

# JUST-IN-TIME SCHEDULES FOR THE SMALL MAKE-TO-ORDER SHOP

GORDON SINNAMON AND SUSAN MILNER

The University of Western Ontario  
and Fraser Valley College

October 15, 1996

ABSTRACT. In order to use Just-In-Time production methods in a small, Make-To-Order environment it is necessary to achieve as balanced a workload as possible. This paper gives an algorithm to determine a production schedule which balances the workload in a situation with unpredictable demand.

## 1. INTRODUCTION

The past decade has seen western manufacturers increasingly adopt elements of the Japanese Just-In-Time (JIT) system of production. A complete JIT system is made up, according to Finch and Cox [1986], of eight points: a focused factory, reduced set-up times, group technology, total preventative maintenance, cross-trained employees, uniform work loads, just-in-time delivery of purchased parts, and the Kanban or pull system of controlling production.

The benefits to large manufacturing companies are well documented, from the original developer, Toyota [Monden 1981], to Westinghouse, Hewlett-Packard and Harley Davidson [O'Grady 1988], to General Electric, RCA, the Pontiac division of GM, and IBM [Gravel and Price 1988]. These benefits include substantial decreases in turnover time as well as in

Typeset by  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$

inventories of completed work and work-in-process, increased productivity and customer satisfaction, and better quality control.

Traditionally, the JIT system has not been considered applicable to small manufacturers or to Make-To-Order (MTO) environments. In recent years, however, there has been increased exploration of the possibilities of extending some elements at least of the JIT system to smaller firms. Finch and Cox [1986] describe a small but repetitive manufacturing environment as suitable for some aspects of JIT operation. They also consider that many elements of a JIT system are attainable by small manufacturing companies which are not necessarily repetitive environments. They identify as one of the major difficulties in an MTO workplace the attainment of a stable work-load; once this has been established, the manufacturer should be able to obtain a degree of success equal to that of the Make-To-Stock (MTS) or repetitive manufacturer. This paper describes a model to achieve as stable a workload as possible in such a workplace.

Achieving a stable workload is not only one of the major difficulties in a small shop, it is also one of the most important of the eight points mentioned above. The benefits of a Just-In-Time system are largely due to the accurate feedback possible in such a system. Reducing inventory and requiring a uniform flow of work through the shop make it easier to pinpoint inefficiencies. An under-utilized station which is 'ahead of schedule' is not as obvious as one with no work to do, and a bottleneck with 12 units waiting to be processed where there should be 10 is not as obvious as one with 3 units where there should be 1. Once identified, management may move with confidence to correct bottlenecks, sources of defects, under-utilized stations, and even poor shop layouts. The ability to make such corrections easily is a requirement for JIT; hence the need for a focused factory, group

technology, and cross-trained employees; but identifying the problems is the difficult first step. This is especially true in a small shop where the responsibility for effective operations will often rest with a single individual.

Clearly a job shop cannot hope to reduce its inventory if it must respond to the traditional single-date, large order demand, although as more firms convert to JIT systems the job shop will be able to benefit from the consequent smoothing of demand. In the current market more benefit is likely to accrue from having less work-in-process to clutter the shop floor, from increased productivity, and from greater customer satisfaction due to fewer due dates missed and improved quality control. Maruchek and McClelland [1986] identify several problems faced by MTO firms, among them engineering changes and material substitutions even up to final assembly, which contribute to the serious problem of frequent changes in promise dates. The JIT system in general allows firms to achieve total quality control; the method of assigning production described in this paper enables the company to produce new items more slowly at first, which allows the manufacturer to get feedback on the design early in the process. This reduces potential waste and is likely to ensure that promise dates are adhered to much more closely.

Gravel and Price [1988] describe the operation of the Kanban method of production in a small firm. In a Kanban system each work station issues a signal to the preceding station for a particular part as the need arises for that part. This 'pulls' work through the shop, in contrast to the usual Western 'push' methods. Gravel and Price use both simulation techniques and a pilot study to show that this method can be successful in the job-shop environment. They identify the necessity of keeping production and the purchase of materials tightly coordinated, as running out of purchased materials makes impossible

the smooth functioning of the Kanban system. Small shops are, in the present market, unlikely to be able to get deliveries ‘just in time’ from suppliers so it is clearly important for management to be able to forecast the amounts of supplied materials required. The schedules produced by the methods of this paper will enable management to so coordinate the purchase of materials.

It should be noted that the schedule which evolves is not a schedule of factory production but rather a schedule of the release of the signals which will pull through the shop the production of each unit. This model provides a sequence of final assembly pull-signals. Each unit so demanded then generates its own pull signals as it progresses through the shop.

We assume here that a balanced schedule of final assembly pull signals will yield a stable work load for the shop. This assumption will be valid if each final assembly is made up of similar components. In particular, this includes fabrication shops where a single piece of raw material undergoes several operations to produce the final product. In effect each ‘assembly’ has only one component.

The paper addresses the problem of balancing the production schedule in an MTO or job-shop environment which meets the JIT criteria of small set-up times and cross-trained employees and satisfies the above assumption. It expands in part on the algorithm introduced in Miltenburg [1989] which determines production schedules for a mixed-model assembly line in a MTS setting. In that paper, ideal production levels arise naturally from demand levels, and units are to be produced in ratios as close as possible to the ratios in which they are demanded; the obvious problem in a job-shop or MTO environment is that demand is often uneven and is generally not known far in advance. Our treatment

includes the situation studied in Miltenburg[1989] as a special case—if all orders have distant due dates the MTO model reduces to the MTS case. However, in Miltenburg and Sinnamon([1989] and [1993]), the MTS problem was revisited with the above assumption removed. The resulting multi-level problem was solved in those papers. We do not attempt to solve the MTS multi-level problem here but recommend it as an area for future research. Despite appearances, it will not be a simple synthesis of this paper and the work of Miltenburg and Sinnamon. For instance, there seems to be no obvious extension of our first pass to the multi-level case.

The mathematical model of the scheduling problem is divided into three passes. The first pass provides a formula for determining ideal production levels in an environment of uneven, unpredictable demand. It is necessary to be able to respond to new orders, to keep the shop working evenly at full capacity, and to complete orders in time. The second pass is concerned with the problem of deriving realistic production targets from the ideal production levels, which generally involve fractions of models. The third pass offers an algorithm for eliminating any infeasibilities which may arise in the construction of the production targets. Eliminating these infeasibilities produces the ‘real-world’ sequence which will best balance the production schedule of the plant. The mathematical model is demonstrated throughout the exposition by the use of an example. A discussion of the efficiency of the model then follows.

## 2. DESCRIPTION OF THE PROBLEM

In the traditional MTO operation, flow control is exercised in two ways: Strategic release of jobs to the shop floor, and dispatching rules used in the shop itself. In order to

regulate the flow of work through the shop, jobs can be held in the job pool or released to take their place in the work in process on the shop floor. Strategic release can be an important method of controlling production. Dealing with high priority jobs at the job release stage is unfortunately simple—they must be released immediately and job control is forfeited.

After release, work in process is moved from machine queue to machine queue, each time subject to the dispatching rules for that machine. Job priority is broken down into subassembly priorities and fed to the dispatching rules with results that are often uncertain. Even if high priority jobs do move rapidly through the shop, the work they push aside is left standing in the aisles (often literally) as work in process inventory. In a factory organized for JIT production many of the problems are alleviated. Inventories do not mount since only downstream need can authorize production. Machines are organized in flow groups and this organization replaces dispatching rules either with the simple physical flow of parts within a groups or with Kanban links between groups. High priority jobs prompt immediate final assembly pull signals and the appropriate signals for subassemblies and parts echo upstream rapidly, altering the focus of the factory without leaving any half-finished work in process.

When this pull system of production replaces the traditional push system the job release concept is lost. Instead of initiating work on the early stages of a job, management in the JIT setting issues a pull signal for a certain finished product. The loss of strategic job release means that one method of controlling the flow of work through the shop is lost. The mechanism of control must now be the timing of pull signals, issued by management, to the final assembly stages. It is the purpose of this paper to provide an algorithm to

issue the sequence of final assembly pull signals which will yield the most balanced flow of work through the shop.

We begin by describing what we mean by a balanced (or smooth or level or stable or uniform) workload. Ideally, we want the proportion of shop capacity devoted to each final assembly to be the same in every time interval. Practically, we want to be as close to this ideal as is feasible. Scheduling such a typical mix of models provides the benefits of JIT mentioned above: Since all the tasks of the shop are done each day (or some time unit which is small in the context of the particular shop) there is no need to stockpile work-in-progress against the time its completing step is scheduled. Also, each work station works at close to its average rate so it is easy to check if a work station is over- or under-utilized and no station will be idle one week and backlogged the next.

In a MTS shop the most typical mix of final assemblies is achieved when production is kept proportional to total demand [Miltenberg, Miltenberg and Sinnamon]. In a MTO operation, however, demand is unpredictable so keeping production proportional to total demand is not an option—total demand is an unknown quantity. Other means of determining the most typical mix of final assemblies must be found. In addition, achieving a balanced work flow is necessarily an ongoing process since a new balance must be struck each time a new order is received. The algorithm presented here uses only the orders currently on the books to produce the shop schedule. No attempt is made to predict future orders. Therefore, each time a new order is received the entire schedule must be recalculated to take into account the new information available. This is necessary in the MTO environment where orders will routinely be placed for items never before produced by the shop.

In order to simplify the statement of the problem it will be assumed that all input data has been preprocessed into convenient forms. This is done in three ways.

1. *The shop is capable of producing  $n$  different final assemblies. These are called models  $1, 2, \dots, n$ .* Since new products may be engineered to each customer's order the value of  $n$  may increase each time a new order is received (and decrease each time an order is filled.) For a single run of the scheduling algorithm, however,  $n$  may be regarded as fixed.

2. *Each unit of each model takes the same length of time to produce, independent of which model is produced.* This length of time is called a *stage* and the shop operates continuously at the rate of *one unit per stage*. All other time measurements (days, weeks, shifts, etc) are converted into stages. In practice, assembly times may differ somewhat and a stage may be taken as an average. This will not cause difficulties unless assembly times are very long or vary quite widely.

3. *Orders come to the shop in the following simple form: model number, quantity, due date.* Quantity is simply the number of units and due date is expressed in stages from the present. Although a customer may order a mix of several models and specify several different due dates, these complex orders may be readily broken down into several orders of the simple type described above.

The due dates used here are determined by the shop management and are realistically set—a problem in its own right. In determining realistic due dates management faces two difficulties. The first must be addressed by any MTO shop whether or not a JIT system is in place. Managers must promise delivery times and actually make deliveries so as to attract new customers and satisfy existing customers. This will mean negotiating and renegotiating delivery dates, incurring tardiness penalties and risking loss of future



business when necessary and so on. A means of accomplishing this must be in place but it is beyond the scope of this paper. The second difficulty is that the time between a final assembly's pull signal (which is what is scheduled) and its completion is always subject to variation in practise. Thus when one counts backward from an actual delivery date to get the due date for pull signals the result will not always be on-time delivery. It is possible to leave extra time in counting back but the JIT approach would be to leave little or no slack. Any problem that arose would then be traced to its source in the shop and corrected. See Hopp, Spearman and Duenyas[1993] for a discussion of due date integrity in a pull system.

The following notation will be used in the paper.

**The Initial Data.**

$n$  the number of models.

$q$  the number of orders.

$m_j$  model number for order  $j$ ,  $1 \leq m_j \leq n$ .

$s_j$  quantity for order  $j$  in units.

$D_j$  due date for order  $j$  in stages from the present.

The orders are arranged by due date so that  $0 < D_1 \leq D_2 \leq \dots \leq D_q$ . The number of units required by due date  $D_j$  is  $\sum_{i=1}^j s_i$ . The due dates are not impossible to meet, therefore  $\sum_{i=1}^j s_i \leq D_j$ .

**Derived Quantities.**

$S$  the total number of units ordered.

The quantities for orders  $1, 2, \dots, q$  are  $s_1, s_2, \dots, s_q$  respectively. The total number of

units ordered is  $S = \sum_{i=1}^q s_i$ . Since the shop always works at the rate of one unit per stage, all production will be finished after  $S$  stages. Thus all due dates greater than  $S$  stages from the present are no restriction at all. We therefore define

$d_j$  the adjusted due date for order  $j$ .

For convenience set  $d_0 = 0$ . For  $j > 0$  set  $d_j = \min(D_j, S)$ . Notice that we still have  $0 = d_0 \leq d_1 \leq \dots \leq d_q \leq S$ .

The following three quantities are defined in Section 3.

$I_j$  the intensity index of order  $j$ .

$R_{i,j}$  the rate of work on order  $i$  during the time interval  $(d_{j-1}, d_j)$ .

$p_{v,k}$  the ideal production of model  $v$  during the first  $k$  stages.

### **Decision Variables.**

$x_{v,k}$  the number of units of model  $v$  produced during the first  $k$  stages.

*Example.* We introduce a small shop with only five orders on the books. For simplicity suppose that each item takes a day to make so that our stage is one day. The orders are as follows.

Order A: 5 units of the item “Cover” due 13 days from now.

Order B: 8 units of the item “Grate” due 30 days from now.

Order C: 7 units of the item “Cover” due 27 days from now.

Order D: 2 units of the item “Panel” due 26 days from now.

Order E: 4 units of the item “Grate” due 20 days from now.

We have  $n = 3$  models and  $q = 5$  orders for a total of  $S = 26$  units on order. The orders are designated by letters here rather than numbers because we must sort by due date

before numbering. After sorting (into the order A, E, D, C, B) the model numbers are  $m_1 = 1$  (Cover),  $m_2 = 2$  (Grate),  $m_3 = 3$  (Panel),  $m_4 = 1$  (Cover), and  $m_5 = 2$  (Grate). The quantities,  $s_j$ , the due dates,  $D_j$ , and the adjusted due dates,  $d_j$ , are easy to read off. For example  $s_3 = 2$ ,  $D_4 = 27$ , and  $d_4 = 26$ .

The problem of determining which pull signals to issue at each stage, or equivalently, which model to schedule at each stage is broken down into three passes. In Pass 1 ideal production-to-date figures are generated which, if achieved, would ensure that production was taking place in a precisely typical mix subject to meeting all due dates. In general it is not possible to meet these production levels. Pass 2 begins the process of determining production-to-date targets which approximate as closely as possible the ideal production-to-date figures of Pass 1. Very often this pass will succeed in solving the problem making Pass 3 unnecessary. In Pass 3 the production targets which best approximate the ideal levels of Pass 1 are determined by making small adjustments to the Pass 2 targets where required.

As we will see in Section 6, Passes 1 and 2 are very rapid and although Pass 3 may be slow in pathological cases, it too is generally a rapid procedure.

Before embarking on a detailed discussion of the three passes we would like to point out that more direct approaches to this scheduling problem have been tried and found wanting. Indeed, the first author's introduction to the problem (in the MTS case) was being asked to show that a certain greedy algorithm always produced optimal schedules. It did not. An example of this failure may be found in Miltenburg [1989] on page 198. Since the MTO case includes the MTS case we do not expect a simple, greedy algorithm to be successful here. If a simpler, faster alternative is desired, one can get reasonably good

schedules by omitting pass three and patching infeasibilities with one of the heuristics described in Miltenburg [1989] but Pass 3's guarantee of a schedule that deviates as little as possible from the ideal would be lost.

### 3. PASS ONE

In this section we define the variables  $I_j$ ,  $R_{i,j}$ , and  $p_{v,k}$  and show that the ideal production to date figures  $p_{v,k}$  represent the production of the most typical mix of final assemblies possible subject to meeting all due dates.

The mix of models produced by the shop may be determined from the percentages of the shop capacity devoted to work on the various orders. We call the percentage of the shop capacity devoted to work on the order  $j$  the *rate of work on order  $j$* . A typical mix of models will result if these percentages are kept fixed at all times during production. However, the due dates must be met. After an order's due date, work on that order ceases and its percentage of the shop capacity drops to zero. This is the restriction imposed by the due dates. The production-to-date figures are calculated, therefore, according to the following guiding principle: *The rates of work on the various orders are kept in proportion to one another, except that completed orders are allowed to drop out.* From the rates of work determined by this principle the ideal production-to-date figures may be calculated which are as close to the typical mix as possible while meeting all due dates.

Consider work on order  $j$ . Orders  $1, 2, \dots, j-1$  must be completed before the due date for order  $j$ . Thus the uncommitted shop capacity (the number of stages) available for work on order  $j$  is  $d_j - (s_1 + s_2 + \dots + s_{j-1})$ . Since  $s_j$  units must be produced to fill

order  $j$ , we define the intensity index of order  $j$  to be

$$I_j = \frac{s_j}{d_j - (s_1 + s_2 + \cdots + s_{j-1})}$$

Notice that the maximum intensity is 1 and that  $I_q = 1$ . The time interval consisting of those stages following the due date for order  $j - 1$  up to and including the due date for order  $j$  will be denoted  $(d_{j-1}, d_j]$ . During the interval  $(d_{j-1}, d_j]$  work progresses on the orders  $j, j + 1, \dots, q$  but work on orders  $1, 2, \dots, j - 1$  has been completed. Thus if  $i < j$  the ideal rate of work on order  $i$  during the interval  $(d_{j-1}, d_j]$  is

$$R_{i,j} = 0 \quad \text{units per stage.}$$

If  $i \geq j$  the ideal rate of work on order  $i$  during the interval  $(d_{j-1}, d_j]$  is

$$R_{i,j} = I_i \prod_{a=j}^{i-1} (1 - I_a) \quad \text{units per stage.}$$

Since the shop operates at one unit per stage we should have

$$R_{j,j} + R_{j+1,j} + R_{j+2,j} + \cdots + R_{q,j} = 1, \quad j = 1, 2, \dots, q.$$

The left hand side represents the combined work rate on all models during the time interval  $(d_{j-1}, d_j]$ . The values  $R_{i,j}$  defined above do satisfy this identity.

To meet the deadline for order  $i$  the equations

$$s_i = \sum_{j=1}^i R_{i,j} (d_j - d_{j-1}), \quad i = 1, 2, \dots, q$$

must be valid. The right hand side represents the total work on order  $i$ . Again the work rates  $R_{i,j}$  satisfy this requirement.

According to the guiding principle above, the rate of work on order  $a$  and the rate of work on order  $b$  should be in constant proportion until one of the orders is complete. That is, the expression

$$R_{a,j}/R_{b,j}$$

should not depend on  $j$  as long as  $a \geq j$  and  $b \geq j$ . This is indeed the case. To summarize, the work rates defined above do not exceed shop capacity, they meet due date requirements, and they maintain a typical mix of final assemblies.

It is now a simple matter to calculate the ideal production-to-date at each stage. Given a number of stages  $k$  the first step is to determine in which time interval  $k$  lies. If  $d_j$  is the due date immediately following  $k$  then  $k$  lies in the interval  $(d_{j-1}, d_j]$  so that  $d_{j-1} \leq k \leq d_j$ . Once this  $j$  has been found then the ideal production-to-date of model  $v$  during the first  $k$  stages is

$$p_{v,k} = \sum \left( R_{i,j}(k - d_{j-1}) + \sum_{l=1}^{j-1} R_{i,l}(d_l - d_{l-1}) \right)$$

where the outer sum is taken over all orders  $i$  such that  $m_i = v$  (order  $i$  is an order for model  $v$ .)

STAGE	PASS ONE			PASS TWO			PASS THREE			Final Schedule
	Ideal Production Levels			Production Targets			Production Targets			
	Model 1	Model 2	Model 3	Mod 1	Mod 2	Mod 3	Mod 1	Mod 2	Mod 3	
	Cover	Grate	Panel	Cover	Grate	Panel	Cover	Grate	Panel	
1	0.570	0.376	0.053	1	0	0	1	0	0	1 Cover
2	1.141	0.753	0.106	1	1	0	1	1	0	2 Grate
3	1.711	1.129	0.159	2	1	0	2	1	0	1 Cover
4	2.282	1.506	0.212	2	2	0	2	2	0	2 Grate
5	2.852	1.882	0.265	3	2	0	3	2	0	1 Cover
6	3.423	2.259	0.319	4	2	0	4	2	0	1 Cover
7	3.993	2.635	0.372	4	3	0	4	3	0	2 Grate
8	4.563	3.012	0.425	5	3	0	5	3	0	1 Cover

9	5.134	3.388	0.478	5	3	1	5	4	0 #	2 Grate
10	5.704	3.765	0.531	6	4	0 !	6	4	0 #	1 Cover
11	6.275	4.141	0.584	6	4	1	6	4	1	3 Panel
12	6.845	4.518	0.637	7	4	1	7	4	1	1 Cover
13	7.416	4.894	0.690	7	5	1	7	5	1	2 Grate
14	7.718	5.506	0.776	8	5	1	8	5	1	1 Cover
15	8.020	6.118	0.863	8	6	1	8	6	1	2 Grate
16	8.322	6.729	0.949	8	7	1	8	7	1	2 Grate
17	8.624	7.341	1.035	9	7	1	9	7	1	1 Cover
18	8.925	7.953	1.122	9	8	1	9	8	1	2 Grate
19	9.227	8.565	1.208	9	9	1	9	9	1	2 Grate
20	9.529	9.176	1.294	10	9	1	10	9	1	1 Cover
21	9.941	9.647	1.412	10	10	1	10	10	1	2 Grate
22	10.353	10.118	1.529	10	10	2	11	10	1 #	1 Cover
23	10.765	10.588	1.647	11	11	1 !	11	11	1 #	2 Grate
24	11.176	11.059	1.765	11	11	2	11	11	2	3 Panel
25	11.588	11.529	1.882	12	11	2	12	11	2	1 Cover
26	12.000	12.000	2.000	12	12	2	12	12	2	2 Grate

**Table 1:** Determination of the final schedule for the example  
! – infeasibility, # – linked chain

*Example (continued).* The Pass 1 calculations conclude with the determination of the ideal production levels of each model at each stage. For our example these are given in the first section of Table 1. Notice that the total production of all models up to stage  $k$  is  $k$  units even though fractions of units are allowed at this point in the process.

Deadlines are also being met, for instance, at stage  $k = 13$  the ideal production levels are  $p_{1,13} = 7.416$ ,  $p_{2,13} = 4.894$ , and  $p_{3,13} = 0.690$ . That is, by the end of the 13'th stage the 5 units of model 1 (the Cover) from order 1 will have been completed (for delivery at  $k = 13$ ), and 2.416 units will have been produced towards the 7 units of model 1 required by stage  $k = 26$  to meet order 4. The production of models 2 and 3 is building to meet the due dates for the other three orders.

The advantage of allowing fractions of models at this stage is that rates of work can be seen to remain fairly constant. 0.053 units per stage is the rate of work on model 3

from stages 1 to 13, this increases to 0.086 from stages 14 to 20 and to 0.118 for the last 6 stages before its delivery date.

#### 4. PASS TWO

Ideal production-to-date figures have been calculated in Pass 1. The variable  $p_{v,k}$  represents the ideal production of model  $v$  during the first  $k$  stages. These figures are called “ideal” because they have been determined without regard to an important restriction—production of actual models can only occur in whole numbers of units. In the example the ideal production-to-date of models 1, 2, and 3 in the first stage turned out to be 0.570, 0.376, and 0.053 units respectively. This does not resemble a useful production schedule. Ideal production-to-date figures are useful nonetheless. They provide a standard against which to measure actual schedules. The schedule which produces actual production-to-date numbers as close as possible to the ideal figures will be the preferred schedule. In this section the closest to ideal production-to-date *using only integers* is determined for each model at each stage. These will be called production targets. The production targets are found by solving the following problem for each stage  $k$ .

**Closest Integer Problem.** *For a fixed integer  $k$  find integers  $x_{1,k}, x_{2,k}, \dots, x_{n,k}$  which minimize*

$$\sum_{v=1}^n \alpha_{v,k} (x_{v,k} - p_{v,k})^2$$

*subject to  $\sum_{v=1}^n x_{v,k} = k$ .*

The constraint  $\sum_{v=1}^n x_{v,k} = k$  expresses the fact that the total production of all models during the first  $k$  stages is  $k$  units—as always the shop operates at one unit per stage. The term  $(x_{v,k} - p_{v,k})^2$  is the squared deviation of the integer  $x_{v,k}$  (actual production-to-date



of model  $v$ ) from the number  $p_{v,k}$  (ideal production-to-date of model  $v$ .) Summing this over all models and minimizing will yield the closest integers to ideal production after  $k$  stages.

The weights  $\alpha_{v,k}$  are positive constants which allow the relative importance of keeping the production of the various models close to ideal production to be adjusted by management. If it is highly desirable, for example, that the actual production of model 3 be kept close to the ideal production level then the weights  $\alpha_{3,k}$  may be set higher than the other weights  $\alpha_{1,k}, \alpha_{2,k}, \alpha_{4,k}, \dots, \alpha_{n,k}$ . Of course more delicate adjustments than these may be made. For example, it may be of value to have some simple scheme whereby the importance of the various orders (above and beyond due date priority) would be translated into values for the weights  $\alpha_{1,k}, \alpha_{2,k}, \dots, \alpha_{n,k}$ .

The dependence of the weights on  $k$  is of no importance in Pass 2 since no interaction is allowed between production levels at the various stages. In Pass 3 such interaction may take place and the dependence of the weights on  $k$  may provide an additional measure of management control. This will be discussed further in the next section.

*Algorithm to solve the Closest Integers Problem.*

Input data:

$k$  a fixed integer.  $1 \leq k \leq S$ .

$\alpha_{v,k}$  a positive weight for each  $v = 1, 2, \dots, n$ .

$p_{v,k}$  from Pass 1, satisfying  $\sum_{v=1}^n p_{v,k} = k$ .

- (1) For each  $v = 1, 2, \dots, n$  set  $x_{v,k} = (p_{v,k}$  rounded to the nearest integer).
- (2) Let  $sum = \sum_{v=1}^n x_{v,k}$ . If  $sum < k$  go to (3). If  $sum > k$  go to (4). If  $sum = k$  go to (5).

- (3) Find  $w$  such that  $\alpha_{w,k}(x_{w,k} - p_{w,k} + \frac{1}{2})$  is a minimum. Increase  $x_{w,k}$  by 1. Go to (2).
- (4) Find  $w$  such that  $\alpha_{w,k}(x_{w,k} - p_{w,k} - \frac{1}{2})$  is a maximum. Decrease  $x_{w,k}$  by 1. Go to (2).
- (5) Stop.  $x_{1,k}, x_{2,k}, \dots, x_{n,k}$  are the required integers.

For a justification of this algorithm see Appendix I.

The Closest Integers Problem can now be solved for each stage  $k = 1, 2, \dots, S$ . The result is a sequence of production targets from which may be read off the most desirable level of production for each model at each stage. If a schedule for the shop results in actual production equal to target production at each stage then the schedule cannot be improved. Moreover, if such a schedule is possible it is easily determined from the production targets themselves. Consider the targets at stage  $k$ ;  $x_{1,k}, \dots, x_{n,k}$ ; and the targets at stage  $k + 1$ ;  $x_{1,k+1}, \dots, x_{n,k+1}$ . The sum of the stage  $k$  targets is  $k$  and the sum of the stage  $k + 1$  targets is  $k + 1$ . Because of this, and because of the way that the  $x_{v,k}$  are determined it generally happens that  $x_{v,k} = x_{v,k+1}$  for all but one model  $v$  and for the remaining model, say model  $w$ ,  $x_{w,k} + 1 = x_{w,k+1}$ . When this is the case it is clear that model  $w$  should be scheduled in stage  $k$ .

*Example (continued).* The production targets for the example are given in the Pass 2 section of Table 1. Although the targets yield a production schedule for the first few stages, a problem arises in passing from stage 9 to stage 10. The sequence of models for stages 0 to 9 is easily seen to be 1, 2, 1, 2, 1, 1, 2, 1, 3 but in passing from the stage 9 target to the stage 10 target production of model 3 drops from 1 to 0—certainly actual

production cannot follow this lead. Another such difficulty is encountered in passing from stage 22 to stage 23. It will be the task of Pass 3 to eliminate these difficulties.

The step from one production target to the next is called *feasible* provided the target production of each model does not drop. Otherwise the step is called *infeasible*. Notice that if a step is feasible then the step is always simply the production of one unit of one model with the production targets of all other models remaining constant.

The existence of infeasible steps in this type of scheduling problem was encountered in [Miltenburg 1989] and examined further in [Miltenburg and Sinnamon 1989] and [Miltenburg and Sinnamon 1992]. The solution there was to apply one of several heuristic techniques to adjust the production targets locally and eliminate the infeasible steps. Unfortunately, once the heuristics were applied there was no longer any guarantee that the resulting schedule was best possible. This difficulty is overcome in the next section. An exact algorithm is given which eliminates any infeasible steps which may have arisen in Pass 2 and guarantees that the resulting schedule has the smallest possible (weighted sum of squares) variation from the ideal production levels of Pass 1.

## 5. PASS THREE

In the previous section targets were produced which minimized variation from ideal production levels and satisfied the constraint  $\sum_{v=1}^n x_{v,k} = k$  for each stage  $k$ . The additional constraint that production levels must never decrease was not imposed and occasionally the production targets generated in Pass 2 will violate this constraint. It is important to stress that if this constraint is not violated by the Pass 2 targets (ie there are no infeasible steps) then the final schedule may be determined without going to Pass 3.

It is the purpose of this section to give an algorithm which will adjust the production targets of Pass 2 in order to eliminate infeasible steps and to make these adjustments in such a way that the resulting production targets do yield a schedule and, subject to this requirement, minimize variation from ideal levels.

The ideal production-to-date figures still provide the standard against which the actual schedules are measured. If there are infeasible steps in the Pass 2 targets then it is not sufficient to solve the CIP at each stage and place the solutions in sequence. It becomes necessary to perform a single minimization step involving all the decision variables at the same time. This is the

**Just-in-time Sequencing Problem.** *Find integers  $x_{v,k}$ ,  $v = 1, 2, \dots, n$ ;  $k = 0, 1, \dots, S$ , which minimize*

$$DEVIATION = \sum_{k=0}^S \sum_{v=1}^n \alpha_{v,k} (x_{v,k} - p_{v,k})^2$$

*subject to the constraints*

$$\sum_{v=1}^n x_{v,k} = k, \quad k = 0, 1, 2, \dots, S,$$

*and*

$$x_{v,k-1} \leq x_{v,k}, \quad v = 1, 2, \dots, n; k = 1, 2, 3, \dots, S.$$

The approach to this problem begins with the Pass 2 targets. These have been computed for each stage  $k$  in Pass 2. No sequence of production levels which satisfy the two constraints of the JSP has a lower value of DEVIATION than the sequence of Pass 2 targets. The only reason that they may not be the solution to the JSP is that they may have infeasible steps and therefore violate the second constraint of the JSP.

To solve the JSP successive adjustments are made to the Pass 2 targets. Each time an adjustment is made the resulting sequence is guaranteed to still have a lower value of DEVIATION than any sequence of production levels which satisfy the two constraints of the JSP. (The new value of DEVIATION will, however, be larger than for the unadjusted Pass 2 targets.) It follows that if (after some number of adjustments) the resulting sequence satisfies the JSP constraints then it will necessarily minimize DEVIATION and constitute a solution to the JSP. It remains to describe the adjustments and show that eventually the constraints will be satisfied.

Some precise language will facilitate the description of the Pass 2 adjustments.

The *stage  $k$  target* refers to the production targets  $x_{1,k}, x_{2,k}, \dots, x_{n,k}$  now considered as a single entity.

The *sequence of targets* refers to the sequence: stage 1 target, stage 2 target,  $\dots$ , stage  $S$  target.

A *chain* of targets refers to a collection of *consecutive* targets in the sequence of targets.

A *feasible chain* is a chain which contains no infeasible step. Note that chains of length 1 are automatically feasible.

The main technique used in the adjustment procedure is the solution of the

**Extended Closest Integers Problem.** For fixed integers  $k$  and  $r$  find integers  $x_{v,l}$ ,  $v = 1, 2, \dots, n$ ;  $l = k, k + 1, \dots, k + r - 1$ ; which minimize

$$\sum_{l=k}^{k+r-1} \sum_{v=1}^n \alpha_{v,l} (x_{v,l} - p_{v,l})^2$$

subject to

$$\sum_{v=1}^n x_{v,l} = l, \quad l = k, k + 1, \dots, k + r - 1,$$

and

$$x_{v,l-1} \leq x_{v,l}, \quad v = 1, 2, \dots, n; l = k + 1, \dots, k + r - 1.$$

Note that for  $r = 1$  this reduces to the CIP and for  $k = 0$  and  $r = S + 1$  this becomes the JSP.

Solving the JSP as a special case of the ECIP is impractical since the algorithm for the ECIP has efficiency on the order of  $n^r$ . For small  $r$ , however, this is quite practical—and this is where the solution to the ECIP is used.

The algorithm which solves the ECIP is similar to, but somewhat more technical than, the algorithm presented in Pass 2 for the CIP. It is therefore relegated to Appendix II.

There is one more concept to introduce before describing the procedure for adjusting the Pass 2 targets. That is the idea of *linking* a chain of targets. This notion is used within the adjustment procedure to keep track of adjustments that have gone before. If a chain of targets is linked by the procedure then the procedure will never adjust one target in the linked chain without adjusting them all. When the procedure begins with the Pass 2 targets no chain is linked.

*Procedure for adjusting targets.*

- (1) If there are no infeasible steps stop. The current sequence of targets is a solution to the JSP.
- (2) Find an infeasible step and identify the smallest chain which contains this infeasible step and which does not begin or end inside any previously linked chain. Note that if the identified chain contains a part of any previously linked chain then it contains the entire linked chain.

- (3) Solve the ECIP using the initial stage  $k$  and the length  $r$  of the identified chain for input. Replace the targets within the identified chain by the solution to the ECIP.
- (4) Link the identified chain and return to step (1).

*Example (continued).* In our example the step from stage 9 to stage 10 was seen to be infeasible after Pass 2. When we identify the chain consisting of stages 9 and 10 and solve the ECIP we obtain the new targets shown in the Pass 3 section of Table 1. To fix the infeasible step from stage 22 to 23 the ECIP is solved for the chain consisting of those two stages. All other targets remain unchanged. Since no infeasible steps remain it is possible to read off the final schedule from the production targets (Table 1). This sequence of models gives the order of final assembly pull signals to be issued to the shop. Notice that all due dates will be met.

At first glance it appears that the adjustment procedure must be run through exactly once for each infeasible step. This is not necessarily the case. It is possible that adjusting the targets within the identified chain may introduce new infeasible steps—not within the identified chain since the solution to the ECIP is always a feasible chain—but at the endpoints of the identified chain. The question arises, therefore, whether the procedure will ever succeed in eliminating all infeasible steps. To see that it will succeed note that each time step 4 is performed there are, overall, more stages within linked, feasible chains than previously. If the procedure has not stopped by the time all steps are linked into a single feasible chain it will certainly do so then.

When the procedure does stop, as it must, the result will be a sequence of targets with

no infeasible step. Thus it satisfies the constraints of the JSP. The remaining question is: Does the resulting sequence indeed minimize DEVIATION subject to the constraints? The answer, of course, is yes it does, and the reasoning relies on the notion of linkage. The resulting sequence of targets may be thought of as a sequence of chains—either linked chains or original Pass 2 targets (chains of length 1.) Each Pass 2 target is the solution to the CIP for that stage. No other target for that stage contributes less to DEVIATION than the Pass 2 target. Each linked chain is the solution to the ECIP for that range of stages. No other *feasible* chain for that range of stages contributes less to DEVIATION than the solution to the ECIP. Thus each chain in the sequence of chains contributes the least amount possible to DEVIATION so DEVIATION is minimized.

## 6. THE EFFICIENCY OF THE ALGORITHM

The three passes which make up the algorithm are performed successively so their efficiencies may be considered separately. Pass 1 is very rapid consisting as it does of straightforward combinations of the initial data. On the order of  $q^2 + S$  operations are required. (Recall that  $q$  is the number of orders and  $S$  is the total quantity on order.) This is a very modest requirement and, indeed, tests show the Pass 1 times to be negligible.

Pass 2 is also fast. The CIP is solved  $S$  times and each time on the order of  $n$  operations are performed. Again, in tests the time for Pass 2 is small.

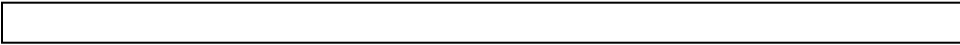
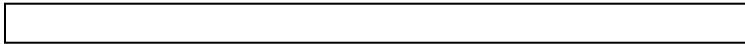


The problem of assessing the theoretical efficiency of Pass 3 is formidable. There seems to be no simple way to predict how many infeasibilities will occur in the Pass 2 targets or how long the linked sections will become during the resolution of those infeasibilities that do occur. The latter question is the important one. The number of operations required

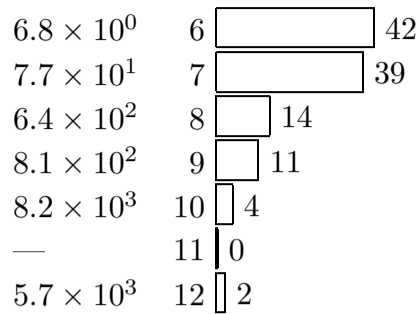


(in the ECIP) to link a chain of length  $r$  is on the order of  $n^{r+1}$  so the length of the longest chain which must be linked in Pass 3 will essentially determine the time taken.

In order to examine the efficiency of Pass 3 the algorithm (all three passes) was run 813 times with randomly generated data. There were between 2 and 20 orders for between 2 and 12 models with actual numbers independently uniformly distributed. The quantity for each order was randomly selected so that the total number of units on order was between 2 and 400. Figure 1 is a histogram of the number of times that the longest linked chain had length  $r$  for all values of  $r$ . (In our implementation of the algorithm only chains of length 3 or more were linked.) It is clear even from this small number of trials that very long linked chains are highly unlikely to occur. Moreover, in nearly one third of the trials (258 of 813) no chains were linked, meaning that Pass 2 produced the final schedule and Pass 3 was not required.

The close connection between the time taken by the algorithm and the length of the longest linked chain is illustrated by the "cpu seconds" column in Figure 1. (Problems were run on a single processor CDC 4680.) The column reports the average of the times taken for all trials having each of the  $r$  values. Because of the small number of trials having  $r$  values of 8, 9, 10, and 12, the averages may not be representative but it is clear that very long times can occur. Despite the occasional occurrence of very long times the algorithm is generally rapid: Over 90% of the trials finished in under one second and over 95% finished in under one minute.

CPU sec.			
0.00	0		258
$2.0 \times 10^{-3}$	3		199
$4.5 \times 10^{-2}$	4		156
$4.4 \times 10^{-1}$	5		88

**Figure 1:** Frequency of longest linked chain

## ACKNOWLEDGEMENTS

The authors would like to thank Dr. John Miltenburg for suggesting several improvements to the paper. Support from the Natural Sciences and Engineering Research Council and from Fraser Valley College is gratefully acknowledged

## APPENDIX I. THE CLOSEST INTEGERS PROBLEM

The object of this discussion is to demonstrate that the algorithm given by (1)-(5) of Section 4 solves the CIP. To do this we use the following result.

**Theorem 1.** *Suppose  $k$  is an integer,  $p_1, p_2, \dots, p_n$  are real numbers,  $\alpha_1, \alpha_2, \dots, \alpha_n$  are weights (non-negative real numbers,) and  $x_1, x_2, \dots, x_n$  are integers satisfying  $x_1 + x_2 + \dots + x_n = k$ . The following two statements are equivalent.*

$$(I.1) \quad \sum_{\nu=1}^n \alpha_{\nu} (x_{\nu} - p_{\nu})^2 \leq \sum_{\nu=1}^n \alpha_{\nu} (y_{\nu} - p_{\nu})^2$$

whenever  $y_1, y_2, \dots, y_n$  are integers such that  $y_1 + y_2 + \dots + y_n = k$ .

$$(I.2) \quad \max_{1 \leq \nu \leq n} \alpha_{\nu} (x_{\nu} - p_{\nu} - \frac{1}{2}) \leq \min_{1 \leq \nu \leq n} \alpha_{\nu} (x_{\nu} - p_{\nu} + \frac{1}{2}).$$

Then proof of Theorem 1 will be given shortly, but first we indicate how the theorem may be used to justify the algorithm of Section 4. According to Theorem 1,

$x_{1,k}, x_{2,k}, \dots, x_{n,k}$  solves the CIP at stage  $k$  provided

$$(I.3) \quad x_{1,k}, x_{2,k}, \dots, x_{n,k} \text{ are integers,}$$

$$(I.4) \quad x_{1,k} + x_{2,k} + \dots + x_{n,k} = k, \text{ and}$$

$$(I.5) \quad \max_{1 \leq \nu \leq n} \alpha_\nu(x_{\nu,k} - p_{\nu,k} - \frac{1}{2}) \leq \min_{1 \leq \nu \leq n} \alpha_\nu(x_{\nu,k} - p_{\nu,k} + \frac{1}{2}).$$

In the algorithm,  $x_{1,k}, x_{2,k}, \dots, x_{n,k}$  are originally defined, in (1), to be integers and are only adjusted, in (3) or (4), by adding or subtracting 1. Thus they always remain integers and so (I.3) is satisfied. Since the variable *sum* gets closer to  $k$  by 1 with every cycle, the algorithm will certainly stop since it does so exactly when  $sum = k$ . Thus (I.4) holds.

To show that (I.5) holds for the result of the algorithm, we show that it holds for the original definition, in (1), of  $x_{1,k}, x_{2,k}, \dots, x_{n,k}$  and that it is preserved whenever there is an adjustment, in (3) or (4). Originally,  $x_{\nu,k}$  is chosen as close as possible to  $p_{\nu,k}$  so that we have  $-\frac{1}{2} \leq x_{\nu,k} - p_{\nu,k} \leq \frac{1}{2}$  for each  $\nu$ . Looking at (I.5) we see that the left hand side is not greater than 0 and the right hand side is not less than 0 so (I.5) is satisfied initially.

Suppose that we enter (3) with some  $x_{1,k}, x_{2,k}, \dots, x_{n,k}$  satisfying (I.5).  $w$  is chosen so that  $\alpha_{w,k}(x_{w,k} - p_{w,k} + \frac{1}{2})$  is a minimum. The action of (3) will be to replace  $x_{w,k}$  by  $x_{w,k} + 1$ . To show that (I.5) will be preserved we note that the right hand side will not decrease so if the left hand side does not increase then (I.5) is preserved. If the left hand side does increase then it must increase to  $\alpha_{w,k}((x_{w,k} + 1) - p_{w,k} - \frac{1}{2}) = \alpha_{w,k}(x_{w,k} - p_{w,k} + \frac{1}{2})$ . The choice of  $w$  ensures that this is still less than or equal to the right hand side so (I.5) is preserved. The argument to show that (I.5) is preserved in (4) is similar.

We conclude that the algorithm does indeed solve the CIP.

*Proof of Theorem 1.* Suppose (I.1) holds. Choose  $i$  and  $j$  so that  $\alpha_i(x_i - p_i - \frac{1}{2})$  is a maximum and  $\alpha_j(x_j - p_j + \frac{1}{2})$  is a minimum. (I.2) becomes

$$(I.6) \quad \alpha_i(x_i - p_i - \frac{1}{2}) \leq \alpha_j(x_j - p_j + \frac{1}{2}).$$

Since  $\alpha_1, \alpha_2, \dots, \alpha_n$  are non-negative this is immediate if  $i = j$ . If  $i \neq j$  define  $y_1, y_2, \dots, y_n$  by  $y_i = x_i + 1$ ,  $y_j = x_j - 1$  and  $y_\nu = x_\nu$  when  $\nu$  is neither  $i$  nor  $j$ . By (I.1),

$$\sum_{\nu=1}^n \alpha_\nu(x_\nu - p_\nu)^2 \leq \sum_{\nu=1}^n \alpha_\nu(y_\nu - p_\nu)^2.$$

Cancelling identical terms in the two sums leaves

$$\alpha_i(x_i - p_i)^2 + \alpha_j(x_j - p_j)^2 \leq \alpha_i((x_i - 1) - p_i)^2 + \alpha_j((x_j + 1) - p_j)^2$$

which can be rearranged to yield (I.6). We have shown that (I.1) implies (I.2) so it remains to establish the converse implication.

Suppose that (I.2) holds and let  $C = \max_{1 \leq \nu \leq n} \alpha_\nu(x_\nu - p_\nu - \frac{1}{2})$ . For all  $\nu$  we have

$$(I.7) \quad \alpha_\nu(x_\nu - p_\nu - \frac{1}{2}) \leq C \quad \text{and} \quad C \leq \alpha_\nu(x_\nu - p_\nu + \frac{1}{2}).$$

We will deduce (I.1) by showing that

$$S = \sum_{\nu=1}^n \alpha_\nu(y_\nu - p_\nu)^2 - \sum_{\nu=1}^n \alpha_\nu(x_\nu - p_\nu)^2$$

is non-negative. We calculate as follows.

$$\begin{aligned} S &= \sum_{\nu=1}^n \alpha_\nu(y_\nu - x_\nu)(y_\nu + x_\nu - 2p_\nu) \\ &= \sum_{y_\nu \geq x_\nu + 1} \alpha_\nu(y_\nu - x_\nu)(y_\nu + x_\nu - 2p_\nu) + \sum_{y_\nu \leq x_\nu - 1} \alpha_\nu(x_\nu - y_\nu)(2p_\nu - y_\nu - x_\nu) \end{aligned}$$

since terms in which  $y_\nu = x_\nu$  drop out. In the left hand sum the first two factors,  $\alpha_\nu$  and  $(y_\nu - x_\nu)$ , are non-negative. Thus the sum will decrease if  $y_\nu$  is decreased to  $x_\nu + 1$  in the third factor. Similarly, in the right hand sum the first two factors,  $\alpha_\nu$  and  $(x_\nu - y_\nu)$ , are non-negative. Thus the sum will decrease if  $-y_\nu$  is decreased to  $-x_\nu + 1$  in the third factor. Therefore,

$$\begin{aligned} S &\geq \sum_{y_\nu \geq x_\nu + 1} \alpha_\nu (y_\nu - x_\nu) (x_\nu + 1 + x_\nu - 2p_\nu) + \sum_{y_\nu \leq x_\nu - 1} \alpha_\nu (x_\nu - y_\nu) (2p_\nu - x_\nu + 1 - x_\nu) \\ &= \sum_{y_\nu \geq x_\nu + 1} 2(y_\nu - x_\nu) \alpha_\nu (x_\nu - p_\nu + \frac{1}{2}) + \sum_{y_\nu \leq x_\nu - 1} 2(x_\nu - y_\nu) \alpha_\nu (p_\nu - x_\nu + \frac{1}{2}). \end{aligned}$$

To complete the proof we use the estimates (I.7) and add in the zero terms again.

$$\begin{aligned} S &\geq \sum_{y_\nu \geq x_\nu + 1} 2(y_\nu - x_\nu)C + \sum_{y_\nu \leq x_\nu - 1} 2(x_\nu - y_\nu)(-C) \\ &= \sum_{\nu=1}^n 2C(y_\nu - x_\nu) = 2C \left( \sum_{\nu=1}^n y_\nu - \sum_{\nu=1}^n x_\nu \right) = 2C(k - k) = 0. \end{aligned}$$

## APPENDIX II. THE EXTENDED CLOSEST INTEGERS PROBLEM

The purpose of this appendix is to solve the Extended Closest Integers problem stated in Section 5. We begin by solving the following problem.

**CIP<sup>+</sup>.** *Suppose the models to be produced in stages  $k, k + 1, \dots, k + r$  are fixed. Note that in this case the production target for stage  $k$  immediately determines the production targets for stages  $k, k + 1, \dots, k + r$ . What is the stage  $k$  production target for which*

$$(II.1) \quad \sum_{l=k}^{k+r-1} \sum_{\nu=1}^n \alpha_{\nu,l} (x_{\nu,l} - p_{\nu,l})^2$$

*is a minimum?*

We will show that the CIP<sup>+</sup> reduces to the CIP which has been solved in Section 4 and Appendix 1. The ECIP is then solved as follows: Enumerate all possible model sequences

for stages  $k, k+1, \dots, k+r$ . For each fixed model sequence solve the CIP<sup>+</sup>. The solution to the ECIP is the model sequence, together with the associated stage  $k$  target produced by the CIP<sup>+</sup>, for which (II.1) is a minimum.

To reduce the CIP<sup>+</sup> to the CIP fix the models  $e_k, e_{k+1}, \dots, e_{k+r-1}$  which are to be produced in stages  $k, k+1, \dots, k+r-1$  respectively. With these fixed the following relation between the stage  $k$  target and the stage  $l$  targets for  $k \leq l \leq k+r$  may be deduced:

$$x_{i,l} - x_{i,k} \text{ is the number of times that model } i \text{ appears among } e_k, e_{k+1}, \dots, e_{l-1}.$$

For convenience we set  $y_{i,l} = p_{i,l} + x_{i,k} - x_{i,l}$  and note that the values of  $y_{i,l}$  are determined by the sequence  $e_k, e_{k+1}, \dots, e_{k+r-1}$  alone and so are fixed for this discussion. Using these relations we can rewrite (II.1) as follows.

$$\begin{aligned} \sum_{l=k}^{k+r} \sum_{\nu=1}^n \alpha_{\nu,l} (x_{\nu,l} - p_{\nu,l})^2 &= \sum_{\nu=1}^n \sum_{l=k}^{k+r} \alpha_{\nu,l} (x_{\nu,k} - y_{\nu,l})^2 \\ &= \sum_{\nu=1}^n \left[ \left( \sum_{l=k}^{k+r} \alpha_{\nu,l} \right) x_{\nu,k}^2 - \left( \sum_{l=k}^{k+r} 2\alpha_{\nu,l} y_{\nu,l} \right) x_{\nu,k} + \left( \sum_{l=k}^{k+r} \alpha_{\nu,l} y_{\nu,l}^2 \right) \right] \\ &= \sum_{\nu=1}^n A_{\nu} (x_{\nu,k} - Y_{\nu})^2 + C \end{aligned}$$

where

$$A_{\nu} = \sum_{l=k}^{k+r} \alpha_{\nu,l}, \quad Y_{\nu} = (2A_{\nu})^{-1} \sum_{l=k}^{k+r} 2\alpha_{\nu,l} y_{\nu,l}, \quad \text{and } C = \sum_{\nu=1}^n \left( -Y_{\nu}^2 + \sum_{l=k}^{k+r} \alpha_{\nu,l} y_{\nu,l}^2 \right).$$

Since  $C$  is independent of  $x_{1,k}, x_{2,k}, \dots, x_{n,k}$  we may minimize the expression (II.1) by minimizing  $\sum_{\nu=1}^n A_{\nu} (x_{\nu,k} - Y_{\nu})^2$  over all integers  $x_{\nu,k}$  such that  $\sum_{\nu=1}^n x_{\nu,k} = k$ . This is just the CIP with weights  $A_1, A_2, \dots, A_n$  and ideal production levels  $Y_1, Y_2, \dots, Y_n$ .

## REFERENCES

- B. J. Finch and J. F. Cox, *An examination of just-in-time management for the small manufacturer: with an illustration*, International Journal of Production Research **24** (1986), 329–342.
- M. Gravel and W. L. Price, *Using the Kanban in a job shop environment*, International Journal of Production Research **26** (1988), 1105–1118.
- H. Groeffin, H. Luss, M. B. Rosenwein, and E. T. Wahls, *Final assembly sequencing for Just-In-Time manufacturing*, International Journal of Production Research **27** (1989), 199–213.
- W. Hopp, M. Spearman, and I. Duenyas, *Economic Production Quotas for Pull Manufacturing Systems*, IIE Transactions **25** (1993), 71–79.
- A. S. Marucheck and M. K. McClelland, *Strategic issues in make-to-order manufacturing*, Production and Inventory Management (1986), 82–95.
- J. Miltenburg, *Level schedules for mixed-model assembly lines in just-in-time production systems*, Management Science **35** (1989), 192–207.
- J. Miltenburg and G. Sinnamon, *Scheduling mixed-model multi-level just-in-time production systems*, International Journal of Production Research **27** (1989), 1487–1509.
- J. Miltenburg and G. Sinnamon, *Algorithms for scheduling multi-level just-in-time production systems*, IIE Transactions (1992) (to appear).
- Y. Monden, *Smoothed production lets Toyota adapt to demand changes and reduce inventory*, Industrial Engineering **13** (1981), 42–51.
- P.J. O’Grady, *Putting the just-in-time philosophy into practice*, Nichols, 1988.

GORDON SINNAMON, DEPARTMENT OF MATHEMATICS, UNIVERSITY OF WESTERN ONTARIO, LONDON, ONTARIO, N6A 5B7, CANADA  
E-mail address: [sinnamon@uwo.ca](mailto:sinnamon@uwo.ca)

SUSAN MILNER, DEPARTMENT OF MATHEMATICS, FRASER VALLEY COLLEGE, ABBOTSFORD, BRITISH COLUMBIA, CANADA