 2$. Now consider the difference

$$\begin{aligned} D &= \left(m_2'T_2 - m_1'T_1\right) + \left(d_2 - d_1\right) \\ &= \left(-ka\right) + \left(ka + l\right) \qquad (139) \\ &= l. \end{aligned}$$

Because $l \geq 0$ (139) shows that $m_1' \varepsilon M_2\left(m_2'\right)$ so

$$min\left(D\right) \leq l \qquad (140)$$

when $m_2 > 0$ and $m_1 \varepsilon M_1\left(m_2'\right)$.

Now the question is whether or not there is an l' such that $D = l' < l$. Consider the difference

$$W = m_2T_2 - m_1T_1 \qquad (141)$$

where $m_2 \geq 0$ and $m_1 \varepsilon M_1\left(m_2\right)$. Dividing both sides of (141) by a yields

$$W / a = m_2b_2 - m_1b_1$$

which shows that $W = wa$ for some integer w. The definition of $M_1(m_2)$ and (135) show that $w \geq -k;$ otherwise, $D < 0$. So (135) and (141) show that

$$\begin{aligned} D &= wa + ka + l \\ &\geq -ka + ka + l \\ &= l. \end{aligned} \tag{142}$$

Inequalities (140) and (142) show that

$$min(D) = l \tag{143}$$

when $m_2 > 0$ and $m_1 \varepsilon M_1(m_2)$.

This completes the proof and shows that l is the minimum separation between the starts of any use periods of product 1 and 2 where the use period of product 1 begins before that of product 2. Because the labeling of the products was arbitrary (143) is valid regardless of the order of d_1 and d_2.

Conclusion

The results of lemma 4 shows that the minimum separation between any two production periods of products 1 and 2 is l. Furthermore, for any schedule $(\tilde{T}, \tilde{d})$ there exist m_1' and m_2' such that $D = l$. Hence, without loss of generality, in the two-products GCM time zero may be selected so that $d_1 = 0$ and $d_2 = l$. Thus, if (126) is not satisfied so that

$$a < q_1 + q_2$$

then either (130) or (131) is not satisfied. Therefore, the conditions of theorem 3 are necessary to assure the feasibility of schedule of two-products. This completes the proof of theorem 3.

CHAPTER 6

N-PRODUCTS MODEL

PURPOSE

In this chapter the necessary and sufficient conditions given by the general theorem 1 on schedule feasibility in chapter 3 are modified somewhat. Then lemma 5 provides a small, but useful, extension to these conditions. Next theorems 4 and 5 are proved. These theorems play important roles in the n-products optimization algorithm in chapter 9. These theorems are also used in chapters 7 and 8, where the results of chapter 5 are extended by developing delay-independent necessary and sufficient conditions for the three-products and four-products models, respectively.

The reader is cautioned that none of the variables in chapter 6 through 9 have exponents. However, in several cases superscripts are used to denote variables that are related to or are factors of similar variable. For example, k_{ij}^{2} is used to denote a factor of k_{ij}.

A straightforward modification of the necessary and sufficient feasibility conditions of theorem 1 in chapter 3 simplifies the presentations in chapters 6 through 9. That is, let

$$v_{ij} = l_{ij} - q_j \tag{144}$$

where l_{ij} is defined in (24) of chapter 3. With this transformation conditions (24) and (25) of chapter 3 become

$$d_i - d_j = k_{ij}a_{ij} + q_j + v_{ij} \tag{145}$$

where

$$0 \le v_{ij} + q_j < a_{ij}. \tag{146}$$

Then constraints (26) and (27) become

$$v_{ij} \ge 0 \tag{147}$$

and

$$v_{ij} \le a_{ij} - q_j - q_i \tag{148}$$

where $i,\ j = 1,\ 2,\ \cdots,\ n$ and $d_i > d_j$. This statement of the feasibility conditions will be used for the remainder of chapter 6 and in chapters 7 through 9.

The optimization algorithm given in chapter 9 is primarily an implicit enumeration over potentially optimal schedules $\Omega = \left(\tilde{T},\ \tilde{d},\ \prod\right)$. Generally the efficiency of such algorithms will be improved if some of the potentially optimal schedules can be eliminated with a priori reasoning. Theorem 4 makes it possible to eliminate a priori a large portion of the potentially feasible delay times $\tilde{d}$. Likewise, theorem 5 allows the a priori elimination of a significant portion of the potentially feasible sequences $\prod$. Before stating theorem 4 and 5, the feasibility conditions in chapter 3 are extended by lemma 5 and a definition is given for compact schedules. Then statements and proofs for theorem 4 and 5 are given.

LEMMA 5 ON EXTENDED FEASIBILITY CONDITIONS

The proof of the general theorem 1 on schedule feasibility assumes that the products have been labeled such that $d_i > d_j$ in (24) and (145). The feasibility conditions can be extended to remove the need for this assumption.

If $d_i > d_j$ then conditions (145), (146), (147), and (148) are valid if and only if there are k_{ij}^2 and v_{ij}^2 such that

$$d_j - d_i = k_{ij}^2 a_{ij} + q_i + v_{ij}^2 \tag{149}$$

where

$$0 \leq v_{ij}^2 + q_i < a_{ij} \tag{150}$$

and where v_{ij}^2 satisfies the constraints

$$v_{ij}^2 \geq 0 \tag{151}$$

and

$$v_{ij}^2 \leq a_{ij} - q_j - q_i. \tag{152}$$

PROOF OF LEMMA 5

The negative of (145) is

$$\begin{aligned} d_j - d_i &= k_{ij}a_{ij} - q_j - v_{ij} \\ &= \left(k_{ij} + 1\right)a_{ij} + q_i + \left(a_{ij} - q_j - q_i - v_{ij}\right) \\ &= k_{ij}^2 a_{ij} + q_i + v_{ij}^2 \end{aligned}$$

where

$$k_{ij}^2 = -\left(k_{ij} + 1\right)$$

and

$$v_{ij}^2 = a_{ij} - q_j - q_i - v_{ij}.$$

If (151) and (152) are satisfied then

$$v_{ij}^2 = a_{ij} - q_j - q_i - v_{ij} \geq 0$$

so that

$$v_{ij} \leq a_{ij} - q_j - q_i.$$

Also

$$v_{ij}^2 \leq a_{ij} - q_j - q_i$$

so that

$$v_{ij} \geq 0.$$

A similar argument shows that if v_{ij} satisfies (147) and (148) then v_{ij}^2 satisfies (151) and (152). Therefore, the feasibility conditions can be expressed by (145), (146), (147) and (148) regardless of the relative magnitudes of d_i and d_j. This extension does require that k_{ij} be permitted to take on negative values.

DEFINITION OF COMPACT SCHEDULE

A schedule $\left(\tilde{T}, \tilde{d}, \prod\right)$ of n-products is a compact schedule if

$$d_{\Pi 1} = 0 \tag{153}$$

$$d_{\Pi 2} = q_{\Pi 1} \tag{154}$$

and

$$d_{\Pi i} = d_{\Pi j} + k_{\Pi i}, \Pi j \bullet a_{\Pi i,\, \Pi j} + q_{\Pi j} \tag{155}$$

for $i = 3, 4, \cdots, n$ where $i > j$. This means that in a compact schedule the use period of product T_i, $\prod i > 2$, begins immediately after the completion of a use period of a product $\prod j$ where product $\prod j$ occurs in the sequence before product $\prod i$. Note that the sequence $\prod$ has a different meaning here than that of sequences of earlier chapters where it is assumed that $d_{\Pi i} > d_{\Pi j}$ if $i > j$. However, here the sequence $\prod$ is defined by (153), (154), and (155) so it is possible that $d_{\Pi j} > d_{\Pi i}$. The extended feasibility conditions of lemma 5 show that this definition of Π is sound.

A comparison of equations (145) and (155) gives an alternate definition of a compact schedule. That is, a schedule Ω is compact if (153) and (154) hold and if for each $i = 3, 4, \cdots, n$ there is at least one j such that $i > j$ and

$$v_{\Pi i,\, \Pi j} = 0. \tag{156}$$

Thus, either (155) or (156) can be used to determine if a given schedule is compact.

THEOREM 4 ON COMPACT SCHEDULES

A schedule Ω is feasible if and only if there exists a series of $(n - 1)$ linear transformations that map Ω to a feasible compact schedule Ω^n.

PROOF OF THEOREM 4

Let $\Omega = \left(\tilde{T}, \tilde{d}, \Pi\right)$ be a schedule of n-product. Without loss of generality it is assumed that the products are labeled such that $\Pi i = i$ and $d_i > d_j$ when $i > j$, $all\ i,\ j,\ i \neq j$.

First suppose that Ω is feasible. The proof must show that if (145) through (148) are satisfied then a new feasible Ω^n can be constructed that satisfies (153), (154), and (155).

To construct a Ω^n where (153) is valid note that because of the defining assumption i in chapter 1 time zero for Ω can be arbitrarily changed without effecting the feasibility of Ω.

In particular, one can arbitrarily select time zero to coincide with the beginning of any use period of any arbitrarily selected product $\boldsymbol{j}$. This means that one can arbitrarily select any product $\boldsymbol{j}$ and revise time zero so that $\boldsymbol{d}_j = \boldsymbol{0}$. The revised schedule $\boldsymbol{\Omega^o}$ is feasible if and only if $\boldsymbol{\Omega}$ is feasible. Then (153) is satisfied by setting $\Pi \boldsymbol{1} = \boldsymbol{j}$ and by redefining $\tilde{\boldsymbol{d}}$ and Π to agree with this new time zero. To find a product $\Pi \boldsymbol{2}$ that satisfies (154) find product $\boldsymbol{j}$ such that

$$\boldsymbol{v}_{j,\,\Pi 1} = \min_{k \neq \Pi 1} \boldsymbol{v}_{k,\Pi 1} = \boldsymbol{v}^2 \tag{157}$$

and let

$$\boldsymbol{d}_i^2 = \boldsymbol{d}_i - \boldsymbol{v}^2 \tag{158}$$

for $\boldsymbol{i} \neq \Pi \boldsymbol{1}$. In case of ties in (157) one of the products in the tie is arbitrarily selected as product $\boldsymbol{j}$. The linear transformation (158) is applied to all products except product $\Pi \boldsymbol{1}$. This yield a new $\left\{\boldsymbol{d}_i^2\right\}$ where $\boldsymbol{d}_{\Pi 1}^2 = \boldsymbol{d}_{\Pi 1} = \boldsymbol{0}$ and

$$\boldsymbol{d}_i^2 = \boldsymbol{d}_i - \boldsymbol{d}_{\Pi 1} = \boldsymbol{k}_{i,\Pi 1}\boldsymbol{a}_{i,\Pi 1} + \boldsymbol{q}_{\Pi 1} + \boldsymbol{v}_{i,\Pi 1}^2 \tag{159}$$

for $\Pi \boldsymbol{1}$. Observe that the differences $\left(\boldsymbol{d}_i - \boldsymbol{d}_j\right)$ where $\boldsymbol{j} \neq \boldsymbol{i}$ and $\boldsymbol{j} \neq \Pi \boldsymbol{1}$ are not effected by the linear transformation (158). Clearly, if $\left\{\boldsymbol{d}_i\right\}$ satisfies (145) through (148) then so does $\left\{\boldsymbol{d}_i^2\right\}$ Thus, if one sets $\Pi \boldsymbol{2} = \boldsymbol{j}$ then $\left\{\boldsymbol{d}_i^2\right\}$ becomes a feasible set of delay times where $\boldsymbol{d}_{\Pi 1}^2 = \boldsymbol{0}$ and

$$\boldsymbol{d}_{\Pi 2}^2 = \boldsymbol{k}_{\Pi 2,\Pi 1}\boldsymbol{a}_{\Pi 1,\Pi 2} + \boldsymbol{q}_{\Pi 1}. \tag{160}$$

By lemma 4 on minimum separation there are $\boldsymbol{m_1}$ and $\boldsymbol{m_2}$ such that

$$\boldsymbol{m_1 T_{\Pi 1} = m_2 T_{\Pi 2} + k_{\Pi 2,\Pi 1} a_{\Pi 1,\Pi 2}}. \tag{161}$$

Now reset time zero again to coincide with the start of the $\boldsymbol{m_1}$ use period of product $\boldsymbol{\Pi 1}$ and redefine $\boldsymbol{\tilde{d}}$ and $\prod$ to agree with this new time zero. This yields a new $\left\{\boldsymbol{d_i^{2b}}\right\}$ where $\boldsymbol{d_{\Pi 1}^{2b} = 0}$ and

$$\begin{aligned} \boldsymbol{d_{\Pi 2}^{2b}} &= \boldsymbol{m_2 T_{\Pi 2} + d_{\Pi 2}^{2} - m_1 T_{\Pi 1}} \\ &= \boldsymbol{q_{\Pi 1}}. \end{aligned} \tag{162}$$

To summarize these first three steps, for each schedule $\boldsymbol{\Omega}$ there exists a companion $\boldsymbol{\Omega^{2b}}$ where (153) and (154) are satisfied. This is true whether or not $\boldsymbol{\Omega}$ is feasible; however, $\boldsymbol{\Omega^{2b}}$ is feasible if $\boldsymbol{\Omega}$ is feasible.

To simplify the notation for the remainder of the proof, assume that $\left\{\boldsymbol{d_i^{2}}\right\}$ is revised to reflect the shift in time zero given by (162) and the products are relabeled so that

$$\boldsymbol{d_1^{2b} = 0}$$

$$\boldsymbol{d_2^{2b} = q_1}$$

and

$$\boldsymbol{d_i^{2b} > d_j^{2b}}$$

for $\boldsymbol{i = 3, \cdots, n;\ i > j.}$

The construction of a compact schedule is completed by performing $(n-2)$ additional transformations. To see this, let

$$\theta^{w} = \{ i \mid d_i \text{ is fixed during the } w\text{th transformation } \}$$

for $w = 3, \cdots, n$ where $\theta^{3} = \{\Pi 1, \Pi 2\} = \{1, 2\}$. That is, θ^{w} is the set of indices of those products, identified by previous transformations, with delay times d_i that satisfy (155). The wth transformation is not applied to the products i where $i \in \theta^{w}$. Thus, d_i, $i \in \theta^{w}$ is not changed by the wth transformation. Suppose that products k and j are the products such that

$$v^{w} = v_{kj}^{w-1} = min\ v_{im}^{w-1} \tag{163}$$

subject to $j, m \in \theta^{w}$ and $k, i \notin \theta^{w}$ for $w = 3, 4, \cdots, n$ where $d_i^{w-1} = d_i^{2b}$ when $w = 3$. Set

$$d_i^{w} = d_i^{w-1} - v^{w} \tag{164}$$

for $i \notin \theta^{w}$ and

$$d_i^{w} = d_i^{w-1}$$

for $i \in \theta^{w}$. Then

$$v_{im}^{w} = v_{im}^{w-1} - v^{w} \tag{165}$$

for $m \in \theta$ and $i \notin \theta$. After transformation (163) and (164) is performed

$$v_{kj}^{w} = 0$$

and

$$d_k^w = d_j^w + k_{ij}a_{ij} + q_j$$

Hence, (155) is satisfied by setting

$$\Pi w = k.$$

Clearly, $\left\{d_i^w\right\}$ is feasible, i.e. satisfies (145) through (148), if $\left\{d_i^{w-1}\right\}$ is feasible. Observe that the differences $\left(d_i - d_j\right)$ for $i,\ j \notin \theta^w$ are not effect by the wth transformation.

Hence, construction of a compact schedule can be completed by starting with $\left\{d_i^{2b}\right\}$ and $\theta^3 = \left(\Pi 1,\ \Pi 2\right)$ and performing transformations defined by (163) and (164). After each transformation, except the last, Πw is added to the set θ^{w-1} to create the set

$$\theta^w = \left\{\Pi 1,\ \Pi 2,\ \cdots,\ \Pi w\right\}$$

that is used to perform the $\left(w + 1\right)th$ transformation. By induction $\left\{d_i^n\right\}$ is feasible if $\left\{d_i\right\}$ is feasible. Hence, the compact schedule $\Omega^n = \left\{T_{\Pi i}, d_{\Pi i}^n, \Pi\right\}$ is feasible if Ω is feasible.

To complete the proof suppose that the compact schedule $\Omega^n = \left\{T_{\Pi i}, d_{\Pi i}^n, \Pi\right\}$ created by transforming Ω is not feasible; thus, one or more of the v_{ij}^n do not satisfy either (147) or (148). Suppose there is a $v_{ij}^n < 0$ so that (147) is violated. The transformation defined by the pairs of equations (158), (159) and (161), (162) and (163), (164) assure that if $v_{ij} \geq 0$ all $i,\ j;\ i \neq j$ then $v_{ij}^{n} \geq 0$. Thus, if there is a v_{ij}^{n} such that $v_{ij}^{n} < 0$ then $v_{ij} = 0$ so v_{ij} violates (147) and Ω is not feasible. By contradiction, if Ω is feasible then $v_{ij}^{n} \geq 0$, all i, j.

Finally, suppose there is a v_{km}^{n} that violates (148) so that

$$v_{km}^{n} > a_{ij} - q_{j} - q_{i}.$$

By construction, the series of transformations assure that

$$v_{ij} \geq v_{ij}^{2} \geq \cdots \geq v_{ij}^{n}$$

for all $i,\ j;\ i \neq j$. Thus, if any v_{km}^{n} does not satisfy (148) then neither does v_{ij}. Again by contradiction, if Ω is feasible then $v_{ij}^{n} \leq a_{ij} - q_{j} - q_{i}$, all i, j. Therefore, a schedule Ω is feasible if and only if the n transformations described above results in a feasible compact schedule Ω^{n}.

DEFINITION OF PARTIAL SCHEDULES AND THEIR ADDITION

Let θ_{n} be the set of n-products produced on a single facility and let τ_{k} be a subset of $k,\ 1 \leq k < n,$ of the products contained in θ_{n}. Further, let τ_{m}' be the complement of τ_{k} that is,

$$\tau_{m}' = \left\{ product\ i \;\middle|\; i \in \theta_{n}\ but\ i \notin \tau_{k} \right\} \qquad (166)$$

where $m = n - k$. Then a partial schedule of size k is the schedule

$$\Omega_k(\tau_k) = \left(\{T_i\}, \{d_i\} \mid i \in \tau_k\right). \tag{167}$$

That is, a partial schedule is a set of specified cycle times and delay times for the k products contained in τ_k. The sequence $\prod$ is not germane in this context so it is not used in definition (167).

Let Ψ_u be a subset of u-products contained in τ'_m then the addition of partial schedules is defined by

$$\begin{aligned}\Omega_{k+u}(\tau_k + \Psi_u) &= \Omega_k(\tau_k) + \Omega_u(\Psi_u) \\ &= \left(\{T_i\}, \{d_i\} \mid i \in \tau_k \ or\ i \in \Psi_u\right).\end{aligned} \tag{168}$$

This definition of partial schedule addition requires that if $i \in \tau_k$ then $i \notin \Psi_u$ and vice versa. One example of partial schedule addition is

$$\Omega = \Omega_n(\theta_n) = \Omega_k(\tau_k) + \Omega_{n-k}(\tau'_{n-k}).$$

THEOREM 5 ON PARTIAL SCHEDULE FEASIBILITY

Let θ_n be the set of n-products produced on a single facility and let τ_k be a specified subset of k, $1 \le k < n$, products contained in θ_n. Let τ'_m be the complement of τ_k where $m = n - k$. Further, let w be any one of the products contained in τ'_m and let τ_{k+1} be the set of $k + 1$ products that is created by adding w to τ_k. Let $\Omega_k(\tau_k)$ be a specified partial schedule of the $\boldsymbol{k}$-products in τ_k. Then there exists a partial schedule $\Omega_m(\tau'_m)$ of products in τ'_m such that the full schedule

$$\Omega = \Omega_k(\tau_k) + \Omega_m(\tau'_m) \tag{169}$$

is feasible only if $\Omega_k(\tau_k)$ is a feasible schedule of $\boldsymbol{k}$-products and there exists a single product schedule

$$\Omega_1(w) = (T_w,\ d_w) \tag{170}$$

such that

$$\Omega_{k+1}(\tau_{k+1}) = \Omega_k(\tau_k) + \Omega_1(w) \tag{171}$$

is a feasible schedule of $(k+1)$ products.

PROOF OF THEOREM 5

Suppose that $\Omega_k(\tau_k)$ is an infeasible schedule of the $\boldsymbol{k}$-product contain in τ_k. Then by theorem 1 on schedule feasibility there is at least one pair of products $i,\ j \in \tau_k$ such that either (147) or (148) is not satisfied. That is, either

$$v_{ij} < 0 \tag{172}$$

or

$$v_{ij} > a_{ij} - q_j - q_i \tag{173}$$

where

$$d_i - d_j = k_{ij}a_{ij} + q_j + v_{ij}. \tag{174}$$

Clearly, if either (172) or (173) are true in the partial schedule $\Omega_k(\tau_k)$ it will be true in any full schedule given by (169) that contains $\Omega_k(\tau_k)$ as a partial schedule. This means that any such full schedule will also be infeasible. Thus, the full schedule Ω given by (169) is feasible only if $\Omega_k(\tau_k)$ is feasible schedule of the k-products contained in τ_k.

Now consider the partial schedule of $k + 1$ products given by (171). The argument of the previous paragraph can be repeated to show that a necessary condition for a full schedule

$$\Omega^2 = \Omega_{k+1}(\tau_{k+1}) + \Omega_{m-1}(\tau'_{k+1})$$

to be feasible is that $\Omega_{k+1}(\tau_{k+1})$ must be a feasible schedule of the $k + 1$ products in τ_{k+1}. Any feasible full schedule containing $\Omega_k(\tau_k)$ as a partial schedule must also contain $\Omega_{k+1}(\tau_{k+1})$ as given by (171) for some single product schedule $\Omega_1(w)$. Therefore if $\Omega_{k+1}(\tau_{k+1})$ is infeasible for every possible choice of $\Omega_1(w) = (T_w,\ d_w)$ then all full schedules containing $\Omega_k(\tau_k)$ as a partial schedule are infeasible. This completes the proof of theorem 5.

LEMMA 6 ON TWO-PRODUCTS PARTIAL SCHEDULES

Let τ_k, $2 \leq k \leq n$, be a subset of n-products produced on a single facility and let $T = \left\{ t_i \mid i \in \tau_k \right\}$ be a specified set of cycle times for the products $i \in \tau_k$. Further, let

$$a_{ij} = gcd\left(t_i,\ t_j\right)$$

for $i,\ j \in \tau_k$ and $i \neq j$ and let Ω be any full schedule of the n-products where $T_i = t_i$ for $i \in \tau_k$. Then Ω is not feasible if there is any pair $i,\ j \in \tau_k$, $i \neq j$, such that

$$a_{ij} < q_i + q_j. \tag{175}$$

PROOF OF LEMMA 6

The proof of lemma 6 follows immediately from theorem 3 in chapter 5 and theorem 5. The optimization algorithm of chapter 9 searches for an optimal full schedule by generating a set of partial schedules. Lemma 6 provides a method for greatly improving the efficiency of this optimization algorithm. The test (175) when applied to all possible pairs ***i, j*** of products in a partial schedule will eliminate many partial schedules from further consideration at each stage of the schedule generation process.

LEMMA 7 ON SOLUTIONS TO AN INTEGER EQUATION

Suppose T_i, T_j is any given pair of integer cycle times and let

$$a_{ij} = gcd(T_i,\ T_j), \tag{176}$$

$$b_{ij} = T_i \,/\, a_{ij}, \tag{177}$$

and

$$b_{ji} = T_j \,/\, a_{ij}. \tag{178}$$

If an integer solution exists to the equation

$$k_{ij}T_i - k_{ji}T_j = w \tag{179}$$

for $-\infty < w < \infty$ where w is an integer then all such integer solutions are of the form

$$k_{ij} = \left(mb_{ji} + u\alpha_{ij}\right) \tag{180}$$

$$k_{ji} = \left(mb_{ij} + u\alpha_{ji}\right) \tag{181}$$

for $-\infty < m < \infty$ where

$$u = w \,/\, a_{ij} \tag{182}$$

is an integer and where α_{ij}, α_{ji} are integers such that

$$\alpha_{ij}b_{ij} - \alpha_{ji}b_{ji} = 1. \tag{183}$$

PROOF OF LEMMA 7

To show that $\boldsymbol{u}$ given by (182) is integer if $\boldsymbol{k_{ij}}$, $\boldsymbol{k_{ji}}$ are integer consider

$$\boldsymbol{k_{ij}T_i - k_{ji}T_j = k_{ij}b_{ij}a_{ij} - k_{ji}b_{ji}a_{ij} = w}$$

so that

$$\boldsymbol{k_{ij}b_{ij} - k_{ji}b_{ji} = w / a_{ij} = u.} \tag{184}$$

Given the definitions (177) and (178) the left hand side of (184) is integer of $\boldsymbol{k_{ij}}$ and $\boldsymbol{k_{ji}}$ are integers. Thus, $\boldsymbol{u}$ must be integer if $\boldsymbol{k_{ij}}$ and $\boldsymbol{k_{ji}}$ are. If $\boldsymbol{w \neq ua_{ij}}$ for some $\boldsymbol{u}$ then it is clear from (184) that no integer solution exists to (179).

By using the definition of the greatest common divisor of two integers one can easily show that $\boldsymbol{b_{ij}}$ and $\boldsymbol{b_{ji}}$, as defined by (177) and (178), are relative prime. A well known result of linear algebra states that if $\boldsymbol{b_{ij}}$ and $\boldsymbol{b_{ji}}$ are relative prime integers then there are integers $\boldsymbol{\alpha_{ij}}$, $\boldsymbol{\alpha_{ji}}$ such that

$$\boldsymbol{\alpha_{ij}b_{ij} - \alpha_{ji}b_{ji} = 1.} \tag{185}$$

Multiplying both sides of (185) by $\boldsymbol{ua_{ij}}$ yields

$$\boldsymbol{u\alpha_{ij}T_i - u\alpha_{ji}T_j = ua_{ij} = w.} \tag{186}$$

Adding

$$mb_{ij}b_{ji}a_{ij} - mb_{ij}b_{ji}a_{ij} = 0$$

to (186) yields

$$\left(mb_{ji} + u\alpha_{ij}\right)T_i - \left(mb_{ij} + u\alpha_{ji}\right)T_j = w$$

for $-\infty < m,\ w < \infty.$ Thus, there exists an infinite number of solutions to (179) of the form given by (180) and (181) for any integers $T_i,\ T_j, u$ for $-\infty < u < \infty.$

Suppose there exists a solution

$$y_{ij}T_i - y_{ji}T_j = ua_{ij} = w \tag{187}$$

to (179) that is not of the form given by (180) and (181). Subtracting (187) from (186) yields

$$\left(u\alpha_{ij} - y_{ij}\right)T_i - \left(u\alpha_{ji} - y_{ji}\right)T_j = 0$$

so

$$\left(y_{ij} - u\alpha_{ij}\right)b_{ij} = \left(y_{ji} - u\alpha_{ji}\right)b_{ji}. \tag{188}$$

Because b_{ij} and b_{ji} are relative prime b_{ji} must divide $\left(y_{ij} - u\alpha_{ij}\right)$ and b_{ij} must divide $\left(y_{ji} - u\alpha_{ji}\right)$ so there are $m_{ij},\ m_{ji}$ such that

$$m_{ij}b_{ij} = \left(y_{ji} - u\alpha_{ji}\right) \tag{189}$$

and

$$\boldsymbol{m_{ji}b_{ji} = y_{ij} - u\alpha_{ij}}. \tag{190}$$

By substituting the left hand side of (189) and (190) into (188) reveals that $\boldsymbol{m_{ij} = m_{ji} = m}$. Thus,

$$\boldsymbol{y_{ij} = mb_{ji} + u\alpha_{ij}}$$

and

$$\boldsymbol{y_{ji} = mb_{ij} + u\alpha_{ji}}$$

which are of the form given by (180) and (181). Hence, by contradiction, if an integer solution to (179) exists then all such solutions are of the form given by (180) and (181). Furthermore, an infinite number of integer solutions to (179) exist if and only if $\boldsymbol{w = ua_{ij}}$ where $\boldsymbol{u}$ is an integer such that $-\infty < \boldsymbol{u} < \infty$.

To consider some special cases suppose that $\boldsymbol{w = 0}$ then (179) yields

$$\boldsymbol{k_{ij}b_{ij} = k_{ji}b_{ji}}$$

so that

$$\boldsymbol{k_{ij} = mb_{ji} = mb_{ji} + u\alpha_{ij}}$$

and

$$\boldsymbol{k_{ji} = mb_{ij} = mb_{ij} + u\alpha_{ji}}.$$

Thus, if $w = 0$ then an infinite number of solutions to (179) exist that are of the form given by (180) and (181).

From (180) and (183) it follows that $k_{ij} = 0$ if and only if $u = u_2 b_{ji}$ and $m = -u_2 \alpha_{ij}$ for u_2 such that $-\infty < u_2 < \infty$. When $k_{ij} = 0$ (181) and (183) and the observations of the previous sentence yields

$$k_{ji} = -u_2 \alpha_{ij} b_{ij} + u_2 b_{ji} \alpha_{ji} = -u_2$$

so

$$k_{ij} T_i - k_{ji} T_j = -k_{ji} a_{ij} b_{ji} = u_2 a_{ij} b_{ji} = u a_{ij}$$

Of course, similar statements are true with respect to k_{ji}. Thus, lemma 7 is valid for the degenerate case of $k_{ij} = 0$.

Finally, if $b_{ij} = b_{ji}$ then $b_{ij} = b_{ji} = 1$. For this degenerate case $\alpha_{ij} = y$ and $\alpha_{ji} = y - 1$ for any y such that $-\infty < y < \infty$.

CHAPTER 7

THREE-PRODUCTS MODEL

PURPOSE

In this chapter the general theorem 1 on schedule feasibility in chapter 3 and the results of chapter 6 will be used to extend the results for the two-products model given in chapter 5 to the three-products model. That is, the primary purpose of this chapter is to develop delay-independent conditions that are necessary and sufficient to assure that a feasible schedule of three-products exists given a specified set of cycle times $\left(T_1,\ T_2,\ T_3\right)$.

In 1992 Glass [30] presented delay-independent conditions for the three-products GCM. These conditions were proposed as both necessary and sufficient to assure schedule feasibility for the three-products GCM. However, a counter-example showing a feasible schedule of three products that does not satisfy Glass' conditions is given in the next section. This counter-example provides interesting insight into the structure of the three-products model. Such counter-examples are easily constructed once the conditions of theorem 6 are known. Theorem 6 will be stated and proved in the two sections following the counter-example.

To complete chapter 7 the two-products partial schedule feasibility test given by lemma 6 in chapter 6 will be extended by developing similar tests for the three-products model. These tests are presented as lemma 8.

COUNTER-EXAMPLE TO GLASS' CONDITIONS

When Glass' [30] three-products feasibility conditions are translated to the notation used here they require that

$$a_{ij} \geq q_i + q_j \tag{191}$$

for $i,\ j = 1,\ 2,\ 3$ and $i \neq j$, and

$$\frac{1}{2} \bullet \left(a_{12} + a_{13} + a_{23} - a_{123}\right) \geq q_1 + q_2 + q_3 \tag{192}$$

where

$$a_{123} = gcd\left\{T_1,\ T_2,\ T_3\right\} \tag{193}$$

and, as before

$$a_{ij} = gcd\left\{T_i,\ T_j\right\}. \tag{194}$$

Consider the three-products schedule $T_1 = 28,\ T_2 = 20,$ and $T_3 = 35$ where the model parameters are selected so that $q_1 = 3,\ q_2 = 1,$ and $q_3 = 4.$ Let

$$\begin{aligned} d_1 &= 0 \\ d_2 &= q_1 = 3 \\ d_3 &= 3 \bullet a_{13} + q_1 = 24. \end{aligned} \tag{195}$$

Thus, the proposed schedule is

$$\Omega = \left(\tilde{T}, \tilde{d}\right) = \begin{pmatrix} 28, 0 \\ 20, 3 \\ 35, 24 \end{pmatrix} \tag{196}$$

The two-products ***gcd*** are $a_{12} = 4$, $a_{13} = 7$, and $a_{23} = 5$; thus, (191) is satisfied by Ω for all i, j. However, $g = 1$ so (192) is not satisfied by Ω.

The least common multiplier of T_1, T_2 and T_3 is 140. This means the schedule given by (196) is composed of an infinite series of identical segments that are each 140 time units in length. Clearly, conflicts exist in the schedule given by (106) if and only if there are conflicts during the first 140 time units. The first 140 time units of the schedule in (196) is depicted by the schedule table of Figure 20. The number in each row of the right most column of figure 20 labeled "conflict" is the difference of the start time of the production cycle of that row less the end of the production cycle of the previous row. One or more of these differences will be negative if there are conflicts in the schedule depicted by the table. As can be seen from the table, there are no conflicts in the schedule of (196). Therefore, Glass' condition (192) is not a necessary condition for schedule feasibility.

SCHEDULE TABLE OF COUNTER-EXAMPLE

ITEM	CYCLE	START	END	CONFLICT
1	1	0	3	
2	1	3	4	0
2	2	23	24	19
3	1	24	28	0
1	2	28	31	0
2	3	43	44	12
1	3	56	59	12
3	2	59	63	0
2	4	63	64	0
2	5	83	84	19
1	4	84	87	0
3	3	94	98	7
2	6	103	104	5
1	5	112	115	8
2	7	123	124	8
3	4	129	133	5
1	6	140	143	7

Figure 20

THREE-PRODUCTS DELAY INDEPENDENT CONDITIONS

This section presents theorem 6 that gives delay-independent necessary and sufficient feasibility conditions for the three-products case of the GCM. Theorem 6 supposes that the cycle times (T_1, T_2, T_3) have been specified. Then the theorem provides a means for testing whether or not a feasible schedule $\Omega = (\widetilde{T}, \widetilde{d}, \Pi)$ exists. The test is based solely on the GCM parameters and the set of specified cycle times. That is, the test does not require knowledge of the delay times $\widetilde{d}$ or the sequence Π.

Theorem 6 on Conditions for Three-Products GCM

Suppose three products are produced on a single facility and that (T_1, T_2, T_3) is a proposed set of cycle times for these three products. Let

$$a_{ij} = gcd(T_i, T_j)$$

for $i, j = 1, 2, 3$ and $i \neq j$ and let

$$a_{ijk} = gcd(T_1, T_2, T_3) \tag{197}$$

and apply the division algorithm to find u^k_{123} and w^k_{123} such that

$$q_k = u^k_{123} a_{123} + w^k_{123} \tag{198}$$

for $k = 1, 2, 3$ where $u^k_{123} \geq 0$ and $0 \leq w^k_{123} < a_{123}$. Then a feasible schedule exist if and only if

$$a_{ij} \geq q_i + q_j, \tag{199}$$

for $i,\ j = 1,\ 2,\ 3$ and $i \neq j$ and there is at least one pair of products $(i,\ j)$ such that

$$a_{ij} \geq min\left\{\left[q_i + q_j + \left(a_{123} - w^i_{123}\right)\right],\ \left[q_i + q_j + w^k_{123}\right]\right\} \tag{200}$$

where $i,\ j,\ k = 1,\ 2,\ or\ 3$ and $i \neq j,\ i \neq k$ and $k \neq j.$ Furthermore, if (199) and (200) are satisfied by $(T_1,\ T_2,\ T_3)$ then a feasible set of delay times $\{d_i\}$ and a feasible Π can be constructed by the method used in the proof.

Proof of Theorem 6

To make a clear presentation of the proof of theorem 6 the proof will be organized into ten major parts. Some of these parts have two sub-parts.

Part 1– Compact Schedules

The results of theorem 4 on compact schedules in chapter 6 allow the search for a feasible schedule to be restricted to a search of compact schedule. Thus, the proof of theorem 6 is restricted to proving that (199) and (200) are necessary and sufficient to assure the feasibility of a compact schedule of the three products. Let Π be any one of the six possible sequences of the three products. Then by theorem 4 it can be assumed without loss of generality that (153), (154), and (155), used to define a compact schedule in chapter 6, are satisfied. To simplify the notation assume the products are labeled so that $\Pi_i = i.$ With this labeling (153) and (154) become

$$d_1 = 0 \tag{201}$$

and

$$d_2 = q_i. \tag{202}$$

Part 2 – Feasibility of Products 1 and 2

Let

$$\Omega_2 = \left[\left(T_1,\ T_2\right),\ \left(d_1,\ d_2\right),\ \Pi\right] \tag{203}$$

where T_1, T_2 are two of the cycle times specified by the hypothesis of theorem 6 and d_1, d_2 are delay times given by (201) and (202). According to theorem 3 on conditions for the two-products GCM given in chapter 5 the partial schedule Ω_2 is a feasible schedule of products 1 and 2 if and only if a_{12} satisfies (199). Furthermore, according to theorem 5 on partial schedule feasibility given in chapter 6, for any feasible three-products schedule containing Ω_2 to exist it is necessary that Ω_2 be a feasible partial schedule of products 1 and 2. Thus, for the balance of the proof, it can be assumed that $a_{12} \geq q_1 + q_2$; otherwise, a feasible full schedule of the three-products does not exist. This means that the remainder of the proof only need to be concerned with possible conflicts between the schedules of products 1 and 3 or the schedules of products 2 and 3.

Part 3 – Delay Time Dependent Feasibility Conditions

When applied to product 3 the revised delay time dependant feasibility condition (145) given in chapter 6 yields

$$d_3 = d_1 + k_{31}a_{13} + q_1 + v_{31} \tag{204}$$

and

$$d_3 = d_2 + k_{32}a_{23} + q_2 + v_{32}. \tag{205}$$

The feasibility constraints require that v_{3i} satisfy

$$0 \le v_{3i} \le a_{i3} - q_i - q_3 \tag{206}$$

for $i = 1, 2$. By substituting (201) and (202) into (204) and (205) and equating the right hand sides of the results one obtains the three products feasibility equation

$$k_{31}a_{13} - k_{32}a_{23} = q_2 - v_{31} + v_{32}. \tag{207}$$

For a feasible three product schedule to exist there must be k_{31}, k_{32}, v_{31}, and v_{32} that satisfy (207) where v_{31} and v_{32} satisfy (206).

Part 4 – Decomposition of Feasibility Equation

It follows from lemma 7 on solutions to an integer equation that solution to the left hand side of (207) are given by

$$k_{31}a_{13} - k_{32}a_{23} = \phi a_{123} \tag{208}$$

for some integer ϕ, $-\infty < \phi < \infty$. This result allows a decomposition of feasibility equation (207) into two parts. It will be shown that these two parts can be independently solved.

Part 5 – Solving for k_{31} and k_{32}

Let

$$b_{ijf} = {}^{a_{ij}}\!/\!_{a_{123}};$$

that is, b_{ijf} are those factors that are in a_{ij} but not in the other pairwise greatest common divisors. Now let ϕ be any integer, then lemma 7 asserts there are α_{31}, α_{32} such that

$$k_{31} = \left(mb_{231} + \phi\alpha_{31}\right) \tag{209}$$

$$k_{32} = \left(mb_{132} + \phi\alpha_{32}\right) \tag{210}$$

solves (208) for any m, $-\infty < m < \infty$. Therefore, ϕ can be set to any value dictated by the right hand side of (207). Once the desired value of ϕ has be determined by part 6 then equations (209) and (210) are used to determine k_{31} and k_{32}.

Part 6 – Solving for v_{31} and v_{32}.

Substituting (198) into the right hand side of the dissected (207) yields

$$\left(u_{123}^{2} - \phi\right)a_{123} + w_{123}^{2} - v_{31} + v_{32} = 0. \tag{211}$$

Because it is possible to restrict the proof to considering only compact schedules theorem 4 given in chapter 6 can be used to assert that either $v_{31} = 0$ or $v_{32} = 0$. Thus, there are two options to be considered.

Part 6A – $v_{32} = 0$

The constraint (206) for v_{32} is clearly satisfied if $v_{32} = 0$ and $a_{23} \geq q_2 + q_3$. Thus, (199) is necessary and sufficient to assure that there are no conflicts between the schedules of products 2 and 3 when $v_{32} = 0$.

Now to examine the feasibility of the schedules for products 1 and 3 solve (211) for v_{31} to find

$$v_{31} = \left(u_{123}^{2} - \phi\right)a_{123} + w_{123}^{2}. \tag{212}$$

Because $0 \leq w_{123}^{2} < a_{123}$ and $v_{31} \geq 0$, it is clear that $\phi \leq u_{123}^{2}$. Thus, from (212) it is clear that constraint (206) on v_{31} can be satisfied if and only if

$$a_{13} \geq q_1 + q_3 + w_{123}^{2} \tag{213}$$

where

$$v'_{31} = \min_{\phi} \; v_{31} = w^2_{123} \tag{214}$$

subject to $v_{31} \geq 0$. The minimum v_{31} occurs at $\phi = u^2_{123}$. Hence, when $v_{32} = 0$ the proper k_{31} and k_{32} can be obtained from (209) and (210) by setting $\phi = u^2_{123}$. Therefore, condition (213) is necessary to prevent conflicts between schedules of products 1 and 3 when $v_{32} = 0$.

<u>***Part 6B –***</u> $v_{31} = 0$

By repeating the argument of the first paragraph of part 6A one finds that (199) is necessary and sufficient to assure that there are no conflicts between the schedules of products 1 and 3 when $v_{31} = 0$. To examine the feasibility of the schedules for products 2 and 3 solve (211) for v_{32} to find

$$v_{32} = \left(\phi - u^2_{123} - 1\right)a_{123} + \left(a_{123} - w^2_{123}\right). \tag{215}$$

Because $0 \leq w^2_{123} < a_{123}$ and $v_{32} \geq 0$, it is clear from (215) that $\phi \geq \left(u^2_{123} + 1\right)$. Thus, when $v_{31} = 0$ constraint (206) on v_{32} can be satisfied if and only if

$$a_{23} \geq q_2 + q_3 + \left(a_{123} - w^2_{123}\right) \tag{216}$$

where

$$v'_{32} = \min_{\phi} \; v_{32} = \left(a_{123} - w^2_{123}\right) \tag{217}$$

subject to $v_{32} \geq 0$. The minimum v_{32} occurs at $\phi = u_{123}^{2} + 1$. Therefore, condition (216) is necessary to prevent schedule conflicts between schedules of products 2 and 3 when $v_{31} = 0$.

Part 7 – Sequence Independent Feasibility Condition

As stated, the delay independent feasibility conditions (213) and (216) are dependent on the scheduling sequence Π. The sequence independent feasibility condition (200) is quickly derived from (213) and (216) by observing that there must be at least one of six possible sequence Π under which either (213) or (216) is satisfied. Otherwise, there will be schedule conflicts for every possible sequence Π. Therefore, (199) and (200) are sequence and delay time independent feasibility necessary conditions for the three-products model of the GCM.

Part 8 – Sufficiency of the Feasibility Conditions

Now assume that (199) and (200) are satisfied for some pair $(i,\ j)$. The details of the proof that these conditions are necessary given in parts 1 through 7 will be used to construct a schedule that is feasible. This constructed feasible schedule proves that (199) and (200) are also sufficient conditions for schedule feasibility.

Part 8A – Feasible Sequence

The first step of the schedule construction is to determine a feasible sequence Π. Suppose (199) is satisfied and that a_{ij} satisfies (200). There are two possible values of (200) obtained from (213) and (216). First, suppose that (216) is satisfied. Set $\Pi 2 = k$ and arbitrarily set $\Pi 1 = j \neq k$ and $\Pi 3 = i,\ i \neq k$ and $i \neq j$ as given in condition (200). To simplify the notation assume that the products are labeled such that $\Pi i = i$. Set d_1 and d_2 to the values given by (201) and (202). According to theorem 3 on conditions for the two-products GCM condition (199) is sufficient to assure that the schedules of products 1 and 2 are feasible.

Part 8B – Feasible Delay Times

To construct a feasible d_3 set $v_{31} = 0$ and $v_{32} = a_{123} - w_{123}^2$ and let

$$\begin{aligned} d_3 &= d_1 + k_{31}a_{13} + q_1 + v_{31} \\ d_3 - d_1 &= k_{31}a_{13} + q_1 \end{aligned} \tag{218}$$

and

$$\begin{aligned} d_3 &= d_2 + k_{32}a_{23} + q_2 + v_{32} \\ d_3 - d_2 &= k_{32}a_{23} + q_2 + \left(a_{123} - w_{123}^2\right) \end{aligned} \tag{219}$$

where k_{31} and k_{32} are given by (209) and (210) of part 5 and where α_{31} and α_{32} are a pair of integers that satisfy

$$\alpha_{31}a_{13} - \alpha_{32}a_{23} = a_{123}. \tag{220}$$

A method for solving (220) for α_{31} and α_{32} is a well known result of linear algebra. This method is based on Euclid's algorithm for finding the greatest common divisor of the two integers. The proof in parts 3 through 6 assure that if the multiplier ϕ in (209) and (210) is set to $\left(u_{123}^2 + 1\right)$ then the two values for d_3 given by (218) and (219) are equal when $v_{31} = 0$ and $v_{32} = a_{123} - w_{123}^2$. Therefore, if (199) and (216) are satisfied then the constructed schedule $\Omega = \left[\left(T_1, T_2, T_3 \right), \left(d_1, d_2, d_3 \right), \Pi \right]$ satisfies the feasibility condition (149) through (152) in chapter 6.

Now suppose that (213) is satisfied. Set $\Pi 2 = k$ where k is the product index in (200). Then arbitrarily set $\Pi 1 = j \neq k$ and $\Pi 3, i \neq k, i \neq j$. By repeating the above argument it is easily shown that by setting the multiplier $\phi = u_{123}^2$ the feasibility equation (207) is valid when $v_{31} = w_{123}^2$ and $v_{32} = 0$. Clearly, if (213) is satisfied, the feasibility conditions (149) through (152) are also satisfied.

The above argument shows that the constructed schedule Ω is feasible if (199) and (200) are valid. Therefore, (199) and (200) are both necessary and sufficient to assure the existence of a feasible schedule of three products that contains a set of three specified cycle times. Furthermore, conditions (199) and (200) depend only on the cycle times and do not require knowledge of either $\{d_i\}$ or the sequence Π.

<u>Part 9 – Well Defined</u> d_3

The constructed schedule Ω posed in part 8 must satisfy another condition to assure that d_3 is well defined. To agree with the definition of d_i given in chapter 1 the delay time d_3 must satisfy

$$d_3 < T_3 - q_3. \tag{221}$$

Equations (209) and (210) reveal that there are an infinite number of pairs $\left(k_{31},\ k_{32}\right)$ that satisfy (218) and (219). Clearly, an m in (209) and (210) can be selected such that either

$$0 \le k_{31} \le b_{231} \tag{222}$$

or

$$0 \le k_{32} \le b_{132} \tag{223}$$

where there is equality in (222) only if $mb_{231} = \phi\alpha_{31}$ and in (223) only if $mb_{132} = \phi\alpha_{32}$.

Clearly an m can be selected such that (222) is valid. Then from (204)

$$\begin{aligned} d_3 &= d_1 + k_{31}a_{13} + q_1 + v_{31} \\ &= k_{31}a_{13} + q_1 + v_{31} \\ &< b_{231}a_{13} + q_1 + w_{123}^2. \end{aligned}$$

From conditions (213) $q_1 + w_{123}^2 \le a_{13} - q_3$ so

$$d_3 < b_{231}a_{13} + a_{13} - q_3 \le b_{231}a_{13} - q_3 \tag{224}$$

Now b_{231} and a_{13} are both factors of $T_3 = \theta b_{231}a_{13}$ for some integer θ ; so that,

$$d_3 \le T_3 - q_3. \tag{225}$$

Thus, d_3 satisfies the definition of d_i if m can be set to satisfy (222).

A similar argument shows that (225) can be satisfied by the proper choice of $\boldsymbol{m}$ when $\boldsymbol{k_{32}}$ satisfies (223). Therefore, a multiplier $\boldsymbol{m}$ in (209) and (210) can be selected that assures $\boldsymbol{d_3}$ satisfies the definition of a delay time.

Part 10 – Conclusion

In summary, theorem 6 provides necessary and sufficient conditions that a feasible schedule of three products exists when the cycle times are equal to some stipulated values. These conditions are quite general in the sense that they only require knowledge of three cycle times and the model parameters. The theorem provides a useful test of the feasibility of a set of cycle times because it can be conducted without having to first determine a sequence and set of delay times. The theorem provides a method for constructing a feasible schedule once it has been determined that the set of cycle times is feasible.

One can make some interesting observations about the hypothesis of theorem 6. For instance, if (199) is taken as the base requirement to assure a feasible schedule of two products then (200) gives the "penalty" that must be paid to schedule three products. At least one pair of the cycle times must pay this penalty by having a greatest common divisor that is greater than the base requirement. The amount of this penalty is $\boldsymbol{min}\left[\left(\boldsymbol{a_{123}} - \boldsymbol{w^i_{123}}\right), \boldsymbol{w^k_{123}}\right]$.

There are two special cases worth noting. If $\boldsymbol{a_{123}} > \boldsymbol{max}\ \boldsymbol{q_i}$ then $\boldsymbol{w^i_{123}} = \boldsymbol{q_i}$ for $\boldsymbol{i} = \boldsymbol{1, 2, 3}$. In this case the penalty is $\boldsymbol{min}\left[\left(\boldsymbol{a_{123}} - \boldsymbol{q_i}\right), \boldsymbol{q_k}\right]$. On the other hand, if $\boldsymbol{a_{123}} = \boldsymbol{q_i}$ then $\boldsymbol{w^i_{123}} = \boldsymbol{0}$ for some $\boldsymbol{i}$. Also, if $\boldsymbol{a_{123}} = \boldsymbol{1}$ then $\boldsymbol{w^i_{123}} = \boldsymbol{0}$ for all $\boldsymbol{i}$. In either case there is no penalty for scheduling three products. Thus, in these cases, the base requirement (199) is all that is needed to test the feasibility of the specified cycle times.

LEMMA 8 ON THREE-PRODUCTS PARTIAL SCHEDULES

Let τ_k, $3 \leq k \leq n$, be a subset of n products produced on a single facility and let $\Psi_k = \left\{ t_i \middle| i \varepsilon \tau_k \right\}$ be a specified set of cycle times for the products $i \varepsilon \tau_k$. Let Ω be any full schedule of the n products where $T_i = t_i$ for $i \varepsilon \tau_k$. Then Ω is not feasible if there is at least one group of three cycle times for products $i \varepsilon \tau_k$ that does not satisfy the hypothesis of theorem 6.

PROOF OF LEMMA 8

This lemma is an extension to lemma 6 on two-products partial schedule feasibility given in chapter 6. The proof of lemma 8 follows immediately from theorem 5 and theorem 6. Lemma 8 provides a useful "filter" that can be used by the optimization algorithm of chapter 9 to eliminate many partial schedules at each stage of the algorithm after the second. However, the "filtering" capability of theorem 6 can be enhanced beyond the straightforward test of lemma 8.

REMARKS ABOUT LEMMA 8

The hypothesis of theorem 6 only requires that (199) and (200) be true for some sequence Π and some $\{d_i\}$. That is, it is supposed that one is free to construct a feasible

Π and $\{d_i\}$ when $\{T_i\}$ satisfies (199) and (200). However, the optimization algorithm generates a series of partial schedules. Each of these partial schedules specifies the cycle times, delay times, and sequence for a subset of the n products. New partial schedules are generated by adding a new product and its schedule to an existing partial schedule Ω. Then, the question is when is the proposed new single-product schedule compatible with the existing partial schedule Ω ? The question is answered by lemma 9.

LEMMA 9 ON ENHANCED THREE-PRODUCTS CONDITIONS

Let τ_k, $3 \le k \le n$, be a subset of n products produced on a single facility and let $\Psi_k = \{t_i | i \varepsilon \tau_k\}$ be a specified set of cycle times for the products $i \varepsilon \tau_k$. Let $\Omega_k = (\{t_i\}, \{d_i\}, \Pi)$ be a feasible partial schedule of the products $i \varepsilon \tau_k$. Suppose product m is added to τ_k to create τ_{k+1}. Let Ω_m be a proposed schedule of product m where $T_m = t_m$. Further, let

$$\Omega_{k+1} = \Omega_k + \Omega_m$$

be the partial schedule for the products $i \varepsilon \tau_{k+1}$ where $T_i = t_i$ for $i \varepsilon \tau_k$ and $T_m = t_m$. Let i, j be any pair of products i, $j \varepsilon \tau_k$ and suppose that $d_j > d_i$. Then let

$$q_i + v_{ij} = u^i_{ijm} a_{ijm} + w^i_{ijm} \tag{226}$$

and

$$q_j + v_{ji} = u^j_{ijm} a_{ijm} + w^j_{ijm} \tag{227}$$

where

$$d_j - d_i = k_{ji}a_{ij} + q_i + v_{ji} \tag{228}$$

and

$$d_i' - d_j' = T_j - \left(d_j - d_i\right) = k_{ij}a_{ij} + q_j + v_{ij} \tag{229}$$

and where $a_{ijm} = gcd\left(T_i,\ T_j,\ T_m\right),\ 0 \le w_{ijm}^i,\ w_{ijm}^j < a_{ijm},\ 0 \le v_{ij},\ v_{ji} < a_{ij}$ and $u_{ijm}^i, u_{ijm}^j, k_{ij}, k_{ji} \ge 0$. Then a necessary condition that a feasible full schedule Ω containing Ω_{k+1} exists is that for all pairs $i,\ j \varepsilon \tau_k$ at least one of the following four constraints is satisfied:

$$a_{im} \ge q_i + q_m + w_{ijm}^j \tag{230}$$

$$a_{jm} \ge q_j + q_m + \left(a_{ijm} - w_{ijm}^j\right) \tag{231}$$

$$a_{jm} \ge q_j + q_m + w_{ijm}^i \tag{232}$$

$$a_{im} \ge q_i + q_m + \left(a_{ijm} - w_{ijm}^i\right). \tag{233}$$

PROOF OF LEMMA 9

The proof of lemma 9 is the same as the theorem 6 except d_i and d_j do not necessarily satisfy (201) and (202); thus, $k_{ij},\ k_{ji},\ v_{ij}$, and v_{ji} may be strictly positive. Without loss of generality assume that $d_j > d_i$. This means product i is like product 1 in theorem 6.

Two modifications to the proof of theorem 6 make it a valid proof of lemma 9. To see this, rewrite the feasibility conditions (204) and (205) to obtain

$$d_m - d_i = k_{mi}a_{im} + q_i + v_{mi} \tag{234}$$

and

$$d_m - d_i = k_{ji}a_{ij} + q_i + v_{ji} + k_{mj}a_{jm} + q_j + v_{mj} \tag{235}$$

where

$$d_j - d_i = k_{ji}a_{ij} + q_i + v_{ji}. \tag{236}$$

Equating the right hand sides of (234) and (235) yields the feasibility equation

$$k_{mi}a_{im} - k_{mj}a_{jm} - k_{ij}a_{ij} = q_j + v_{ji} - v_{mi} + v_{mj} \tag{237}$$

where k_{ij} and v_{ji} are specified values. By observing that $a_{ij} = b_{ijm}a_{ijm}$ one can divide (237) into two parts where the left hand side is given by

$$k_{mi}a_{im} - k_{mj}a_{jm} = \phi a_{ijm} \tag{238}$$

where $\phi = \left(\phi' + k_{ij}b_{ijm}\right)$. This modification allows (238) to have the same form as (208) in part 4 of the proof of theorem 6.

To examine the right hand side of the feasibility equation (237) express $\left(q_j + v_{ji}\right)$ as shown by (227) to yield

$$\phi a_{ijm} = u_{ijm}^{j} a_{ijm} + w_{ijm}^{i} - v_{mi} + v_{mj}. \tag{239}$$

This second modification allows (238) to have the same form as (211) in part 6 of the proof of theorem 6. With these two modifications one can repeat the proof of theorem 6 to assert that if $d_j > d_i$ then (230) and (231) are necessary and sufficient to assure that a feasible schedule exist for products ***i, j, m***. This assertation assumes that the values of T_i, T_j, T_m and d_i, d_j are set to previously specified values.

Now to examine the case when $d_i' > d_j'$ obtain a new sequence and new delay times d_i', d_j'. Do this by shifting time zero to the start of the use period of product ***j*** that is the last such period before the first use period of product ***i***. That is, subtract T_j from d_j and shift time zero to the resultant time. After this shift of time zero $d_i' > d_j'$ and $d_i' - d_j'$ is given by (228). This means that product ***j*** is like product ***l*** in theorem 6. With this modification the proof of lemma 9 when $d_i' > d_j'$ is the same as the proof given above for $d_j > d_i$. Thus, if $d_i' > d_j'$ then (232) and (233) are necessary and sufficient to assure that a feasible schedule exists for products ***i, j, m***. As before, this assertation assumes that the values of T_i, T_j, T_m and d_i', d_j' are set to previously specified values.

By theorem 5 on partial schedule feasibility, the partial schedule of products ***i, j, m*** for all possible pairs $i, j \varepsilon \tau_k$ must be feasible for Ω_{k+1} to be feasible. Furthermore, Ω_{k+1} must be feasible for there to exist a feasible full schedule of n products that contains Ω_{k+1}. Therefore, for a feasible full schedule containing Ω_{k+1} to exist, at least one of the constraints (230) through (233) must be satisfied by all pairs of products $\boldsymbol{i, j} \varepsilon \tau_k$ and product $\boldsymbol{m}$.

REMARKS ABOUT LEMMA 9

Lemma 9 provides feasibility tests that are more robust than those of lemma 6 and lemma 8. Furthermore, these tests can be applied to, and are affected by, the partial schedules generated by the optimization algorithm of chapter 9.

CHAPTER 8

FOUR-PRODUCTS MODEL

PURPOSE

In this chapter the results of chapters 5 and 7 will be extended to the four-products model. That is, the primary purpose of this chapter is to develop delay-independent feasibility conditions for the four-products GCM given a specified vector of cycle times (T_1, T_2, T_3, T_4) There are two sets of conditions. A necessary set must be satisfied by any feasible schedule of the four products. On the other hand, at least one feasible schedule exists if the sufficient set of conditions are satisfied.

Some required notation is explained in the next section. Then the four-products delay-independent feasibility conditions will be stated as theorem 7. After a proof for theorem 7 is given, lemmas 10 and 11 will extend the results of lemmas 8 and 9 to apply the results of theorem 7 to partial schedules.

Although the number of products only increases by one, the effort required to develop and prove delay-independent conditions for the four-products model was at least two orders of magnitude greater than that required for the three-products model. The primary cause of this increased difficulty is that three integer feasibility equations are required by the four-models model as opposed to the one feasibility equation of the three-products model. Furthermore, these integer equations are highly intertwined. The computational effort required to apply the four-products conditions to a given $\{T_i\}$ is also somewhat greater than that required by the three-products conditions. Nevertheless, the nature of these computations is straightforward so these delay-independent conditions are a reasonable test of the feasibility of a specified set of four cycle times.

NOTATION AND DEFINITIONS

As defined in previous chapters, e.g. (194) of chapter 7, let

$$a_{ij} = gcd(T_i,\ T_j).$$

The definition (193) of chapter 7 is generalized by letting

$$a_{ijk} = gcd(T_i, T_j, T_k) \tag{240}$$

for any three products where $i \neq j,\ j \neq k$ and $i \neq k.$ To simplify the notation somewhat let

$$g = a_{1234} = gcd(T_1,\ T_2,\ T_3,\ T_4). \tag{241}$$

The statement and proof of theorem 7 requires the following factorization of the cycle times:

$$\beta_{ijk} = gcd\left(T_i/g,\ T_j/g,\ T_k/g\right) \tag{242}$$

$$\beta_{ij} = gcd\left[T_i/(\beta_{ijk}\beta_{ijm}g),\ T_j/\beta_{ijk}\beta_{ijm}g\right], \tag{243}$$

where $i, j, k, m = 1, 2, 3, 4$ and no two indices are equal. Finally, let β_i, $i = 1, 2, 3, 4$ be the factors such that

$$T_1 = \beta_1\beta_{12}\beta_{13}\beta_{14}\beta_{123}\beta_{124}\beta_{134}g \tag{244}$$

$$T_2 = \beta_2\beta_{12}\beta_{23}\beta_{24}\beta_{123}\beta_{124}\beta_{234}g \tag{245}$$

$$T_3 = \beta_3\beta_{13}\beta_{23}\beta_{34}\beta_{123}\beta_{134}\beta_{234}g \tag{246}$$

and

$$T_4 = \beta_4\beta_{14}\beta_{24}\beta_{34}\beta_{124}\beta_{134}\beta_{234}g. \tag{247}$$

These factors play an important role in the statement and proof of theorem 7. Note that

$$\beta_i,\ \beta_{ij},\ \beta_{ijk},\ g \geq 1, \tag{248}$$

$$\beta_{ij} = \beta_{ji} \tag{249}$$

and

$$\beta_{ijk} = \beta_{ikj} = \beta_{jik} = \beta_{jki} = \beta_{kij} = \beta_{kji} \tag{250}$$

for all i, j, k where no two indices are equal.

From their definition it is clear that all pairs of these factors are relative prime except the following:

1. β_i is not necessarily relative prime to β_{ij}, β_{ijk}, and g,

2. β_{ij} is not necessarily relative prime to β_{ijk} and g and

3. β_{ijk} is not necessarily relative prime to g

for all i, j, k where no two indices are equal. A useful interpretation that aids in understanding these factors is that

1. β_i are those factors in T_i that are not factors of any other T_j where $i \neq j$,

2. β_{ij} are those factors in T_i and T_j that are not factors of any other T_k where $i \neq k$ and $j \neq k$,

3. β_{ijk} are those factors in T_i, T_j, T_k that are not factors of T_m where $m \neq i, m \neq j$, and $m \neq k$, and

4. g is those factors common to all T_i.

Several greatest common divisors will appear in theorem 7. The proof will be easier to understand if the divisors are expressed as products of the factors defined by (241) through (246). An examination of (244), (245), and (246) reveals that

$$a_{12} = \beta_{12}\beta_{123}\beta_{124}g \tag{251}$$

$$a_{13} = \beta_{13}\beta_{123}\beta_{134}g \tag{252}$$

$$a_{14} = \beta_{14}\beta_{124}\beta_{134}g \tag{253}$$

$$a_{23} = \beta_{23}\beta_{123}\beta_{234}g \tag{254}$$

$$a_{24} = \beta_{24}\beta_{124}\beta_{234}g \tag{255}$$

$$a_{34} = \beta_{34}\beta_{134}\beta_{234}g \tag{256}$$

and

$$a_{ijk} = \beta_{ijk}g \tag{257}$$

for all i, j, k where $i \neq j,\ i \neq k,\ j \neq k.$ These factorizations of the greater common divisors will be used repeatedly in the proof of theorem 7.

Suppose that

$$\Omega^3 = \left[\left(T_1,\ T_2,\ T_3\right),\ \left(d_1,\ d_2,\ d_3\right),\ \Pi\right] \tag{258}$$

is a compact schedule of three products where, to simplify the notation, it is assumed the products are labeled so that $\Pi i = i.$ Then the delay times $\left(d_1,\ d_2,\ d_3\right)$ satisfy (201), (202), (204) and (205) of chapter 7.

Apply the division algorithm to express

$$q_2 = u_{124}^2 a_{124} + w_{124}^2 \tag{259}$$

$$q_3 + v_{31} = u^3_{134} a_{134} + w^3_{134} \tag{260}$$

$$q_3 + v_{32} = u^3_{234} a_{234} + w^3_{234} \tag{261}$$

where u^2_{124}, u^3_{134}, $u^3_{234} \geq 0$ and $0 \leq w^m_{ijk} < a_{ijk}$. Also, apply the division algorithm to find

$$u_B \beta_{234} a_{124} + w_B = q_2 \tag{262}$$

and

$$u^2_B \beta_{234} a_{134} + w^2_B = d_3 - q_1 + q_3 \tag{263}$$

let

$$\Delta_{42} = \begin{cases} w^2_B - w_B, & w^2_B \geq w_B \\ \beta_{234} a_{124} - \left(w_B - w^2_B\right), & w^2_B < w_B \end{cases} \tag{264}$$

and

$$\Delta_{43} = \begin{cases} w_B - w^2_B, & w_B \geq w^2_B \\ \beta_{234} a_{134} - \left(w^2_B - w_B\right), & w_B < w^2_B \end{cases} \tag{265}$$

where u_B, $u^2_B \geq 0$, $0 \leq w_B < \beta_{234} a_{124}$ and $0 \leq w^2_B < \beta_{234} a_{134}$.

FOUR-PRODUCTS DELAY INDEPENDENT CONDITIONS

This section first states theorem 7 that gives delay-independent feasibility conditions for the four-products case of the GCM. That is, theorem 7 supposes that the cycle times $\left(T_1,\ T_2,\ T_3,\ T_4\right)$ have been specified. The theorem then provides conditions that are necessary for a feasible schedule of the four products to exist. The theorem also provides conditions that are sufficient to assure that a feasible schedule exist. The tests are based solely on the GCM parameters and the vector of specified cycle times. Thus, these tests are said to be delay-time independent.

Theorem 7 on Conditions for Four-Products GCM

Suppose four products are produced on a single facility and that $\left(T_1,\ T_2,\ T_3,\ T_4\right)$ is a proposed set of cycle times for these four products. Then a feasible schedule

$$\Omega^4 = \left[\left(T_1,\ T_2,\ T_3,\ T_4\right),\ \left(d_1,\ d_2,\ d_3,\ d_4\right),\ \Pi\right]$$

of the four products exists only if there is at least one feasible partial schedule Ω^3 of three of the products where at least one of the following three sets of inequalities are satisfied for the product not in Ω^3:

Set 1: $a_{14} \geq q_1 + q_4$ (266)

$$a_{24} \geq q_2 + q_4 + \left(a_{124} - w_{124}^2\right) \tag{267}$$

$$a_{34} \geq q_3 + q_4 + \left(a_{134} - w_{134}^2\right) \tag{268}$$

Set 2: $$a_{14} \geq q_1 + q_4 + w_{124}^2 \tag{269}$$

$$a_{24} \geq q_2 + q_4 \tag{270}$$

$$a_{34} \geq q_3 + q_4 + \left(a_{234} - w_{234}^3\right) \tag{271}$$

Set 3: $$a_{14} \geq q_1 + q_4 + w_{134}^3 \tag{272}$$

$$a_{24} \geq q_2 + q_4 + w_{234}^3 \tag{273}$$

$$a_{34} \geq q_3 + q_4. \tag{274}$$

Furthermore, at least one feasible schedule of the four products will exist if there is a Ω^3 with a d_3 such that at least one of the following three sets of inequalities are satisfied for the product not in Ω^3:

Set 1: $$a_{14} \geq q_1 + q_4 \tag{275}$$

$$a_{24} \geq q_2 + q_4 + \left(\beta_{234} a_{124} - w_B\right) \tag{276}$$

$$a_{34} \geq q_2 + q_4 + \left(\beta_{234} a_{134} - w_B\right) \tag{277}$$

Set 2: $$a_{14} \geq q_1 + q_4 + w_B \tag{278}$$

$$a_{24} \geq q_2 + q_4 \tag{279}$$

$$a_{34} \geq q_3 + q_4 + \Delta_{43} \tag{280}$$

Set 3: $$a_{14} \geq q_1 + q_4 + w_b^2 \quad (281)$$

$$a_{24} \geq q_2 + q_4 + \Delta_{42} \quad (282)$$

$$a_{34} \geq q_3 + q_4. \quad (283)$$

To simplify the notation this statement of the theorem assumes the products are labeled so that $\Pi i = i$.

Proof of Theorem 7

To further explain theorem 7, consider the results of theorem 6 in chapter 7. If (213) is satisfied then a compact partial schedule of three products exists with

$$v_{31} = w_{123}^2 \quad (284)$$

and

$$v_{32} = 0. \quad (285)$$

On the other hand, if (216) is satisfied a compact partial schedule of three products exists with

$$v_{31} = 0 \quad (286)$$

$$v_{32} = \left(a_{123} - w_{123}^2\right). \quad (287)$$

The proper set of these values of v_{31} and v_{32} must be used in (260), (261), (264) and (265) to test the inequalities of theorem 7. However, if both (213) and (216) are satisfied, the inequalities must be tested for both sets of $\left(v_{31},\ v_{32}\right)$. Except for this mild dependence, the necessary conditions (266) through (274) are independent of the delay times. However, these necessary conditions must be tested for every sequence Π of the four products where a feasible partial schedule Ω^3 of products $\Pi 1$, $\Pi 2$, and $\Pi 3$ exist. A vector of cycle times $\left(T_1,\ T_2,\ T_3,\ T_4\right)$ is rejected only after all such tests with the necessary conditions have failed.

The delay-independent form of the sufficient conditions (275) through (283) require more tests. Like the necessary conditions the sufficient conditions must also be tested for every feasible Π related to each sub-group of three products. In addition, these sufficient conditions must be tested at each value of d_3 that is feasible where $v_{31} = 0$ or $v_{32} = 0$. Other feasible d_3 do not have to be considered because the search for a feasible schedule is restricted to a search over compact schedules. Finally, the defining equations (262) through (265) use the factor β_{234}. Similar definitions are possible with β_{124} and β_{134} that leads to sufficient conditions similar to (275) through (283).

Observe that the necessary conditions (266) through (274) are equivalent to the sufficient conditions (275) through (283) if

$$\beta_{234} = 1. \qquad (288)$$

This is also true for the sufficient conditions related to β_{124} and β_{134} when either $\beta_{124} = 1$ or $\beta_{134} = 1$.

As was done for the proof of theorem 6 this proof will be organized into nine major parts. One of these parts has two sub-parts.

Part 1 – Compact Schedule

Because of theorem 4 on compact schedules in chapter 6 the search for a feasible schedule can be restricted to a search over compact schedules. For the remainder of the proof it will be assumed that all schedules are compact.

Part 2 – Feasibility of Products 1, 2, and 3

Let $\tau = \left(\Pi 1,\ \Pi 2,\ \Pi 3\right)$ be the set of indices for any one of the four possible subsets of three products taken from a set of four products. From lemma 8 it is known that a feasible schedule of the four products with a specified vector of cycle times $\left(T_{\Pi 1},\ T_{\Pi 2},\ T_{\Pi 3},\ T_{\Pi 4}\right)$ exists only if there is a feasible partial schedule of the three products in τ. Furthermore, this must be true for all four possible τ.

The search for a feasible schedule is limited to compact schedules; thus, it can be assumed that the delay times $\left(d_{\Pi 1},\ d_{\Pi 2},\ d_{\Pi 3}\right)$ are given by (201), (202), (204) and (205) found in chapter 7. For the remainder of the proof it is assumed that for any τ the schedule $\Omega^3 = \left(\{T_i\},\ \{d_i\},\ \Pi \,|\, i\varepsilon\tau\right)$ is feasible. Otherwise, the specified cycle times do not satisfy the hypothesis of theorem 7 so no further proof is required.

To simplify the notation it is assumed that the three products of the schedule Ω^3 are labeled so that $\Pi i = i$. It is also assumed that $\Pi 4 \notin \tau$ is labeled as product 4. This shows that the remainder of the proof must only be concerned with schedule conflicts between product 4 and each of the other three products. The feasibility of Ω^3 assures one that no other schedule conflicts occurs.

Part 3 – Delay Time Dependent Feasibility Conditions

When applied to product 4 the revised delay time dependent feasibility conditions (145) through (148) given in chapter 6 yield

$$d_4 = d_1 + k_{41}a_{14} + q_1 + v_{41} \tag{289}$$

$$d_4 = d_2 + k_{42}a_{24} + q_2 + v_{42} \tag{290}$$

and

$$d_4 = d_3 + k_{43}a_{34} + q_3 + v_{43} \tag{291}$$

where

$$0 \leq v_{4i} \leq a_{i4} - q_i - q_4 \tag{292}$$

for *i, 1, 2, 3*. By substituting (201) and (202) into (289) and (290) and equating the right hand sides of each pair of the resulting equations one obtains the following three feasibility equations

$$k_{41}a_{14} - k_{42}a_{24} = q_2 - v_{41} + v_{42} \tag{293}$$

$$k_{43}a_{34} - k_{41}a_{14} = -d_3 + q_1 - q_3 + v_{41} - v_{43} \tag{294}$$

and

$$k_{42}a_{24} - k_{43}a_{34} = d_3 - q_1 - q_2 + q_3 - v_{42} + v_{43}. \tag{295}$$

Clearly, the conditions imposed on the schedule of product 4 by the general theorem 1 given in chapter 3 are satisfied if and only if (293) through (295) are satisfied. These feasible equations are three equations in the six unknowns k_{4i} and v_{4i}, $i = 1, 2, 3$. The other terms become fixed constants once one specifies a set of four cycle times $\{T_i\}$ and a feasible partial schedule Ω^3 of the first three products. Hence, a feasible schedule of the four products exists if and only if there are k_{4i} and v_{4i}, $i = 1, 2, 3$ that satisfy (293) through (295).

Determining the feasibility of four products requires solving three feasibility equations rather that the one feasibility equation of the three products model. Furthermore, as (293) through (295) show these integer equations are highly intertwined. Finding general solutions to the feasibility equations of the four-products model is a much more challenging task than solving the three-products feasibility equation.

Part 4 – Decomposition of Feasibility Equations

Lemma 7 on solutions to an integer equation asserts that all integer solutions to the left-hand sides of (293), (294) and (295) are given by

$$k_{41}a_{14} - k_{42}a_{24} = \lambda_{124}a_{124} \tag{296}$$

$$k_{43}a_{34} - k_{41}a_{14} = \lambda_{134}a_{134} \tag{297}$$

$$k_{42}a_{24} - k_{43}a_{34} = \lambda_{234}a_{234} \tag{298}$$

where all parameters and variables are integers. This result allows a decomposition of the feasibility equations into left-hand and right-hand parts. It will be shown that these two parts can be treated independently.

Of course, any one of the three feasibility equations is redundant. However, as will be seen later, having the flexible to select which two equations to solve is not only convenient but necessary to the proof. Also, the redundant equation places additional restraints on any integer solution to the feasibility equations. In particular, by summing the left-hand equations one finds that

$$\lambda_{124}a_{124} + \lambda_{134}a_{134} + \lambda_{234}a_{234} = 0. \tag{299}$$

Thus, for a valid integer solution to exist there must be integers λ_{124}, λ_{134} and λ_{234} that satisfy (299). To complete the decomposition of the feasibility equations the right-hand feasibility equations are:

$$\lambda_{124}a_{124} = q_2 - v_{41} + v_{42} \tag{300}$$

$$\lambda_{134}a_{134} = -d_3 + q_1 - q_3 + v_{41} - v_{43} \tag{301}$$

$$\lambda_{234}a_{234} = d_3 - q_1 - q_2 + q_3 - v_{42} + v_{43}. \tag{302}$$

Part 5 – Solving the Left-hand Equations

To find integer solutions for the left-hand equations, (296), (297) and (298) are initially disconnected by rewriting the equations as

$$k_{41}a_{14} - k_{42}a_{24} = \lambda_{124}a_{124} \tag{303}$$

$$k_{43}a_{34} - k_{41}^{2}a_{14} = \lambda_{134}a_{134} \tag{304}$$

$$k_{42}^{2}a_{24} - k_{43}^{2}a_{34} = \lambda_{234}a_{234}. \tag{305}$$

Lemma 7 in chapter 6 shows that there are α_{4i}, α_{4i}^{2}, $i = 1, 2, 3,$ such that all solutions to (303), (304), and (305) are of the form

$$k_{41} = \left(m_1\beta_{24}\beta_{234} + \lambda_{124}\mu_{41}\right) \tag{306}$$

$$k_{41}^{2} = \left(m_2\beta_{34}\beta_{234} + \lambda_{134}\mu_{41}^{2}\right) \tag{307}$$

$$k_{42} = \left(m_1\beta_{14}\beta_{134} + \lambda_{124}\mu_{42}\right) \tag{308}$$

$$k_{42}^{2} = \left(m_3\beta_{34}\beta_{134} + \lambda_{234}\mu_{42}^{2}\right) \tag{309}$$

$$k_{43} = \left(m_2\beta_{14}\beta_{124} + \lambda_{134}\mu_{43}\right) \tag{310}$$

$$k_{43}^{2} = \left(m_3\beta_{24}\beta_{124} + \lambda_{234}\mu_{43}^{2}\right). \tag{311}$$

Thus, solutions for the left-hand equations exist if and only if there is an **i = 1, 2, 3** such that

$$k_{4i} = k_{4i}^{2}. \tag{312}$$

Suppose that $k_{41} = k_{42}^2$ then

$$m_1\beta_{24}\beta_{234} + \lambda_{124}\mu_{41} = m_2\beta_{34}\beta_{234} + \lambda_{134}\mu_{41}^2$$

or

$$\left(m_1\beta_{24} - m_2\beta_{34}\right)\beta_{234} = \lambda_{134}\mu_{41}^2 - \lambda_{124}\mu_{41}. \qquad (313)$$

By setting $m_1 = m_2 = 0$ one obtains

$$\mu_{41}\beta_{14}\beta_{134} - \mu_{42}\beta_{24}\beta_{234} = 1 \qquad (314)$$

and

$$\mu_{43}\beta_{34}\beta_{234} - \mu_{41}^2\beta_{14}\beta_{124} = 1. \qquad (315)$$

An examination of (314) and (315) reveals that β_{234} is relative prime to both μ_{41} and μ_{41}^2. Let

$$\in_{41} = gcd\left(\mu_{41},\ \mu_{41}^2\right), \qquad (316)$$

$$\mu_{41} = \gamma_{41} \in_{41} \qquad (317)$$

and

$$\mu_{41}^2 = \gamma_{41}^2 \in_{41} \qquad (318)$$

Substituting (317) and (318) into (313) yields

$$\left(m_1\beta_{24} - m_2\beta_{34}\right) = \theta \in_{41} \tag{319}$$

and

$$\left(\lambda_{134}\gamma_{41}^2 - \lambda_{124}\gamma_{41}\right) = \theta\beta_{234} \tag{320}$$

for some integer θ. Lemma 8 asserts that there are m_1, m_2 that satisfy (319) for any integer θ. Also lemma 8 shows that there are δ_{41}, δ_{42}^2 such that all solutions to (320) are given by

$$\lambda_{124} = y_{41}\gamma_{41}^2 + \theta\beta_{234}\delta_{41} \tag{321}$$

and

$$\lambda_{134} = y_{41}\gamma_{41} + \theta\beta_{234}\delta_{41}. \tag{322}$$

As will be shown later solving the right-hand feasibility conditions requires a minimization of v_{41}, v_{42}, and v_{43} with respect to λ_{124}, λ_{134} and λ_{234}. In the general four-products model this minimization is constrained by (321) and (322). No a priori solution to this constrained minimization problem has been found except for special cases.

One such special case is defined by requiring that $y_{41} = y_{42}^2\beta_{234}$ so that

$$\gamma_{124} = \gamma_{124}^2\beta_{234} \tag{323}$$

and

$$\lambda_{134} = \lambda_{134}^2\beta_{234}. \tag{324}$$

Substituting (323) and (324) into (320) yields

$$\lambda_{134}^{2}\gamma_{41}^{2} - \lambda_{124}^{2}\gamma_{41} = 0. \tag{325}$$

These relationships are important because, for this special case, one can independently select any integer values for λ_{124}^{2} and λ_{134}^{2} and be guaranteed by lemma 8 that there are m_1 and m_2 that satisfy (319). Thus, for this special case the minimization problems of part 6 are disconnected to become two unconstrained minimization problems.

Of course, special cases analogous to (323) and (324) can be defined for the factors β_{124} and β_{134}. The results for these other cases can be generated from the β_{234} case by merely re-labeling the first three products. If any $\beta_{ijk} = 1$ then clearly the sufficient conditions related to that β_{ijk} are also necessary feasibility conditions.

Part 6 – Solving the Right-hand Equations

There are two major sub-parts to part 6. First a proof is provided for the necessary conditions (266) through (274). Then a similar proof is given for the sufficient conditions (275) through (283). The principal difference between these proofs is that in the proof of the necessary conditions the variables of the right-hand feasibility equations (300) through (302) are not required to simultaneously satisfy all three equations. This relaxation of equality converts the feasibility equations into independent constraints that are not necessarily satisfied by the same set of variable values.

Part 6.A – Necessary Conditions

The proof begins with new expressions of d_3 derived from (204) and (205) of chapter 7. By applying the factorization given by (252) and (254) one obtains

$$a_{13} = \beta_{13}\beta_{123}a_{134} \tag{326}$$

and

$$a_{23} = \beta_{23}\beta_{123}a_{234}. \tag{327}$$

Substituting (201), (202), (326), and (327) into (204) and (205) yields

$$d_3 = k_{31}\beta_{13}\beta_{123}a_{134} + q_1 + v_{31} \tag{328}$$

and

$$d_3 = k_{32}\beta_{23}\beta_{123}a_{234} + q_2 + v_{32}. \tag{329}$$

By substituting (257), (260), (261), (328) and (329) into (300), (301) and (302) one obtains the following revised right-hand feasibility equations.

$$-v_{41} + v_{42} = \left(\lambda_{124} - u^2_{124}\right)a_{124} - w^2_{124} \tag{330}$$

$$v_{41} - v_{43} = \left(\lambda_{134} + k_{31}\beta_{13}\beta_{123} + u^3_{134}\right)a_{134} + w^3_{134} \tag{331}$$

$$-v_{42} + v_{43} = \left(\lambda_{234} - k_{32}\beta_{23}\beta_{123} - u^3_{234}\right)a_{234} - w^3_{234}. \tag{332}$$

The revised feasibility conditions (145) through (148) in chapter 6 show that

$$v_{41} \geq 0 \tag{333}$$

and

$$a_{i4} \geq q_i + q_4 + v_{4i} \tag{334}$$

for $i = 1, 2, 3$. Thus, (330) through (334) are a set of five constraints in six unknowns.

As noted in part 5 no a priori solution to these five constraints has been found. However, by treating each feasibility equation independently it is possible to place a lower bound on each v_{41}. This, in turn, provides a lower bound on each a_{i4} that must be satisfied by any feasible schedule of the four products.

Because only compact schedules are being considered at least one $v_{4i} = 0$. Thus, there are three cases to be considered. The lower bounds on a_{i4} developed for each of these cases is one set of the necessary conditions (266) through (274).

An examination of the feasibility equations (330), (331) and (332) reveals the usefulness of the redundant equations. When any $v_{4i} = 0$ and λ_{124}, λ_{134} and λ_{234} are specified two of these feasibility equations become equations with only one unknown v_{4i} each. Hence, one can easily find lower bounds for the non-zero v_{4i}.

To find the first set of necessary feasibility conditions suppose that $v_{41} = 0$ and set

$$\lambda_{124} = u_{124}^2 + 1 \tag{335}$$

and

$$\lambda_{134} = -\left(k_{31}\beta_{13}\beta_{123} + u_{134}^{2} + 1\right) \tag{336}$$

in (330) and (331), respectively. With these values the feasibility constraints become

$$v_{41} = 0 \tag{337}$$

$$v_{42} \geq a_{124} - w_{124}^{2} \tag{338}$$

$$v_{43} \geq a_{134} - w_{134}^{3}. \tag{339}$$

These bound are valid because any values for δ_{124} or δ_{134} other than (335) and (336) yields either a greater value for v_{42} or v_{43} or negative values for v_{42} or v_{43}. The relationships (338) and (339) must be expressed as lower bounds. In general, it is not possible to simultaneously set λ_{124} and λ_{134} according to (335) and (336) because there may be no solutions to the left-hand feasibility equations when λ_{124} and λ_{134} are set to these values. The lower bounds (337) through (339) prove that if $v_{41} = 0$ then the first set of necessary conditions (266), (267) and (268) must be satisfied by any feasible schedule of four products. One can prove that the other two sets of feasibility conditions (278) through (283) are valid by repeating the above proof when $v_{42} = 0$ and then when $v_{43} = 0$.

Part 6.B – Sufficient Conditions

The proof in part 5 shows that when (323) and (324) are satisfied it is always possible to find k_{4i}, $i = 1, 2, 3,$ which solves the first two left-hand equations (296) and (297). This means that the first two right-hand equations can be solved for any specified integers λ^2_{124} and λ^2_{134}. Hence, the third feasibility equation is dropped because it is redundant and unnecessary.

By substituting (262), (263), (335) and (336) into (300) and (301) one obtains the two right-hand feasibility equations

$$-v_{41} + v_{42} = \left(\lambda^2_{124} - u_B\right)\beta_{234}a_{124} - w_B \tag{340}$$

$$v_{41} - v_{43} = \left(\lambda^2_{134} + u^2_B\right)\beta_{234}a_{134} + w^2_B. \tag{341}$$

When $v_{41} = 0$ using equations (340) and (341) to prove the validity of the first set of sufficient conditions (275), (276) and (277) is a straightforward task.

The other two cases have an additional complication. For instance, when $v_{42} = 0$ then

$$min\ v_{41} = w_B \tag{342}$$

and

$$v_{43} = -\left(\lambda^2_{134} + u^2_B\right)\beta_{234}a_{134} + v_{41} - w^2_B \tag{343}$$

observe that the minimum of v_{43} occurs at the minimum of v_{41}. Thus, if $w_B \geq w^2_B$ one easily finds that

$$min\ v_{43} = w_B - w^2_B \tag{344}$$

occurs at $\delta^2_{134} = -u^2_B$. The other case is also easily found by observing that when $w^2_B > w_B$

$$\beta_{234}a_{134} > w_B^2 > w_B^2 - w_B \qquad (345)$$

so that over positive values

$$\min\, v_{43} = \beta_{234}a_{134} - \left(w_B^2 - w_B\right) \qquad (346)$$

occurs at $\lambda_{134}^2 = -\left(u_B^2 + 1\right)$. Thus, Δ_{43} given by (265) is the minimum of v_{43} when $v_{42} = 0$. This immediately leads to the sufficient conditions (278), (279) and (280).

By setting $v_{43} = 0$ and repeating the above argument one finds that

$$\min\, v_{42} = \Delta_{42}$$

where Δ_{42} is defined by (264). With this substitution the feasibility equations (340) and (341) show that the third set of sufficient conditions (281), (282) and (283) are valid.

Part 7 – Feasibility Conditions Independent of Ω^3

As explained in the introduction to the proof of theorem 7, to reject a set of specified cycle times $\left(T_1, T_2, T_3, T_4\right)$ as infeasible one must use (266) through (274) to test every potentially feasible sequence Π of the four products. Once a sequence is specified the delay times for the first three products are given by theorem 6 in chapter 7. The necessary conditions are independent of these delay times except for the mild dependence that enters through equations (260) and (261). The three-products variables v_{31} and v_{32} are constants in (260) and (261). This mild dependence is removed by repeating the feasibility tests for each of the two possible values of v_{31}, v_{32} that are feasible. These sets of values are given by (284) through (287).

Part 8 – Constructing a Feasible Schedule

Suppose that $\left(T_1, T_2, T_3, T_4\right)$ satisfy the sufficient conditions for some sequence Π and some feasible $d'_{\Pi 3}$. A feasible schedule can be constructed where

$$d_{\Pi 1} = 0$$
$$d_{\Pi 2} = q_1$$

and

$$d_{\Pi 3} = d'_{\Pi 3}.$$

To complete the construction one uses Euclid's algorithm to find μ_{41}, μ_{41}^2, μ_{42}, μ_{43}, m_1 and m_2 in (314), (315) and (319). The integer θ is derived from (325) where λ_{124}^2 and δ_{124}^2 are obtained from the solution of the right-hand sufficient conditions. Then k_{4i}, $i = 1, 2, 3$ can be determined from (306), (308) or (310). Finally, a feasible d_4 can be derived from (289), (290) or (291).

Part 9 – Conclusion

In summary, theorem 7 provides a set of necessary conditions that must be satisfied by all feasible schedules of a four-products GCM. These necessary conditions are used in the optimization algorithm to quickly eliminate as many infeasible partial schedules as possible.

The theorem also provides a set of sufficient conditions. When satisfied these sufficient conditions provide assurance that at least one feasible schedule exists for a specified vector of four cycle times. One of the principal benefits of these sufficient conditions is that they reveal the "penalty" that must be paid to schedule four products. As shown by the sufficient conditions (275) through (283) in most cases two of the greatest common divisors a_{i4}, $i = 1, 2, 3$ must "pay" this "penalty".

LEMMA 10 ON FOUR-PRODUCTS PARTIAL SCHEDULES

Let $\tau_k, 4 \leq k \leq n,$ be a subset of n products produced on a single facility and let $\Psi_k = \left\{t_i \middle| i \varepsilon \tau_k\right\}$ be a specified set of cycle times for the products $i \varepsilon \tau_k$. Suppose that $\theta_4 = \left(t_j,\ t_m,\ t_\rho,\ t_w\right)$ is any set of four cycle times in τ_k. Let Ω be any full schedule of the n products where $T_i = t_i$ for $i \varepsilon \tau_k$. Then Ω is not feasible if there exist a θ_4 such that the $t_i \varepsilon \theta_4$ do not satisfy the hypothesis of theorem 7.

PROOF OF LEMMA 10

This lemma is an extension to lemma 8 on three-products partial schedules given in chapter 7. The proof of lemma 10 follows immediately from theorem 5 on partial schedule feasibility given in chapter 6 and theorem 7.

REMARKS ON LEMMA 10

Like lemmas 6 and 8, lemma 10 provides useful "filters" that could be used by the optimization algorithm of chapter 9 to eliminate many infeasible partial schedules at each stage of the algorithm after the third. However, lemma 11 provides tests that enhances the "filtering" capability of theorem 7 beyond that of the straightforward tests provided by lemma 10.

ENHANCED FOUR-PRODUCTS CONDITIONS

Purpose

The hypothesis of theorem 7 only requires that one set of the necessary conditions (266) through (274) be valid for some sequence Π. That is, theorem 7 supposes that all possible sequences Π of the four products will be tested with the necessary conditions (266) through (274). A set of four cycle times is not rejected unless all Π fail these tests. The proof of theorem 7 also supposes that the schedule Ω^3 of the first three products will be a compact one. Thus, it is supposed that either

$$d_{\Pi 1} = v_{\Pi 2,\, \Pi 1} = v_{\Pi 3,\, \Pi 1} = 0 \tag{347}$$

or

$$d_{\Pi 1} = v_{\Pi 2,\, \Pi 1} = v_{\Pi 3,\, \Pi 2} = 0. \tag{348}$$

However, one is not free to make these choices as schedules are constructed by the optimization algorithm in chapter 9. In this algorithm new partial schedules are generated at each stage by adding another product and its schedule to an existing partial schedule Ω. Thus, neither (347) nor (348) may be true for an arbitrarily selected set of three product from those contained in Ω. The purpose of lemma 11 is to determine when $T_{\Pi 4}$ is incompatible with the existing partial schedule Ω.

Lemma 11 on Enhanced Four-Products Conditions

Let τ_k, $4 \leq k \leq n$, be a subset of the indices of n products produced on a single facility and let $\Psi_k = \left\{ t_i \,\middle|\, i \varepsilon \tau_k \right\}$ be a specified set of cycle times for the products $i \varepsilon \tau_k$. Let $\Omega_k = \left(\{t_i\}, \{d_i\}, \Pi \right)$ be a feasible partial schedule of the products $i \varepsilon \tau_k$. Suppose product m is added to τ_k to create τ_{k+1}. Let τ_3 be any set of three products $i \varepsilon \tau_k$. Without loss of generality assumes that the products have been labeled such that $\tau_3 = (1, 2, 3)$ and $m = 4$ where $d_1 < d_2 < d_3$. Let Ω_4 be a proposed schedule of product 4 where $T_4 = t_4$. Further, let

$$\Omega_{k+1} = \Omega_k + \Omega_4$$

be the partial schedule for the products $i \varepsilon \tau_{k+1}$ where $T_i = t_i$ for $i \varepsilon \tau_k$ and $T_4 = t_4$. Apply the division algorithm to obtain

$$q_2 + v_{21} = y_{124}^2 a_{124} + x_{124}^2. \tag{349}$$

Then a feasible full schedule Ω that contains Ω_{k+1} where $T_4 = t_4$ exists only if at least one set of the necessary conditions (266) through (274) are satisfied when $w_{124}^2 = x_{124}^2$.

Proof of Lemma 11

The principal difference between lemma 11 and theorem 7 is that, in general, d_1 and d_2 are not given by the compact schedule values. Thus, in general, $d_1 \neq 0$, $d_2 \neq q_1$, and $v_{21} \neq 0$. By manipulating the transformed equation (145) for $d_i - d_j$ given in chapter 6 one obtains the following general expressions regarding d_4

$$d_4 - d_1 = k_{41}a_{14} + q_1 + v_{41} \tag{350}$$

$$d_4 - d_1 = d_2 - d_1 + k_{42}a_{24} + q_2 + v_{42} \tag{351}$$

$$d_4 - d_1 = d_3 - d_1 + k_{43}a_{34} + q_3 + v_{43} \tag{352}$$

where

$$d_2 - d_1 = k_{21}a_{12} + q_1 + v_{21} \tag{353}$$

$$d_3 - d_1 = k_{31}a_{13} + q_2 + v_{31} \tag{354}$$

and

$$d_3 - d_2 = k_{32}a_{23} + q_2 + v_{32}. \tag{355}$$

Substitute the left-hand feasibility equations (296), (297) and (298) and (260), (261) and (349) into (353), (354) and (355) and express a_{12}, a_{13} and a_{23} as given by the factorization equations (251), (252) and (254) to obtain

$$-v_{41} + v_{42} = \left(\lambda_{124} - k_{21}\beta_{12}\beta_{123} - y_{124}^{2}\right)a_{124} - x_{124}^{2} \tag{356}$$

$$v_{41} - v_{43} = \left(\lambda_{134} + k_{31}\beta_{13}\beta_{123} + u_{134}^{3}\right)a_{134} + w_{134}^{3} \tag{357}$$

$$-v_{42} + v_{43} = \left(\lambda_{234} - k_{34}\beta_{23}\beta_{123} - u_{234}^{3}\right)a_{234} - w_{234}^{3}. \tag{358}$$

Comparing (356), (357) and (358) with the right-hand feasibility equations (330), (331) and (332) in part 6.A of the proof of theorem 7 reveals three differences, all in equation (356) versus (330). The first two difference is that the multiplier of a_{124} has an extra term and u_{124}^{2} is replaced with y_{124}^{2}. Neither of these differences has an effect on the necessary feasibility conditions because the development of these conditions assumes that λ_{124} can be set to any desired value.

The last difference is that w_{124}^{2} in (330) has been replaced with x_{124}^{2} in (356). Thus, if one sets

$$w_{124}^{2} = x_{124}^{2}$$

then the proof of the necessary conditions given in part 6.A of theorem 7 shows that lemma 11 is valid.

After the third stage of the optimization algorithm given in chapter 9 three or more products have been scheduled during previous stages. That is, the cycle times and delay times of these scheduled products are specified during previous stages. Suppose at stage m, $4 \leq m \leq n$, a product m is proposed as the ***mth*** product and a cycle time t_m is proposed for T_m. Then all groups of three of the scheduled products, with their specified cycle times and delay times, can be tested by lemma 11. Suppose there is a group of three products such that none of the three sets of necessary conditions (266) through (274) are satisfied. In this case t_m can be rejected as a possible value of T_m unless the partial schedule developed in stages before the ***mth*** is changed.

CHAPTER 9

OPTIMIZATION ALGORITHM

PURPOSE

The purpose of this chapter is to present an algorithm that produces optimal schedules for the ***n***-products GCM. As shown by theorem 3 in chapter 5 the algorithm given in chapter 4 will find optimal schedule for the two-products GCM. Thus, in this chapter it is assumed that $n \geq 3$. The ***n***-products algorithm is the culmination of the results given in the previous chapters. The general necessary and sufficient conditions in chapter 3 are essential to the creation of an optimization algorithm. Without these conditions it is not possible to mathematically determine whether or not a schedule is feasible.

The results of chapters 4 through 8 improve the efficiency of the optimization algorithm. These improvements are also essential to the success of the algorithm. Gallego and Shaw [26] show that the GCM is "NP-hard in the strong sense". Thus, optimizing the GCM will require some type of implicit enumeration scheme. Optimization algorithms based on implicit enumeration schemes must be very efficient if they are to provide solutions to practical problems within a reasonable time and with reasonable computer resources. The efficiency of the algorithm presented here is enhanced in two major ways. Cost bounds are used to eliminate many non-optimal schedules and the results of chapters 3 through 8 are used to eliminate many infeasible schedules.

SUMMARY OF OPTIMIZATION ALGORITHM

As noted above the algorithm is based on an implicit enumeration scheme. This scheme is quite different from the usual branch and bound implicit enumeration method. The algorithm begins by finding an optimal schedule for the BPM. The BPM is a restricted version of the GCM so the cost of this schedule is used as the initial upper bound on the optimal cost of the GCM z^*. The BPM algorithm is used to establish a lower bound on z^*. The initial search domain is defined by using these bounds on the optimal cost and the strict convexity of the cost functions to place bound on each cycle time T_i.

The bulk of the algorithm is a n-stage implicit enumeration scheme. This scheme, if possible, generates a schedule for the n-products that has a lower cost than the previous best schedule. If a lower cost schedule is found it replaces the previous best schedule and its cost replaces the previous cost upper bound. Then a new search domain is obtained by updating the limits on the cycle times to reflect the revised upper bound. Eventually, the cost of all remaining schedules are greater than the previous best schedule. When this occurs the previous best schedule is optimal. The algorithm produces a series of schedules with costs that are a monotonically decreasing series. This series is forced to be strictly monotonically decreasing by subtracting an acceptable amount from each new cost upper bound. Because this series is bounded below by the finite initial lower bound the algorithm will terminate within a finite number of steps.

One is assured that the final schedule is optimal because the algorithm implicitly examines every possible schedule of the n products and selects one with minimum cost. As the algorithm searches over the domain of potentially optimal schedules, it eliminates many partial schedules. This, of course, eliminates a greater number of full schedules. The design of the algorithm prevents these schedules from being eliminated unless they are either non-optimal or infeasible.

In several places in the description of the algorithm it is stated that the algorithm terminates. Terminate here means that the current best schedule is output as the optimal schedule, the current upper bound on the total cost of the n-products Z_u is output as the optimal cost and the algorithm stops.

An important aspect of the algorithm has to do with the structure of the search region. The beginning search region is defined by placing a lower and an upper bound on the cycle time of each product. These bounds define a region of potentially optimal schedules. At each stage the algorithm searches for an optimal, feasible cycle time for the products assigned to that stage. It begins this search in the "middle" of the search region defined by the initial cycle time bounds of the assigned product.

If this "middle" value is excluded from further consideration it becomes the first cycle time value in the exclusion zone of the search region. Thereafter, the search region is divided into three mutually exclusive zones: a lower search zone, an exclusion zone, and an upper search zone. Each trial cycle time is either the greatest integer less than the lower limit of the exclusion zone or the smallest integer greater than the upper limit of the exclusion zone. If a trial cycle time is rejected it is added to the exclusion zone before a new trial cycle time is selected.

Thus, as the algorithm proceeds the exclusion zone grows while the lower and upper zones shrink. At different times each of these zones may be the null set. In any case, each of the zones of the search region for the assigned product remains contiguous throughout the algorithm. The mechanics required to implement this partitioning of the search region is explained as the steps of the algorithm are described below.

BASIC N-PRODUCTS ALGORITHM

To make a clear presentation the essential framework of the algorithm will be given in this section. Several enhancements to the basic algorithm will be described in the following section.

Stage 0

The algorithm actually has $n + 1$ stages where stage 0 consists of certain preliminary steps that are only performed once for each scheduling problem. The results of stage 0 are used throughout the remaining steps.

Step 1 – Obtain an Upper Bound on Schedule Cost

Use the algorithm of chapter 4 to solve the BPM version of the current problem. The optimal schedule of the BPM is a feasible schedule in the GCM. Thus, the cost of the optimal BPM schedule is an upper bound on the optimal cost of the GCM. The BPM optimal schedule is saved as the current best schedule. The cost of the optimal BPM schedule less a small positive $\in$ is saved as the current cost upper bound Z_u. This assures that the next cost upper bound, if any, will be strictly less than the cost the BPM optimum schedule.

Step 2 – Obtain a Lower Bound on z^*

For each pair of products use the BPM algorithm to find the optimal cost when these products are considered the products of a two-products model. Denote the BPM optimal cost related to product ***i, j*** as Γ_{ij}°. As shown by theorem 3 in chapter 5, Γ_{ij}° is also the minimum cost of products ***i*** and ***j*** in the two-products GCM. Clearly, Γ_{ij}° is a lower bound on the combined cost of products ***i*** and ***j*** in the ***n***-products GCM. Thus, a lower bound on z^* is given by:

$$Z_l^{\circ} = \left[\sum_{i-1}^{n-1} \sum_{j=i+1}^{n} \Gamma_{ij}^{\circ}\right] / (n-1). \tag{359}$$

If the cost upper bound equals the cost lower bound then the optimal schedule of the BPM found in step 1 is also an optimal schedule for the GCM. In this case terminate the algorithm.

Step 3 – Obtain Single Product Optimal Cycle Times

In this step the algorithm finds an optimal solution to the ***n***-single-product EMQ models by treating each product as if it is the only product of a single product model. That is, solve (68) and (69) in chapter 4 to obtain the integer single-product optimal cycle times for each of the n products.

Step 4 – Obtain Bounds on the Cycle Times

By rearranging (359) to extract the cost contribution of product k one obtains

$$Z_{lk}^{\circ} = \left[\sum_{\substack{i=1 \\ i\neq k}}^{n-1} \sum_{\substack{j=i+1 \\ j\neq k}} \Gamma_{ij}^{\circ} \right] / (n-2). \tag{360}$$

This is a lower bound on the sum of the costs of all products except product k. Clearly,

$$\theta = Z_u - Z_{lk}^{\circ} \tag{361}$$

is an upper bound on the cost of product k in any schedule with a cost lower than the current upper bound Z_u. If the algorithm is not terminated in steps 2 then $Z_u > Z_l^{\circ}$. Also, it is easily shown that

$$Z_l^{\circ} \geq z_{sk}^{*} + Z_{lk}^{\circ} \tag{362}$$

where z_{sk}^{*} is the optimal cost of the ***kth*** single product model. Hence, the quadratic in T_k

$$c_k T_k + H_k T_k = \theta, \tag{363}$$

where c_k is the setup cost and H_k is defined by (51) in chapter 4, always has two positive real roots. These two roots are lower and upper bounds on the cycle time of product ***k***. The bounds on the integer T_k are

$$T^{\circ}_{lk} = \left\{\left[\theta - \sqrt{\theta - 4H_k c_k}\right] / 2H_k\right\}^{+} \leq T_k \tag{364}$$

$$T^{\circ}_{uk} = \left\{\left[\theta + \sqrt{\theta - 4H_k c_k}\right] / 2H_k\right\}^{-} \geq T_k. \tag{365}$$

The number of potentially optimal cycle times for each product ***k*** is given by the range count

$$R^{\circ}_k = T^{\circ}_{uk} - T^{\circ}_{lk} + 1. \tag{366}$$

If the range count is zero for one or more of the products there is no schedule of integer cycle times that has a cost lower than the current upper bound Z_u so the algorithm is terminated. Otherwise, the range count R°_k is a finite upper limit on the number of possible values of T_k that must be examined by the algorithm. This assures that the algorithm will require only a finite number of operations.

Step 5 – Assign Products to Stages

The products are sorted in ascending order of their range count R°_k. Then the products are relabeled according to this order so that the product with the ***ith*** largest range count is labeled as product ***i***. Finally, product ***i*** is assigned to the ***ith*** stage of the algorithm. This means that the cycle time and delay time of the ***ith*** product will be determined during the ***ith*** stage of the algorithm. In the basic algorithm this assignment of products to stages is performed once for each problem so it is fixed for the remainder of the algorithm.

<u>Step 6 – Select</u> d_1

As observed several times in previous chapters, because of the defining assumption i in chapter 1, without loss of generality, time zero can be selected so that the first production period of product 1 is set to zero. Thus the delay time for product 1 is set to zero. Setting $d_1 = 0$ simplifies the algorithm somewhat so the first delay time is set to zero for the remainder of the algorithm.

Stage 1

<u>Remarks</u>

As discussed above at each of the stages 1 through $n - 1$ the algorithm attempts to build a series of partial feasible schedules until a feasible full schedule is generated in stage n. Because limits (364) and (365) and similar ones are imposed on the schedule generation at each stage of the algorithm the cost of any full schedule generated in stage n will be less than the cost of the previous best schedule.

The schedule generation process begins and ends in stage 1. The algorithm ends after all R_1° possible values for the cycle time of product 1 have been eliminated. A value t_1 for cycle time T_1 is eliminated when

a. All feasible, potentially optimum, schedules where $T_1 = t_1$ have been generated,

b. t_1 is identified as infeasible, or

c. t_1 is identified as not optimum.

When all t_1 in the range $T_{l1}^{\circ} \leq T_1 \leq T_{u1}^{\circ}$ have been eliminated the algorithm is terminated.

<u>**Step 7 – Select**</u> t_1

Stage 1 is entered from stage 0, stage 2, or, perhaps, stage n. It is entered from stage 0 only once for each scheduling problem. Stage 1 may be entered from stage n only after a new feasible full schedule is generated. This will be explained when stage n is described below.

Suppose a feasible schedule Ω_s exists where the cycle time of each product equals its single-product optimal cycle time; that is, when $T_i = T_{si}^{*}$. Then Ω_s is clearly an optimal schedule. The search strategy of each stage of the algorithm is predicated on this observation and the strict convexity of the cost functions $Z_i(T_i)$.

When step 7 is executed for the first time set the cycle time of the first product to its single-product optimal; that is, set $t_1 = T_{s1}^{*}$ Then go to step 9 in stage 2.

Each time stage 1 is entered either from stage 2 or stage n the current value of T_1 is eliminated and a new value t_1 is selected. In particular, the second time stage 1 is entered the previous trial cycle time $t_1 = T_{s1}^{*}$ is rejected. Thus, the single-product optimal cycle time is the "middle" value used above to describe the partitioning of the search region for each product. As described above, once T_{s1}^* is rejected the search region of the first cycle time is partitioned into three zones. Let L_1 be the upper limit of the lower search zone and J_1 be the lower limit of the upper search zone. Then the search region of the optimal cycle time for product 1 is divided into a lower zone where

$$T_{l1} \leq t_1 \leq L_1 \leq T_{s1}^* \tag{367}$$

an upper zone where

$$T_{u1} \geq t_1 \geq J_1 \geq T_{s1}^* \tag{368}$$

and an exclusion zone where

$$L_1 < t_1 < J_1. \tag{369}$$

The exclusion zone consists of those cycle times t_1 that have been excluded from further consideration. The mechanics required to implement this strategy are to set

$$t_1 = L_1 = J_1 = T_{s1}^* \tag{370}$$

the first time that step 7 is executed. Thereafter, terminate the algorithm if both the lower and upper search zones are null. Otherwise, if the previous $t_1 = L_1$ then subtract one from L_1 If the previous $t_1 = J_1$ then add one to J_1. Then use these updated values of L_1, J_1 to set

$$z_{L1} = \begin{cases} \infty, & L_1 < T_{l1} \\ Z_1(L_1), & L_1 \geq T_{l1} \end{cases} \tag{371}$$

and

$$z_{J1} = \begin{cases} \infty, & J_1 < T_{u1} \\ Z_1(J_1), & J_1 \geq T_{u1} \end{cases} \tag{372}$$

and set the new trial value of T_1 to

$$t_1 = \begin{cases} L_1, & z_{L1} \leq z_{J1} \\ J_1, & z_{L1} > z_{J1} \end{cases} \tag{373}$$

Step 8 – Obtain New Bounds on T_i

If setting T_1 to its new trial value increases the cost lower bound given by (360) for one or more of the other products, then compute new cycle time bounds for each such product.

If $T_{lk} > T_{uk}$ for any k then terminate the algorithm. Terminating the algorithm here is the proper action because the present trial cycle time t_1 has the lowest cost among those cycle times that have not been rejected. Thus, selecting any other possible trial value will not decrease T_{lk} or increase T_{uk} for any k. Hence, the cost of all schedules in the current search region are greater than the present upper bound.

If the algorithm is not terminated go to step 9 in stage 2.

Stage m, $2 \leq m \leq n$

<u>**Step 9 – Select**</u> t_m

Stage m is entered from stage $m - 1$, stage $m + 1$ or, perhaps, stage n. During the course of solving a given problem stage m may be entered many times from each of these other steps. The mechanics of selecting t_m, the next trial value for T_m, is the same as that given for product 1. The appropriate relationships are obtained by changing the subscript **1** to m in (367) through (373).

This step of stage m is entered either from step 8 of stage 1, if $m = 2$ or step 11 of stage $m - 1$, or from step 11 of stage m. When step 9 is entered from either step 8 or stage 1 or step 11 of stage $m - 1$ the search regions for T_m are re-initialized as shown by (370). This is necessary because these regions for stage m are functions of the partial schedule $\Omega_{m-1} = \left\{ T_i, d_i \;\middle|\; i = 1, 2, \cdots, m - 1 \right\}$. Stage m is entered from stage $m - 1$ only after a new partial schedule Ω_{m-1} has been generated. Thus, it is necessary to restart the search for an optimal T_m at the single product optimal cycle time.

When this step is entered from step 11 of stage m all possible values of the mth delay time d_m, given Ω_{m-1} and the current trial cycle time t_m, have been eliminated. In this case, a new trial t_m is selected in a manner similar to that shown by (371) through (373) for selecting a new t_1. If the updated search regions for T_m are both null then the algorithm returns to step 11 of stage $m - 1,\ d_{m-1}$, or if $m = 2$, to step 7 of stage 1 to select a new t_1. Otherwise, the algorithm goes to step 10 of stage m to update the cycle time bounds for the unscheduled products $m+1, \cdots, n$.

Step 10 – Update Cycle Time Bounds

This step is entered only from step 9 of stage m. This step is skipped if $m = n$. If setting T_m to its new trial value increases the cost lower bound given by (360) for one or more of the unscheduled products $m+1, \cdots, n$, then compute new cycle time bounds for each such product.

If any of the updated bounds are such that $T_{lk} > T_{uk}$ for any $k = m+1, \cdots, n$ then return to either step 11 of stage $m - 1$ to select a new delay time for product $m - 1$ or, if $m = 2$ to step 7 of stage 1 to select a new cycle time t_1. Otherwise, go to step 11 of stage m to select a delay time for the ***mth*** product.

Step 11 – Select d_m

This step is entered from step 10 of stage m, from step 9 of stage $m + 1$, or step 13.c of stage n. When this step is entered from step 10 of stage m a new trial cycle time t_m has just been selected in step 9. Hence, no previous value of delay time d_m has been selected for this t_m. It is easy to show that d_m must satisfy

$$q_1 \leq d_m \leq t_m - q_m. \tag{374}$$

Thus, with each new trial t_m the algorithm initializes d_m by setting it to q_1 and then searches upward to find the smallest d_m that satisfies the necessary and sufficient conditions of theorem 1, given in chapter 3, with respect to each (T_k, d_k) contained in the partial schedule Ω_{m-1}. If there are no feasible d_m given the present trial cycle time t_m and the partial schedule Ω_{m-1} the algorithm returns to step 9 of stage m to select a new t_m. Otherwise, go to step 9 of stage $m + 1$ or to step 12 if $m = n$.

If the algorithm returns to this step from either step 9 of stage $m + 1$ or step 13.c of stage n the present trial delay time d_m can be eliminated because:

a. All feasible full schedules Ω_n that contain the current partial schedule Ω_m have been generated,

b. The partial schedule Ω_m is not feasible, or

c. The partial schedule is non-optimal.

After eliminating the previous d_m the algorithm starts at $d_m + 1$, where d_m is the previous trial delay time, and searches upward to find the smallest d_m that satisfies theorem 1. If no feasible delay time d_m is found the algorithm returns to step 9 of stage m to select a new t_m. Otherwise, the algorithm goes to step 9 of stage $m + 1$ if $m < n$ or to step 12 if $m = n$.

Step 12 – Update the Best Schedule and Cost Upper Bound

The algorithm enters this step only from step 11 of stage $\boldsymbol{n}$. It does so only after a feasible cycle time and delay time are found for product $\boldsymbol{n}$ in steps 9 and 11. These steps assure one that the first feasible cycle time t_n found in stage $\boldsymbol{n}$ will be an optimum T_n given the partial schedule Ω_{n-1}. Thus, there is no need to examine other possible values for T_n until Ω_{n-1} is changed.

The cycle time bounds restrict the search region of the algorithm to those schedules with costs that are strictly less than the current upper bound. Therefore, each time the algorithm enters step 12 an improved schedule Ω_n has been generated in the previous steps. This Ω_n is saved as the current best schedule and its cost is saved as the current cost upper bound Z_u. A small positive ϵ is subtracted from Z_u to assure that the algorithm generates a strictly monotonically decreasing series of upper bounds. Then the algorithm proceeds to step 13.

Step 13 – Restarting the Algorithm

A. Use the revised upper bound on cost and the lower bounds (360) from stage 0 to determine revised cycle time bounds for product 1. These new cycle time bounds lead to revised search zones for product 1. If both the lower and the upper search zones are now null; that is, if $L_1 < T_{l1}$ and $J_1 > T_{u1}$, then terminate the algorithm because all potentially optimal values of T_1 have been eliminated; otherwise, go to 13.B.

B. Use the revised upper bound $\boldsymbol{Z_u}$ to update the cycle time bounds computed in step 8. As before, if there are revised bounds such that $\boldsymbol{T_{lk}} > \boldsymbol{T_{uk}}$ then terminate the algorithm. Otherwise, if both the revised lower and upper search zones of product 2 are now null restart the algorithm by returning to step 7 to select a new value for $\boldsymbol{T_1}$. Otherwise, go to 13.C.

C. Beginning with $\boldsymbol{m = 2}$ use the revised upper bound $\boldsymbol{Z_u}$ to update the cycle time bounds for products $\boldsymbol{m+1, \cdots, n}$ computed in step 10 of stage $\boldsymbol{m}$. If any of the revised bounds are such that $\boldsymbol{T_{lk}} > \boldsymbol{T_{uk}}$ for any $\boldsymbol{k = m+1, \cdots, n}$ then restart the algorithm by returning to either step 11 of stage $\boldsymbol{m - 1}$ to select a new delay time, $\boldsymbol{d_{m-1}}$, for product $\boldsymbol{m - 1}$, or, if $\boldsymbol{m = 2}$, to step 7 of stage 1 to select a new cycle time $\boldsymbol{t_1}$.

If $\boldsymbol{T_{lk}} \leq \boldsymbol{T_{uk}}$ for all $\boldsymbol{k = m+1, \cdots, n}$ and if both the revised lower and upper search zones for product $\boldsymbol{m+1}$ are null then restart the algorithm by returning to step 11 of stage $\boldsymbol{m}$ to select the next value of $\boldsymbol{d_m}$. Otherwise, repeat step 13.C until the algorithm is restarted. As observed as the revised lower and upper search zones for product $\boldsymbol{n}$ with respect to the partial schedule $\boldsymbol{\Omega_{n-1}}$ are null. Hence, the algorithm will restart in step 11 of stage $\boldsymbol{n - 1}$ if not before.

IMPROVEMENTS TO THE ALGORITHM

Purpose

The basic algorithm will solve the n-model GCM; however, the results of the previous chapters can be used to improve the efficiency of the algorithm. These improvements are of two basic forms: those that reduce the search region and those that reduce the search effort. The first three improvements described below reduce the search region while the last one reduces the effort required to search the remaining region for a feasible schedule.

Improvement 1: Two-Products Compatibility

Stage 1

As shown by lemma 6 in chapter 6 all pairs of cycle times in a feasible schedule must satisfy (126) of theorem 3 in chapter 5. This result provides a powerful means for improving the efficiency of the basic algorithm.

To explain the advantages of this improvement and how it is implemented suppose that t_1 is a trial value of the cycle time of the first product. Further, suppose that T^*_{sw} is the single-product optimal product w. Finally, suppose that t_1 and T^*_{sw} do not satisfy (126) of theorem 3. Then, as shown by lemma 6 there are no feasible full schedules where $T_1 = t_1$ and $T_w = T^*_{sw}$. The basic algorithm will discover this incompatibility at stage w after testing all possible values of d_w, the delay time for product w. However, lemma 6 provides a method for eliminating such values of T_w at stage 1. This early recognition of incompatible cycle times for product w will, in most cases, greatly improve the efficiency of the algorithm.

The mechanics of implementing improvement 1 is to insert step 7.B into the basic algorithm. After each new trial cycle time t_1 is selected, step 7.B conducts two searches over the search region of each product w for all $w = 2, \cdots, n$. The revised algorithm begins its first search with T^*_{sw} and, if necessary, moves upward checking each value of T_w until it finds the minimum T_w in the upper search zone that satisfies (126). The algorithm sets J'_w, the lower limit on the upper search zone of T_w, to this minimum T_w. The second search of step 7.B also begins at T^*_{sw}, but moves downward until it finds the maximum T_w that satisfies (126). The algorithm sets L^1_w, the upper limit on the lower search zone of T_w, to this maximum T_w.

If after these searches both the lower and upper search zones of any product w, $w \neq 1$, are null, then, by lemma 6, t_1 is not a feasible value of T_1. In this case the algorithm returns to step 7 to eliminate the present trial value of T_1 and to select another. Step 7 and 7.B are repeated until a trial value t_1 is found that is compatible with at least one t_w for all $w = 2, \cdots, n$. The procedure will find two compatible t_w for all $w = 2, \cdots, n$ unless the searches eliminate all values of T_w in one of its search zones but finds a compatible value in the other zone. If the iterative application of steps 7 and 7.B does not find a t_1 that is compatible with at least one T_w for all $w = 2, \cdots, n$ then all feasible values of T_1 will have been eliminated. When this occurs the algorithm is terminated; otherwise, the algorithm proceeds to step 8.

Improvement 1 also requires a revision to step 8. If any values of T_w are eliminated by step 7.B then the lower bounds on costs given by (360) may need to be increased. This is because

$$Z'_{lw} = min\left[Z_w\left(L'_w\right),\ Z_w\left(J'_w\right)\right] \tag{375}$$

is a lower bound on the cost of product w given that $T_1 = t_1$. These lower bounds on costs may lead to greater lower bounds than those given by (360). Thus, step 8 is revised to appropriately use the bounds given by (375) while determining the cycle time bounds on T_w for each $w = 2, \cdots, n.$

If these revised cycle time bounds for any product w are less than the previous bounds, then at least one of the revised search zones for product $w = 2, \cdots, n$ will be smaller than the previous search zone. If this eliminates both zones for any product $w = 2, \cdots, n,$ then the present trial value t_1 can be eliminated. In this case the algorithm returns to step 7 to select another trial t_1.

__Stage m,__ $2 \leq m \leq n$

Improvement 1 is also applied to stage m of the algorithm by adding a step 9.B that determines if the trial cycle time t_m is potentially feasible. The procedure for stage m differs from that of stage 1 in two ways.

The first difference is that the search procedure of step 9.B has a "look-forward" phase and a "look-backward" phase. The look-forward phase is the same as the search procedure used in step 7.B except the search is limited to the unscheduled products where $w = m + 1, \cdots, n$. Of course, there is no look-forward phase to stage n.

The look-backward phase determines if t_m and each t_w, $w = 1, \cdots, m - 1$, of the scheduled products are compatible, that is, if these cycle times satisfy (126). If t_m is not compatible with any t_w of a scheduled product then that trial t_m is eliminated and the algorithm returns to step 9 to select another trial value. The other difference in the procedure of stage m versus that of stage 1 is the starting points of the searches during the look-ahead phase. In stage 1 for each new t_1 the starting point for the search over the values of T_w are initialized to T^*_{sw}. However, once an exclusion zone, $L^{m-1}_w < t_w < J^{m-1}_w$, is determined for T_w at stage $m - 1$ there is no need to consider any T_w in that exclusion zone unless the partial schedule Ω_{m-1} is changed. Thus, the two searches over T_w at stage m begin at J^{m-1}_w and L^{m-1}_w for each new trial t_m.

As is done in stage 1, step 10 must be revised to take advantage of the cost lower bounds

$$Z^m_{tw} = min\left[Z_w\left(L^m_w\right),\ Z_w\left(J^m_w\right)\right]. \tag{376}$$

The revised cycle time bounds from step 10 may lead to revised lower and upper search zones for one or more products $w = m + 1, \cdots, n$. If there is an unscheduled product where one, and only one, of its revised search zones is null but the previous version of that search zone was not null, then the cost lower bounds given by (376) must be computed again. If the revised cost bounds for one or more of the unscheduled products is greater than the previous cost bound then these revised cost bounds are used to compute new cycle time bounds for each of the unscheduled products. This iterative procedure continues until there is no increase in the cost lower bounds given by (376).

If there is at lease one of the unscheduled products where both of its search zones are null, then the trial cycle time t_m can be eliminated. In this case the algorithm returns to step 9 to select another trial cycle time. Otherwise, the revised algorithm goes to step 11 to select a delay time for product m.

Improvements 2: Three-Products Compatibility

Lemmas 8 and 9 in chapter 7 provide two sets of necessary conditions that any feasible full schedule must satisfy. Both sets of these conditions are based on the results of theorem 6 on conditions for the three-products GCM. These conditions require more effort to apply than the two-products conditions used in improvement 1. Thus, the algorithm test whether the three-products conditions are satisfied only after the actions of improvement 1 have been completed.

The mechanics of improvement 2 is to insert a step 10.B into stage m of the algorithm to implement the three-products tests. At this point the algorithm has generated the partial schedule $\Omega_{m-1} = \left(t_i,\ d_i \,\middle|\, i = 1,\ \cdots,\ m-1 \right)$ and has selected a trail t_m. The algorithm has also identified, for each unscheduled product $i = m+1,\ \cdots,\ n,$ a lower search zone $T_{li}^{m} \leq T_i \leq L_i^{m}$ or has shown that this lower search zone is null. A similar statement is true for the upper search zone $J_i^{m} \leq T_i \leq T_{ui}^{m}$. Let

$$\theta_l^m = \left\{ i \middle| L_i^m \geq T_{li}^m \right\} \ and \ \theta_u^m \left\{ i \middle| J_i^m \leq T_{ui}^m \right\}.$$

Observe that d_m has not been selected at this point in the algorithm. The three-products tests are performed before selecting d_m to reduce the number of delay times that must be generated by the algorithm.

The following test are performed by step 10.B:

1. If $m \geq 3$ then this step tests whether $\left(t_m,\ t_i,\ t_j,\ d_i,\ d_j\right)$, for each $i,\ j = 1,\ 2,\ \cdots,\ m-1,\ i \neq j,$ satisfy the conditions of lemma 9. Observe that both d_i and d_j have been assigned values so lemma 9 is the appropriate test. If there is a pair of products *i. j* such that the conditions of lemma 9 are not satisfied then t_m is eliminated so the algorithm returns to step 9 to select another trial t_m.

2. If $m \geq 2$ this step tests whether $\left(t_m,\ t_i,\ L_j^m\right)$, for each $i = 1,\ 2,\ \cdots,\ m-1$ and each $j \in \theta_l^m$, satisfy the conditions of lemma 8. If there is a pair of products *i, j* such that $\left(t_m,\ t_i,\ L_j^m\right)$ do not satisfy lemma 8 then the current value of L_j^m is eliminated.

3. If $m \geq 2$ this step tests whether $\left(t_m,\ t_i,\ J_i^m\right)$, for each $i = 1,\ 2,\ \cdots,\ m-1$ and each $j \in \theta_u^m$, satisfy the conditions of lemma 8.

If there is a pair of products *i, j* such that $\left(t_m,\ t_i,\ J_j^m\right)$ do not satisfy lemma 8 then the current value of J_j^m is eliminated.

4. If $m \geq 3$ this step tests whether $\left(t_i,\ t_j,\ L_k^m\right)$ for each pair $i, j = 1, 2, \cdots, m-1,\ i \neq j$, and each $k \in \theta_l^m$ satisfies the conditions of lemma 9. If there is a pair of products *i, j* such that the conditions of lemma 9 are not satisfied then L_k^m is eliminated.

5. If $m \geq 3$ this step tests whether $\left(t_i,\ t_j,\ J_k^m\right)$, for each pair $i, j = 1, 2, \cdots, m-1,\ i \neq j$, and each $k \in \theta_u^m$, satisfy the conditions of lemma 9. If there is a pair of products *i, j* such that the conditions of lemma 9 are not satisfied then J_k^m is eliminated.

After completing these tests if none of the L_i^m, J_i^m have been eliminated the algorithm goes to step 11. On the other hand, if there is an unscheduled product *w* where both the lower and upper search zones are now null then t_m is eliminated and the algorithm returns to step 9 to select another trial t_m. Otherwise, the algorithm returns to step 9.B to conduct the two-products tests on the revised limits L_i^m, J_i^m.

Improvement 3: Four-Products Compatibility

Lemmas 10 and 11 in chapter 8 also provides two sets of necessary conditions that any feasible full schedule must satisfy. Both sets of these conditions are based on the necessary conditions of theorem 7 on conditions for the four-products GCM. These four-products conditions require even more effort to apply than the three-products conditions of improvement 2. Thus, the algorithm test whether the four-products conditions are satisfied only after the algorithm leaves step 10.B for step 11 as described in improvement 2.

The mechanics of improvement 3 parallel those of improvement 2. The tests and resulting actions required to implement the four-products condition are implemented by inserting a step 10.C between step 10.B and step 11. Step 10.C is a straightforward adaptation of step 10.B.

Improvement 4: Skipping with Euclid

Improvement 4 is a marvelous method for improving the efficiency of the searches for compatible cycle times in steps 7, 7.B, 9 and 9.B of improvement 1. Rather than check each t_i of an unscheduled product i for compatibility with t_k that was specified at an earlier stage m it is possible to skip many values of t_i. This skipping method is based on Euclid's algorithm for finding the greatest common divisor of two integers.

To illustrate, suppose that a trial $t_k, k = 1, \cdots, m,$ has been selected at stage m and that it is being tested against $t_i, i < m$ to determine if these two cycle times satisfy the necessary condition (126). Applying this test requires determining the greatest common divisor of t_k and t_i. The first iteration of Euclid's algorithm is an application of the division algorithm that yields

$$t_i = \alpha_i t_k + \gamma_i \tag{377}$$

when $t_i \geq t_k$ and

$$t_k = \alpha_k t_i + \gamma_k \tag{378}$$

when $t_k \geq t_i$ where $\alpha_i, \alpha_k, \gamma_i, \gamma_k \geq 0, \gamma_i < t_i$, and $\gamma_k < t_k$.

If $\gamma_i = 0$ when $t_i \geq t_k$ then $gcd\left(t_k, t_i\right) = t_k$; thus, to satisfy (126) $t_k \geq q_i + q_k$. Clearly, to be compatible, t_i and t_k must satisfy

$$t_k \geq q_k + s_i + \left[\rho_i t_i\right]^+ \tag{379}$$

and

$$t_i \geq q_k + s_i + \left[\rho_i t_i\right]^+ \tag{380}$$

where t_k is a previously specified value. Bounds (379) and (380) must be satisfied when the cycle time bounds T_{li}^m and T_{ui}^m, $i = m+1, \cdots, n$ are set at stage $m = 1, 2, \cdots, n-1$. Henceforth, it is assumed that steps 7, 7.B, 9 and 9.B are modified to enforce these bounds as each trial cycle times t_m is selected and when the limits L_i^m and J_i^m are determined. Thus, for the remainder of the discussion of improvement 4, it is assumed that if $\gamma_i = 0$ or $\gamma_k = 0$ then t_i, t_k are compatible.

Now suppose that γ_i is such that

$$0 < \gamma_i < q_i + q_k. \tag{381}$$

Let γ_i^j be the jth residual of Euclid's algorithm when finding $gcd(T_i, T_k)$. A well known result from linear algorithm states that if any of the residuals, γ_i^j, of Euclid's algorithm are greater than zero then

$$a_{ik} = gcd(T_i, T_k) \leq \gamma_i^j. \tag{382}$$

Hence, if (381) is true then t_i is not compatible with t_k so t_i can be eliminated. This means that the trial cycle time t_i can be eliminated as soon as any residual of Euclid's algorithm is found that is less than $q_i + q_k$.

The last observation improves the efficiency of the algorithm because, for many incompatible t_i, it will not be necessary to preform all steps of Euclid's algorithm. However, there is an even more powerful ramification of (381). Suppose that $t_i \geq t_k$ and the algorithm is searching upward in the upper search zone of product i when a residual γ_i satisfying (381) is found. (Note that $\gamma_i = \gamma_i^1$. The superscript 1 is dropped to simplify the notation.) Then from (377) it is clear that all values of T_i from t_i to $(t_i + \Delta)$ where

$$\Delta = (q_i + q_k - \gamma_i - 1) \tag{383}$$

can be skipped. That is, all T_i in the range $[t_i, t_i + \Delta]$ are incompatible with t_k. As will be explained next, Euclid's algorithm yields four different "skipping" values.

The searches of step 7, 7.B, 9 and 9.B all presume that cycle times $t_1, t_2, \cdots, t_m, m = 1, 2, \cdots, n-1,$ have been specified. The object of each search is to find a value of T_i, $i > m$, that is compatible with all t_k, $k = 1, \cdots, m$. Improvement 4 divides each search into two phases.

The first phase is an iterative "skipping" procedure that starts at $k = 1$ and at some initial value of $T_i = t_i$. Then the first iteration of Euclid's algorithm yields the residual γ_j, $j = i$ *or* k. If $\gamma_j \geq q_i + q_k$ then the algorithm repeats this step for product $k + 1$ if $k < m$ and for product 1 if $k = m$. The algorithm continues this cyclical procedure until m consecutive residuals γ_j satisfy $\gamma_j \geq q_i + q_k$. It then goes to phase 2 of the search.

On the other hand, if $0 < \gamma_j < q_i + q_k$ it is possible to skip one or more values of T_i by adding a Δ to the current trial value t_i. The size of this Δ depends on whether the search is being made downward in the lower search zone or upward in the upper search zone. It also depends on whether $t_i \geq t_k$ or vice versa. The table in figure 20 gives the proper Δ for each of the four combinations of these conditions.

	Lower Search Zone Searching Downward	Upper Search Zone Searching Upward
$t_i > t_k$	$-\gamma_i$	$\left[\left(q_k + s_i - \gamma_i + \rho_i t_i\right) / \left(1 - \rho_i\right)\right]^+$
$t_i < t_k$	$-\left[\gamma_k / \alpha_k\right]^+$	$\left[\gamma_k / \alpha_k\right]^+$

Figure 21 – Skip Values if $\gamma_j < q_i + q_k$

Of course, if this procedure causes t_i to skip by T_{li} or T_{ui} then the lower or upper search zone, respectively, is null. As stated above the skipping procedure continues until it finds m consecutive residuals $\gamma_j \geq q_i + q_k,\ j = i \text{ or } j = k.$ Once this occurs the algorithm goes to phase 2 of improvement 4.

Phase 2 is a straightforward application of Euclid's algorithm. This phase uses Euclid's algorithm to find the greatest common of t_i and t_k for $k = 1, 2, \cdots, m.$ If any positive residual for any product k is found that is less than $q_i + q_k$ then t_i is eliminated by setting the next trial value of $T_i = t_i + 1.$ The algorithm then returns to phase 1 of improvement 4 with the new starting value for $T_i.$ If the greatest common divisor is found for all pairs $t_i,\ t_k,\ k = 1, 2, \cdots, m$ then all $t_i,\ t_k$ satisfy (126). In this case the algorithm returns to step 7.B or 9.B.

CHAPTER 10

NUMERICAL RESULTS

The basic GCM algorithm in chapter 9 was coded in a compiled version of the Basic programming language. To simplify the program the algorithm to solve Bomberger's BPM given in chapter 4 was not coded. Likewise, improvements 2 and 3 to the GCM algorithm were not included in the code.

This computer program was used to solve three example problems proposed by Bomberger [4]. The model parameters for these three problems are given in Figures 22, 23 and 24. Bomberger constructed these data to have a facility utilization of 22 percent in problem 1, 66 percent in problem 2, and 88 percent in problem 3.

To replace the upper bounds provided by the BPM algorithm, Bomberger's published results were used as initial upper bounds for the problems. The two-products BPM lower bounds were replaced with the sum of the single-product EMQ optimals. To measure the usefulness of the BPM algorithm the problems were also solved with the initial upper bounds set to Hanssman's [42] CCM results for each problem.

To simplify the execution of his algorithm Bomberger used an eight hour day as the basic time unit. However, the setup times of all three problems are given in hours. Hence, it seems that a time unit if an hour or a fraction of an hour is more natural. A time unit of one hour was

Part No.	Setup Cost	Inventory Cost	Inventory Cost per Hour	Hourly Production Rate	Hourly Demand Rate	Setup Time
	$ per set up	$ per part per hr. times 10,000	$ per hr. times 10,000	Parts per hr.	Parts per hr.	Hours per setup
i	c_i	h_i	H_i	p_i	r_i	s_i
1	15	0.00339	0.02109	3,750.00	12.50	1
2	20	0.06120	0.37771	1,000.00	12.50	1
3	30	0.06641	0.81260	1,187.50	25.00	2
4	10	0.05208	1.23264	937.50	50.00	1
5	110	1.45052	1.79502	250.00	2.50	4
6	50	0.13932	0.17357	750.00	2.50	2
7	310	0.78125	0.29224	300.00	0.75	8
8	130	3.07292	15.25747	162.50	10.63	4
9	200	0.46875	2.38440	250.00	10.63	6
10	5	0.02083	0.12934	1,875.00	12.50	1

Figure 22 – Model Parameters of Bomberger's Problem 1

Part No.	Setup Cost	Inventory Cost	Inventory Cost per Hour	Hourly Production Rate	Hourly Demand Rate	Setup Time
	$ per set up	$ per part per hr. times 10,000	$ per hr. times 10,000	Parts per hr.	Parts per hr.	Hours per setup
i	c_i	h_i	H_i	p_i	r_i	s_i
1	15	0.00339	0.06284	3,750.00	37.50	1
2	20	0.06120	1.10443	1,000.00	37.50	1
3	30	0.06641	2.33296	1,187.50	75.00	2
4	10	0.05208	3.28125	937.50	150.00	1
5	110	1.45052	5.27627	250.00	7.50	4
6	50	0.13932	0.51724	750.00	7.50	2
7	310	0.78125	0.87231	300.00	2.25	8
8	130	3.07292	39.36805	162.50	31.88	4
9	200	0.46875	6.51819	250.00	31.88	6
10	5	0.02083	0.38281	1,875.00	37.50	1

Figure 23 – Model Parameters of Bomberger's Problem 2

Part No.	Setup Cost	Inventory Cost	Inventory Cost per Hour	Hourly Production Rate	Hourly Demand Rate	Setup Time
	$ per set up	$ per part per hr. times 10,000	$ per hr. times 10,000	Parts per hr.	Parts per hr.	Hours per setup
i	c_i	h_i	H_i	p_i	r_i	s_i
1	15	0.00000	0.00001	3,750.00	50.00	1
2	20	0.00001	0.00015	1,000.00	50.00	1
3	30	0.00001	0.00030	1,187.50	100.00	2
4	10	0.00001	0.00041	937.50	200.00	1
5	110	0.00015	0.00070	250.00	10.00	4
6	50	0.00001	0.00007	750.00	10.00	2
7	310	0.00008	0.00012	300.00	3.00	8
8	130	0.00031	0.00482	162.50	42.50	4
9	200	0.00005	0.00083	250.00	42.50	6
10	5	0.00000	0.00005	1,875.00	50.00	1

Figure 24 – Model Parameters of Bomberger's Problem 3

elected for all problems and Bomberger's model parameters were converted to the hourly values shown in Figures 22, 23 and 24.

Reducing the time unit will usually reduce the optimal cost found by the GCM algorithm. These improved results are expected because the smaller time unit gives the GCM more flexibility. However, it is also expected that the computer effort required to solve each problem will be significantly greater when time is measured on an hourly basis rather than a daily basis. Where appropriate, Bomberger's and Hanssman's results are expressed in terms of an hourly time unit in the figures below.

As explained in chapter 9 an acceptable small value must be subtracted from each new upper bound found by the algorithm to assure these upper bounds are a strictly monotonically decreasing series. This is implemented in the computer program by multiplying each new upper bound by the factor $(1-\Delta)$. A Δ of 0.001 was used when solving all three problems.

The optimal costs for each problem obtained from the computer program is shown in Figure 25. These results are compared with Hanssman's and Bomberger's results. As expected, the difference between Bomberger's results and these of the GCM algorithm widens as the facility utilization increases. The GCM offers significantly better schedules than the BPM for problems 2 and 3.

Problem Number	Utilization	Hanssman's CCM	Bomberger's BPM	Results from Algorithm	Improvement over Bomberger
		($ / hour)*	($ / hour)*	($ / hour)*	(%)
1	0.22	2.8128	2.1180	2.1103	0.4
2	0.66	4.5848	3.7263	3.4994	6.1
3	0.88	5.1203	4.5618	3.9395	13.6

*There are eight hours per day.

Figure 25 – Comparison of Average Hourly Cost

The cycle times and delay times of the optimal schedules found for each problem are shown in Figures 26, 27 and 28. The GCM cycle times are compared with those obtained by Hanssman's and Bomberger's.

The times to solve each of Bomberger's problems on a personal computer is shown by Figure 29. Each problem was solved twice: first with Hanssman's and then with Bomberger's optimal cost as the initial upper bound. Figure 29 provides a measure of the benefits offered by the BPM algorithm of chapter 4.

Another, perhaps better, measure of the computational effort required by each case of each problem is shown in figures 30, 31 and 32. As explained in chapter 9, a stage ***m***, except ***m* = *1*** or ***m* = *n***, can be entered only from the previous stage ***m* - *1***, the next stage ***m* + *1***, or rarely, from stage ***n*** after a new upper bound is found. Figures 30, 31 and 32 show the number of times each stage is entered from the immediately previous stage and from the

Product	Integer Lower Bound	Hanssman's Integer Solution	Bomberger Results	Results from Algorithm	
				Cycle Times	Delay Times
	(hours)	(hours)	(hours)	(hours)	(hours)
1	2,667	626	2,560	2,619	709
2	728	626	640	776	273
3	608	626	640	582	403
4	285	626	320	291	24
5	783	626	800	776	177
6	1,697	626	1,760	1,746	1,000
7	3,257	626	3,200	3,298	629
8	292	626	320	291	0
9	916	626	800	873	359
10	622	626	640	485	189

Figure 26 – Comparison of Cycle Times for Problem 1

Product	Integer Lower Bound	Hanssman's Integer Solution	Bomberger Results	Results from Algorithm	
				Cycle Times	Delay Times
	(hours)	(hours)	(hours)	(hours)	(hours)
1	1,545	384	1,680	1,086	964
2	426	384	280	1,629	403
3	359	384	280	362	328
4	175	384	280	362	41
5	457	384	280	181	337
6	983	384	280	543	434
7	1,885	384	1,400	1,810	456
8	182	384	280	181	0
9	554	384	560	543	117
10	361	384	280	543	509

Figure 27– Comparison of Cycle Times for Problem 2

Product	Integer Lower Bound	Hanssman's Integer Solution	Bomberger Results	Results from Algorithm	
				Cycle Times	Delay Times
	(hours)	(hours)	(hours)	(hours)	(hours)
1	1,545	344	320	1,494	48
2	426	344	320	332	289
3	359	344	320	332	123
4	175	344	320	166	86
5	457	344	320	332	104
6	983	344	320	747	153
7	1,885	344	320	1,660	971
8	182	344	960	166	0
9	554	344	320	498	381
10	361	344	320	332	225

Figure 28– Comparison of Cycle Times for Problem 3

Problem	Initial Upper Bound		
	Hanssman (seconds)	Bomberger (seconds)	Improvement Ratio
1	11	6	1.8
2	1,996	35	57.0
3	1,451	297	4.9

Figure 29 – Computer Time Required to Find Optimal

higher number stages. The initial upper bound did not effect the number of times each stage is executed to solve problem 1. This result is probably due to the low facility utilization of problem 1. However, as shown by figures 31 and 32, the initial upper bound has a huge effect on the number of stage executions required to solve Bomberger's problem 2 and 3 where there is a moderate to high facility utilization.

Figures 30, 31 and 32 also show the assignment of Bomberger's products to stages of the GCM algorithm. These assignments were determined off-line and included as input to the computer program. These assignments were not changed during execution of the GCM program.

		Initial Upper Bound			
		Bomberger		Hansmann	
Stage	Product	Entered From Previous Stage	Entered From Next Stage	Entered From Previous Stage	Entered From Next Stage
1	8	0	1	0	1
2	4	1	251	1	251
3	9	251	387	251	387
4	5	387	76	387	76
5	3	76	148	76	148
6	2	148	42	148	42
7	10	42	2	42	2
8	6	2	420	2	420
9	7	420	274	420	274
10	1	274	0	274	0

Figure 30 – Stage Executions for Problem 1

		Initial Upper Bound			
		Bomberger		Hansmann	
Stage	Product	Entered From Previous Stage	Entered From Next Stage	Entered From Previous Stage	Entered From Next Stage
1	8	0	19	0	19
2	9	19	198	19	198
3	5	198	35	198	42
4	4	35	26	42	26
5	3	26	1	26	42
6	2	1	300	42	17,261
7	7	300	216	17,261	1,092,0488
8	6	216	33	1,092,0488	635,523
9	10	33	75	635,523	86
10	1	75	0	86	0

Figure 31 – Stage Executions for Problem 2

		Initial Upper Bound			
		Bomberger		Hansmann	
Stage	Product	Entered From Previous Stage	Entered From Next Stage	Entered From Previous Stage	Entered From Next Stage
1	8	0	11	0	12
2	9	11	249	12	84
3	5	249	23,097	84	7,879
4	4	23,097	108,555	7,879	23,102
5	3	108,555	35,482	23,102	17,194
6	2	35,482	23,973	17,194	20,089
7	7	23,973	7,969	20,089	331,622
8	6	7,969	294,369	331,622	387,581
9	10	294,369	2,275	387,581	322
10	1	2,275	0	322	0

Figure 32 – Stage Executions for Problem 3

REFERENCES

REFERENCES

1. Axsater, S., (Feb, 1987), Extension of the extended basic period approach for economic lot scheduling problems, *Journal Optimization Theory and Applications*, Vol. 52, No. 2, pp. 179-189.

2. Baker, K. R., (May, 1970), On Madigan's approach to deterministic multi-product production and inventory problem, *Management Science*, Vol. 16, No. 9, pp. 636-638.

3. Boctor, F. F., (Jul, 1982), The two-product, single-machine, static demand, infinite horizon lot scheduling problem, *Management Science*, Vol. 28, No. 7, pp. 798-807.

4. Bomberger, E. E., (Jul, 1966), A dynamic programming approach to a lot size scheduling problem, *Management Science*, Vol. 12, No. 11, pp. 778-784.

5. Bourland, K. E., and Yano, C. A., (Dec, 1994), The strategic use of slack capacity in the economic lot scheduling problem with random demand, *Management Science*, Vol. 40, No. 12, pp. 1690-1704.

6. Bourland, K. E., and Yano, C. A., (Feb, 1997), A comparison of solutiion approaches for the fixed-sequence economic lot scheduling problem, *IIE Transactions*, Vol. 29, No. 2, pp. 103-108.

7. Carreno, J. J., (Mar, 1990), Economic lot scheduling for multiple products on parallel identical processors, *Management Science*, Vol. 36, No. 3, pp. 348-358.

8. Cook, W. D., Saipe, A. L., and Seiford, L. M., (1983), Production runs for multiple products: The full-capacity heuristic, *Journal of the Operational Research Society*, Vol. 31, No. 5, pp. 405-412.

9. Crowston, N. B., Wagner, M., and Williams, J. F., (Jan, 1973), Economic lot size determination in multi-stage assembly systems, *Management Science*, Vol. 19, No. 5, pp. 517-527.

10. Davis, S. G., (Aug, 1990), Scheduling economic lot size (ELS) production runs, *Management Science*, Vol. 36, No. 8, pp. 985-998.

11. Davis, S. G., (1995), An improved algorithm for solving the economic lot size problem (ELSP), *International Journal of Production Research*, Vol. 33, No. 4, pp. 1007-1026.

12. Delporte, C. M., and Thomas, L. J., (Jun, 1977), Lot sizing and sequencing for N products on one facility, *Management Science*, Vol. 23, No. 10, pp. 1070-1079

13. Ditt, S., and Kuhn, H., (Jun, 1997), An improved algorithm for solving the economic lot size problem (ELSP): A note, *International Journal of Production Research*, Vol. 35, No. 6, pp. 1785-1787.

14. Dobson, G., (Sep-Oct, 1987), The economic lot scheduling problem: achieving feasibility using time-varying lot sizes, *Operations Research*, Vol. 35, No. 5, pp. 764-771.

15. Doll, C. L., and Whybark, D. C., (Sep, 1973), An iterative procedure for the single-machine multi-product lot scheduling problem, *Management Science*, Vol. 20, No. 1, pp. 50-55.

16. Eilon, S., (1959), Economic batch-size determination for multi-product scheduling, *Operations Research Quarterly*, Vol. 10, No. 4, pp. 217-227.

17. El-Najdawi, M. K., (Apr, 1992), A compact heuristic for lot-size scheduling in multi-stage, multi-product production processes, *International Journal of Production Economics*, Vol. 27, No. 1, pp. 29-41.

18. El-Najdawi, M. K., and Kleindorfer, P. R., (Jul, 1993), Common cycle lot-size scheduling for multi-product, multi-stage production, *Management Science*, Vol. 39, No. 7, pp. 872-885.

19. El-Najdawi, M. K., (Jun, 1994), A job-splitting heuristic for lot-size scheduling in multi-stage, multi-product production processes, *European Journal of Operational Research*, Vol. 75, No. 2, pp. 365-377.

20. Elmaghraby, S. E., (Feb, 1978), The economic lot scheduling problem (ELSP): review and extensions, *Management Science*, Vol. 24, No. 6, pp. 587-598.

21. Elmaghraby, S., and Eliman, A., (Mar, 1980), Knapsack based approaches to the makespan problem on multiple processors, *AIEE Transactions*, Vol. 12, No. 1, pp. 87-99.

22. Fujita, S., (Dec, 1978), The application of marginal analysis to the economic lot scheduling problem, *AIEE Transactions*, Vol. 10, No. 4, pp. 354-361.

23. Gallego, G., (Jun, 1990), An extension to the class of easy economic lot scheduling problems, *IIE Transactions*, Vol. 22, No. 2, pp. 189-190.

24. Gallego, G., and Moon, I., (1992), The effect of externalizing setups in the economic lot scheduling problem, *Operations Research*, Vol. 40, No. , pp. 614-619.

25. Gallego, G., and Roundy, R., (1992), The extended economic lot scheduling problem, *Naval Research Logistics Quarterly*, Vol. 39, No. , pp. 729-829.

26. Gallego, G., and Shaw, D. X., (Feb, 1997), Complexity of the ELSP with general cyclic schedules, *IIE Transactions*, Vol. 29, No. 2, pp. 109-113.

27. Galvin, T. M., (Jan-Mar, 1987), Economic lot scheduling problem with sequence-dependent setup costs, *Production and Inventory Management*, Vol. 28, No. 1, pp. 96-105.

28. Galvin, T. M., and Van Deusen, D. L., (Apr-Jun, 1988), Interactive economic lot scheduling: A successful implementation, *Production and Inventory Management*, Vol. 29, No. 2, pp. 1-5.

29. Geng, P. C., and Vickson, R. G., (1988), Two heuristics for the Economic Lot Scheduling Problem: An experimental study, *Naval Research Logistics Quarterly*, Vol. 35, No. , pp. 605-617.

30. Glass, C. A., (Oct, 1992), Feasibility of scheduling lot sizes of three products on one machine, *Management Science*, Vol. 38, No. 10, pp. 1482-1492.

31. Goyal, S. K., (1973), Scheduling a multi-product single-machine system, *Operations Research* Quarterly, Vol. 24, No. , pp. 261-266.

32. Goyal, S., (1975), Scheduling a multi-product single machine system - A new approach, *International Journal of Production Research*, Vol. 13, No. 5, pp. 487-493.

33. Goyal, S. K., (1984), Determination of economic production quantities for a two-product single machine system, *Operations Research Quarterly*, Vol. 22, No. , pp. 121-126.

34. Graves, S. C., (Mar, 1979), On the deterministic demand multi-product single machine lot scheduling problem, *Management Science*, Vol. 25, No. 3, pp. 276-280.

35. Graves, S. C., (Sep, 1980), The multi-product production cycling problem, *AIEE Transactions*, Vol. 12, No. 3, pp. 233-240.

36. Gray, J. R. (Oct. 1973), Economic Manufacturing Quantities for a Multi-product, Single Facility System, *Proceedings: Ninth Annual Meeting Southeastern Chapter, The Institute of Management Sciences*, Vol. 3, pp 285-297.

37. Gunasekaran, A., Goyal, S. K., Martikainen, T., and Yli, O.P., (Mar, 1993), Multi-level lot-sizing in a rayon yarn company: a case study, *European Journal of Operational Research*, Vol. 65, No. 2, pp. 159-174.

38. Haessler, R. W., (Dec, 1971), A Note on Scheduling a Multi-Product Single Machine System for an Infinite Planning Period, *Management Science*, Vol. 18, No. 4, Part 1 of 2, pp. B240-B241.

39. Haessler, R., and Hogue, S., (Apr, 1976), A note on the single-machine, multi-product lot scheduling problem, *Management Science*, Vol. 22, No. 8, pp. 909-912.

40. Haessler, R. W., (Dec, 1979), An improved extended basic period procedure for solving the economic lot scheduling problem, *AIEE Transactions*, Vol. 11, No. 4, pp. 336-340.

41. Haji, R., and Mansuri, M., (Mar-Apr, 1995), Optimum common cycle for scheduling a single-machine Multi-product system with a budgetary constraint, Production Planning and Control, Vol. 6, No. 2, pp. 151-156.

42. Hanssman, F., (1962), *Operations Research in Production and Inventory Control*, John Wiley and Sons Inc., New York, New York, pp. 155-160.

43. Hodgson, T. J., (Mar, 1970), Addendum to Stankard and Gupta's Note on Lot Size Scheduling, *Management Science*, Vol. 16, No. 7, pp. 514-517.

44. Hodgson, T. J., and Nuttle, H. L. W., (1986), A note on linear programming and the single machine lot size scheduling problem, *International Journal of Production Research*, Vol. 24, No. , pp. 939-943.

45. Hottenstein, editor, M. P., (1968), Models and Analysis for Production Management, International Textbook Company, Scranton, Pennsylvania, pp. 511-549.

46. Houshyar, A., (Jun, 1991), Optimal cycle time in a multi-product single machine with unit load, and storage space considerations, Computers in Industry, Vol. 16, No. , pp. 197-208.

47. Hsu, W. L., (Jan, 1983), On the general feasibility of scheduling lot sizes on several product on one machine, *Management Science*, Vol. 29, No. 1, pp. 93-105.

48. Hwang, H., and Moon, D. H., (1991), A production inventory model for producing two-products at a single facility with deteriorating raw material, Computers and Industrial Engineering, Vol. 20, No. 1, pp. 141-147.

49. Jones, P. C., and Inman, R. R., (Mar, 1989), When is the economic lot scheduling problem easy?, *IIE Transactions*, Vol. 21, No. 1, pp. 11-20.

50. Khouja, M., (Jan, 1997), The scheduling of economic lot sizes on volume flexible production systems, *International Journal of Production Economics*, Vol. 48, No. 1, pp. 73-86.

51. Kim, K. M., and Hwang, H., (1991), Integrated production inventory model with a powers-of-two restriction, Computers and Industrial Engineering, Vol. 20, No. 1, pp. 149-153.

52. Kim, S. L., Hayya, J. C., and Hong, J., (1995), Setup reduction and machine availability, Production Operations Management, Vol. 4, No. , pp. 76-90.

53. Koulamas, C., (Oct, 1995), Application of ELSP solution techniques to the deterministic robotic scheduling problem, *International Journal of Production Research*, Vol. 33, No. 10, pp. 2933-2944.

54. Krone, Jr., L. H., (Apr, 1964), A Note on Economic Lot Sizes for Multi-Purpose Equipment, *Management Science*, Vol. 10, No. 3, pp. 461-464.

55. Larraneta, J., and Onieva, L., (Apr, 1988), The economic lot scheduling problem: a simple approach, *Journal of the Operational Research Society*, Vol. 39, No. 4, pp. 373-379.

56. Lasdon, L., and Terjung, R., (1971), An efficient algorithm for multi-item scheduling, *Operations Research*, Vol. 19, No. , pp. 946-969.

57. Leachman, R. C., and Gascon, A., (Mar, 1988), A heuristic scheduling policy for multi-item, single-machine production systems with time-varying stochastic demands, *Management Science*, Vol. 34, No. 3, pp. 377-390.

58. Leachman, R. C., Xiong, Z. K., Gascon, A., and Park, K., (Sep, 1991), An improvement to the dynamic cycle lengths heuristic for scheduling the multi-item, single machine, *Management Science*, Vol. 37, No. 9, pp. 1201-1205.

59. Madigan, J. G., (Jul, 1968), Scheduling a Multi-Product Single Machine System for an Infinite Planning Period, *Management Science*, Vol. 14, No. 11, pp. 713-719.

60. Manne, A., (Jan, 1958), Programming of economic lot sizes, *Management Science*, Vol. 4, No. 2, pp. 115-135.

61. Matsuo, H., (Oct, 1990), Cyclic sequencing problems in the two-machine permutation flow shop: complexity, worst-case, and average-case analysis, Naval Research Logistics, Vol. 37, No. 5, pp. 679-694.

62. Matthews, J. P., (1988), The optimality of the zero-switch rule for a class of economic lot-scheduling problems, *Journal of the Operational Research Society*, Vol. 39, No. 12, pp. 1155-1161.

63. Maxwell, W. L., (Jun-Sep, 1964), The Scheduling of Economic Lot Sizes, *Naval Research Logistics Quarterly*, Vol. 11, No. , pp. 89-124.

64. Maxwell, W., and Singh, H., (Sep, 1983), The effect of restricting cycle times in the economic lot scheduling problem, *AIEE Transactions*, Vol. 15, No. 3, pp. 235-241.

65. Maxwell, W. L., and Singh, H., (May-Jun, 1986), Scheduling cyclic production on several identical machines, *Operations Research*, Vol. 34, No. 3, pp. 460-463.

66. Moon, I., Gallego, G., and Simchi-Levi, D., (Dec, 1991), Controllable production rates in a family production context, *International Journal of Production Research*, Vol. 29, No. 12, pp. 2459-2470.

67. Narro Lopez, M. A., and Kingsman, B. G., (Oct, 1991), The economic lot scheduling problem, *International Journal of Production Economics*, Vol. 23, No. 1, pp. 147-164.

68. Park, K. S., and Yun, D. K., (Dec, 1984), A stepwise partial enumeration algorithm for the economic lot scheduling problem, *IIE Transactions*, Vol. 16, No. 4, pp. 363-370.

69. Park, K. S., and Yun, D. K., (1985), Optimal scheduling of periodic activities, *Operations Research* Quarterly, Vol. 33, No. 3, pp. 690-695.

70. Park, K. S., and Yun, D. K., (Jun, 1987), Feasibility test for multi-product lot size scheduling on one facility, International Journal of Policy and Information, Vol. 11, No. 1, pp. 101-108.

71. Parsons, R. J., (Jul, 1966), Multi-product lot size determination when certain restrictions are active, Journal of Industrial Engineers, Vol. 17, No. 7, pp. 360-365.

72. Qiu, J., and Loulou, R., (Feb, 1995), Multi-product production/inventory control under random demands, IEEE Transactions on Automatic Control, Vol. 40, No. , pp. 350-356.

73. Rogers, J., (Apr, 1958), A Computational Approach to the Economic Lot Scheduling Problem, *Management Science*, Vol. 4, No. 3, pp. 264-291.

74. Roundy, R., (Dec, 1989), Rounding off to powers of two in continuous relaxations of capacitated lot sizing problems, *Management Science*, Vol. 35, No. 12, pp. 1433-1442.

75. Saipe, A. L., (Aug, 1977), Production runs for multiple products: the two product heuristic, *Management Science*, Vol. 23, No. 12, pp. 1321-1327.

76. Salveson, M. E., (Feb, 1953), Mathematical Theory of Production, Journal of Industrial Engineering, Vol. 4, No. 1, pp. 35877.

77. Schweitzer, P. J., and Silver, E. A., (1983), Mathematical pitfalls in the one machine multi-product economic lot scheduling problem, *Operations Research*, Vol. 31, No. , pp. 401-405.

78. Schweitzer, P. J., and Seidmann, A., (Apr, 1991), Optimizing processing rates for flexible manufacturing systems, *Management Science*, Vol. 37, No. 4, pp. 454-466.

79. Stankard, Jr, M. F., and Gupta, S. K., (Mar, 1969), A Note on Bomberger's Approach to Lot Size Scheduling: Heuristic Proposed, *Management Science*, Vol. 15, No. 7, pp. 449-452.

80. Vemuganti, R. R., (Nov, 1978), On the feasibility of scheduling lot sizes for two products on one machine, *Management Science*, Vol. 24, No. 15, pp. 1668-1673.

81. Vemuganti, R. R., (Dec, 1987), The maximum value of inventory and storage in production lot size systems, *IIE Transactions*, Vol. 19, No. 4, pp. 404-411.

82. Vemuganti, R. R., Arsham, H., and Shao, Jr., S. P., (1989), An implementation of Lagrangian decomposition in solving a multi-item production scheduling problem with changeover cost and restrictions, Mathematical and Computer Modeling, Vol. 12, No.

83. Ware, N., and Keown, B., (Jan-Mar, 1987), Common cycle scheduling: A successful application, *Production and Inventory Management* , Vol. 28, No. 1, pp. 16-22.

84. Zipkin, P. H., (Jan-Feb, 1986), Models for design and control of stochastic multi-item batch production systems, *Operations Research*, Vol. 34, No. 1, pp. 91-104.

85. Zipkin, P. H., (Jan-Feb, 1991), Computing optimal lot sizes in the economic lot scheduling problem, *Operations Research*, Vol. 39, No. 1, pp. 56-63.

VITA

John Roy Gray was born in Obion, Tennessee on January 12, 1942. He attended school in Obion where he graduated from Obion High School in May of 1960. He entered the University of Tennessee at Martin in September of 1960 and two years later transferred to the University of Tennessee at Knoxville where in June of 1964 he received a Bachelor of Science in Electrical Engineering. After working for a year with Newport News Shipbuilding and Dry Dock Company in Newport News, Virginia and for two years with the United States Air Force in Mobile, Alabama he returned to the University of Tennessee at Knoxville to pursue a graduate degree in Management Science. He received the Master of Science in Management Science in August of 1969. In August of 1990 he re-entered the Graduate School as a part-time student to pursue a Doctorate in Management Science. In December of 1996, after working at the Department of Energy's Y-12 Plant for 27 years, he took early retirement to return to the University of Tennessee full-time. He received the Doctor of Philosophy in Management Science in May of 1999.