CYCLIC SCHEDULING TO MINIMIZE INVENTORY IN A BATCH FLOW LINE

Gregory Dobson
William E. Simon Graduate School
of Business Administration
University of Rochester
Rochester, NY 14627

Candace Arai Yano
Department of Industrial & Operations Engineering
University of Michigan
Ann Arbor, MI 48109-2117

Technical Report 90-36

December 1990

CYCLIC SCHEDULING TO MINIMIZE INVENTORY

IN A BATCH FLOW LINE

Abstract

This paper addresses the problem of cyclic production in a flow line. We assume a constant supply of raw materials and a constant demand for all finished goods. Material which has completed processing at one stage is transferred to the next stage in small transfer batches. Inventory may be held before the line, at the end of the line, or between any pair of adjacent stations. The objective is to find a sequence of production and a cycle length that minimize the average cost per unit time of holding inventory. A linear programming formulation is given that determines the optimal cycle length and finishing times for a given (set of) sequence(s). Two heuristics are presented for computing near-optimal sequences; one is applicable to a 2-machine flow line and the other is applicable to an m-machine line. The conclusions of the computational study are threefold: 1) permutation schedules, i.e., same sequence on all machines, are nearly always optimal, 2) the heuristics produce near optimal solutions, 3) the batching decision, i.e the choice of cycle length, is substantially more significant than the sequencing decision for minimizing inventory costs.

1. Introduction

Flow lines are a common means of producing discrete parts. In a flow line, each part visits a series of machines in the same sequence. Flow lines that produce multiple parts typically are designed so that, given the anticipated product mix, the total workload on each machine is roughly the same. In other instances, however, technological considerations and changes in the product mix may cause the various workloads to differ widely. In either case, the processing rate and, where applicable, the setup time, for each product is likely to differ across machines. As a consequence of these differences, the sequence in which products are produced may have an impact on both the total value of the inventory in the system and the amount of buffer space required between adjacent machines to accommodate work-in-process (WIP) inventory.

We consider a flow line which produces several types of parts, each with a constant demand rate. Each part has a known processing rate and a sequence-independent setup time (which may be zero) on each machine. Our goal is to find a cyclic (pure rotation) schedule that minimizes the average cost per unit time of the sum of finished goods, WIP, and raw materials inventory. By appropriately choosing the inventory holding costs, we can also consider such objectives as minimizing average inventory (in units), or minimizing average WIP inventory. Most of this paper deals with the case of two machines, but one of the proposed heuristics is applicable to larger numbers of machines.

Our work on this problem was motivated by several applications in discrete parts manufacturing environments where the demand rates for the parts are fairly constant, such as component fabrication systems supplying parts to fixed-pace automobile assembly lines, systems in which capacity constraints dictate constant production targets, and highly automated systems producing parts at a relatively constant rate for long-term contracts. Our concern about WIP inventory is in the spirit of just-in-time principles of reducing overall inventory levels for the sake of faster feedback with regard to quality, and to provide a competitive advantage by using

"continuous flow manufacturing" to reduce lead times (cf. Singh 1990). In addition, we are aware of several instances in which inter-machine buffer space has been reduced, presumably to force inventory reductions, and thus prompting the need for more careful scheduling.

The remainder of this paper is organized as follows. Section 2 provides a formal problem statement and a brief review of related literature. In Section 3 we develop two formulations. The first formulation is useful for understanding the various components of the cost function and the nature of the constraints. The second formulation is more useful for developing heuristic solution procedures. Properties of the optimal solution and related conjectures are discussed in Section 4. This provides the foundation for development of heuristic procedures in Section 5. We also develop worst case error bounds for these heuristics. In Section 6, we report results obtained in a series of computational experiments. Section 7 concludes the paper with a summary and discussion.

2. Problem Statement and Literature Review

We investigate the problem of finding cyclic schedules for a flow line which produces multiple types of parts. The demand rates are known and constant. Production rates and setup times are assumed to be deterministic, and the latter are assumed to be sequence-independent. We assume that a pure rotation policy is used on each machine; that is, a particular permutation of the products is repeated again and again, but the sequences on the machines may differ. Pure rotation schedules are used frequently because of their ease of implementation. In addition, Jones and Inman (1989) have found that pure rotation schedules perform nearly as well as more complicated schedules in single-machine problems.

The objective is to minimize the average cost per unit time of finished goods, WIP, and raw materials inventory. We assume that the transfer batch size is very small in comparison to the total quantity produced in a production run, and we model it as if it were infinitesimal. Thus, the inventory can be viewed as flowing continuously from an upstream buffer (or raw materials) into a

machine while it is producing, from a machine into its downstream buffer while the machine is producing, and from the final machine into finished goods inventory while the final machine is producing. For ease of exposition, we assume that one unit of output of a machine is required for each unit of output of its downstream machines, but any constant multiplicative relationship can be incorporated.

A restricted version of this problem has been addressed by El-Najdawi (1989), who considers a two-machine problem in which the permutation sequences on the two machines are required to be identical, and where the transfer batch for the upstream machine, but not the downstream machine, is equal to the production batch. Thus, our analysis can be viewed as a generalization of his with regard to sequencing, and somewhat more pragmatic with regard to transfer batches in view of the emphasis on inventory reduction. In addition, we investigate some subtle decisions that were ignored in El-Najdawi's analysis. This point is described in more detail later.

There is a considerable amount of research on the single-machine economic lot scheduling problem (ELSP), of which the pure rotation scheduling problem is a special case. The pure rotation problem was first studied by Hanssmann (1962). Recent research on the ELSP includes Dobson (1988) and Gallego (1990). It is useful to point out that since all of the research on the single-machine ELSP deals with only finished goods inventory, the fundamental nature of our problem is somewhat different because it raises the possibility of a tradeoff between work-in-process and finished goods inventory. This tradeoff is explained by way of an example in Appendix 1.

Finally, there is some work on cyclic scheduling, principally on two machines, where the "jobs" and hence also the processing durations, are defined in advance, and where the entire job constitutes the transfer batch. Examples include papers by Matsuo (1987) and Roundy (1988). These problems can be viewed as special cases of our problem in which the processing times are negligible, so only the defined-duration setups need to be scheduled with the restriction that the setups for a given product cannot occur concurrently on the two machines. Alternatively, we can

obtain a very similar restricted problem by specifying the cycle duration, hence also the processing time per batch, and assume that the setup times are zero. (The latter version is not completely equivalent since the resulting cycle duration is not guaranteed to be equal to the assumed duration.)

3. Problem Formulations

In this section, we develop two formulations of the two-machine problem. The first is more traditional and treats the processing of each part *on each machine* as a separate entity. The second formulation, on the other hand, views the scheduling problem from the perspective of parts, and treats the processing times associated with a particular part as a single entity.

We begin by formulating a simplified version of the problem for a rotation cycle where the sequence is fixed and the same sequence is used on both machines. A related formulation is given in El-Najdawi (1989), but ours differs because of the assumptions about transfer batches. The notation is defined below.

Data:

- i — index for the positions in the sequence. For this formulation, in which the sequence is fixed and the same sequence is used on both machines, i will also index the part.
- j — index for the machines in the flow shop.
- n — number of positions in cycle.
- m — number of machines in flow shop.
- d_i — the rate of demand for part i.
- p_{ij} — the rate of production of part i on machine j.
- ρ_{ij} — $= d_i/p_{ij}$, the utilization of j by part i.
- s_{ij} — setup time of part i on machine j.
- h_{ij} — the weight or holding cost on part i for inventory after machine j. Inventory "after machine 0" is raw material.

Variables:

t_{ij} processing duration of part i on machine j.

u_{ij} idle time on machine j before setup of part i.

f_{ij} finishing time of part i on machine j.

T cycle length.

δ_{ij} $= (t_{ij+1} - t_{ij})^+$, i.e., the amount by which the finishing time of part i on machine j must exceed the finishing time of part i on machine j-1

v_{ij} delay in processing of part i on machine j beyond the earliest feasible time, given f_{ij-1}; also, the additional time, beyond δ_{ij}, by which the finishing time of part i on machine j+1 exceeds that of part i on machine j.

We now define the constraints that must be satisfied. We start with the constraints on the finishing times:

$$f_{ij} = f_{ij-1} + v_{ij-1} + \delta_{ij} \qquad \forall\ i,j \qquad (1)$$

$$f_{ij} = f_{i-1j} + u_{ij} + s_{ij} + t_{ij} \qquad \forall\ i,j \qquad (2)$$

$$f_{0j} = f_{nj} - T \qquad \forall\ j \qquad (3)$$

$$u_{ij},\ v_{ij} \geq 0 \qquad \forall\ i,j. \qquad (4)$$

Constraints (1) force the finishing time of part i on machine j to be no earlier than the finishing time of part i on machine j - 1. If $\delta_{ij} > 0$ then machine j processes part i more slowly than machine j - 1. Adding δ_{ij} to the right hand side of (1) ensures that the starting time of i on machine j - 1 precedes the starting time on machine j. The variable v_{ij} is a slack variable which reflects the delay of the start (finish) of processing of part i on machine j in comparison to its earliest feasible starting (finishing) time. Thus, v_{ij}, in conjunction with the processing rates for part i at machines j and j-1, defines the extent of the work-in-process buildup of part i between the two machines. Constraints (2) ensure that for each machine j there is sufficient time to setup and produce part i after part i - 1 is completed. The length of the cycle is set to T for every machine by constraints (3). To define the production times as a function of the cycle duration, we require that

$$t_{ij} = \rho_{ij} T.$$

If this were a one-machine flow shop then the minimum cycle length would be

$$T_{min} = \frac{\sum_{i=1}^{n} s_{i1}}{1 - \sum_{i=1}^{n} \rho_{i1}}.$$

Because this is an m-machine flow shop, the minimum cycle length is

$$T_{min} = \max_{j} \left\{ \frac{\sum_{i=1}^{n} s_{ij}}{1 - \sum_{i=1}^{n} \rho_{ij}} \right\}.$$

Next we derive the objective. The finished goods inventory cost is

$$\frac{T}{2} \sum_{i=1}^{n} h_{im} d_i (1 - \rho_{im}).$$

If the raw materials inventory is delivered as needed then there would be no such inventory. If the raw materials arrive at a constant rate, d_i, for part i, then the cost for it will be

$$\frac{T}{2} \sum_{i=1}^{n} h_{i0} d_i (1 - \rho_{i0}).$$

The two previous formulas are standard expressions. We now derive expressions for the amount of WIP after machine j. To simplify the notation we consider the case of WIP between machines 1 and 2 and suppress the subscript for the part i. There are three cases. The first case, where the processing rate on machine 2 is slower than that on machine 1, and where the processing on machine 2 is delayed by an amount v_1, is pictured in Figure 1.

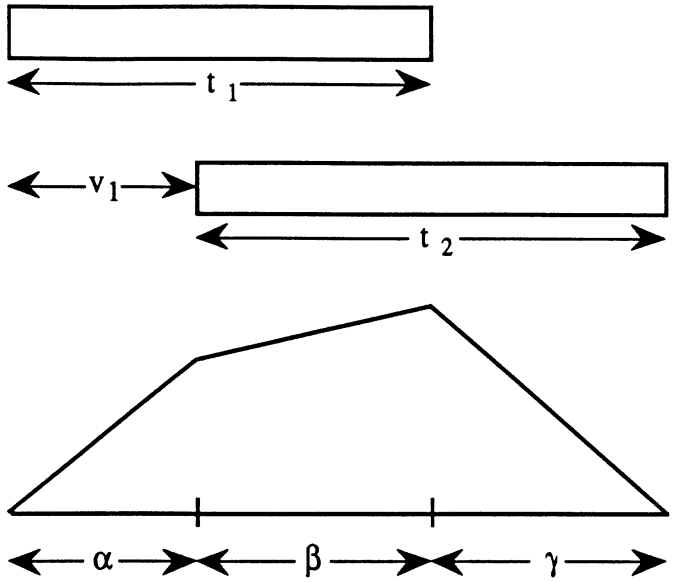


Figure 1: Example of WIP Inventory

The area of the trapezoid, I, in Figure 1 is:

$$I = \frac{1}{2}\alpha^2 p_1 + \frac{1}{2}\gamma^2 p_2 + \frac{1}{2}(\alpha p_1 + \gamma p_2)\beta$$

$$= \frac{1}{2}(\alpha p_1(\alpha + \beta) + \gamma p_2(\gamma + \beta))$$

where

$$\alpha = v_1$$

$$\beta = t_1 - v_1$$

$$\gamma = t_2 + v_1 - t_1 .$$

This reduces to

$$I = \frac{1}{2}(v_1 (t_1 p_1 + t_2 p_2) + (t_2 - t_1) t_2 p_2).$$

To obtain the average WIP for this item over the cycle, we divide by T and substitute $t_i = \rho_i T$ to obtain

$$\frac{I}{T} = v_1 d + \frac{T}{2} d (\rho_2 - \rho_1).$$

Similar derivations (see Appendix 2) for the other two cases give us the formula for any case as

$$\frac{I}{T} = v_1 d + \frac{T}{2} d\,|\rho_2 - \rho_1|.$$

The average cost per unit time of the WIP is thus:

$$\sum_{j=1}^{m-1} \sum_{i=1}^{n} h_{ij} \left(v_{ij} d_i + \frac{T}{2} d_i\,|\rho_{j+1} - \rho_j|\right).$$

We can now formulate the optimization problem for a fixed sequence. To emphasize the dependence on T, we eliminate the t_{ij} variables by substituting $\rho_{ij} T = t_{ij}$. We can simplify the objective by defining $\rho_{i0} = \rho_{im+1} = 1$ for all i. The optimization problem is a linear program:

(P1)
$$\text{Min} \quad \left(\frac{T}{2}\right)\left(\sum_{j=0}^{m} \sum_{i=1}^{n} h_{ij} d_i\,|\rho_{ij+1} - \rho_{ij}|\right) + \sum_{j=1}^{m-1} \sum_{i=1}^{n} h_{ij} d_i v_{ij} \qquad (5)$$

subject to
$$f_{ij} = f_{ij-1} + T(\rho_{ij} - \rho_{ij-1})^+ + v_{ij-1} \qquad \forall\ i,j \qquad (6)$$

$$f_{ij} = f_{i-1j} + s_{ij} + T\rho_{ij} + u_{ij} \qquad \forall\ i,j \qquad (7)$$

$$f_{0j} = f_{nj} - T \qquad \forall\ j \qquad (8)$$

$$f_{ij},\ u_{ij},\ v_{ij} \geq 0. \qquad \forall\ i,j \qquad (9)$$

Problem (P1) is a very constrained version of the problem. Not only is the sequence specified, but the constraints (5) require that the units processed on machine j-1 in $[f_{0j-1}, f_{0j-1} + T]$ be processed on machine j in $[f_{0j}, f_{0j} + T]$. There are, however, instances where allowing earlier processing of parts (i.e., in the previous cycle) on machine j-1 might be advantageous.

Consider the following example. There are two machines j = 1, 2 and seven parts indexed i = 1,...,7. All parts have the same demand rate of 1 unit per 18 time units. Parts 1, 2, 3, 4, and 5 are equally expensive to hold in WIP and parts 6 and 7 have a positive but negligible WIP holding cost, ε. All of the parts have extremely large finished goods holding costs, i.e., large enough so

that it is optimal to produce only one unit at a time. The setup times and processing times (per unit) of the parts are given in Table 1.

Table 1: Example Data

i	s_{i1}	s_{i2}	p_{i1}^{-1}	p_{i2}^{-1}
1	2	0	2	2
2	2	2	0	1
3	1	1	1	1
4	3	0	1	1
5	1	1	0	3
6	1	1	1	1
7	1	1	2	2

Assuming that only one unit of each part is processed in each cycle, the total processing time required on each machine is 18 units on machine 1 and 17 units on machine 2. Thus, we can satisfy demand exactly with no finished goods inventory. Suppose now that the sequence of parts on machine 1 is {1, 2, 6, 3, 4, 5, 7} and the sequence on machine 2 is {6,1,2,3,7,4,5}. With the current formulation, the resulting schedule is as shown in Figure 2. In the figure, s indicates a setup, t indicates processing, and the numerical index denotes the part.

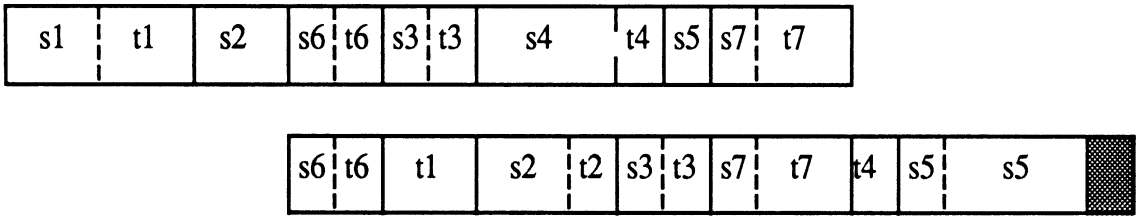


Figure 2: Gantt Chart with Constraints (6)

This schedule has inventory holding costs of $(31/18)h_{\bullet 1}$ per unit time.

On the other hand, by allowing parts 6 and 7 processed on machine 1 in one cycle to be processed on machine 2 in the next cycle, we could have obtained the schedule shown in Figure 3.

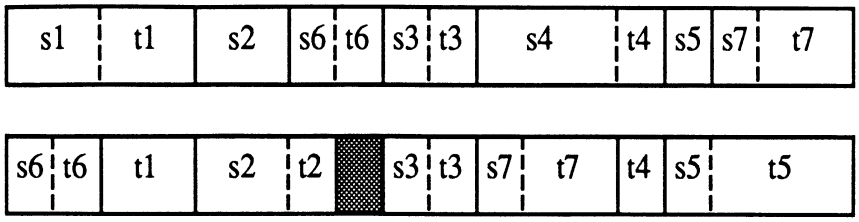


Figure 3: Gantt Chart with Parts 6 and 7 Wrapped

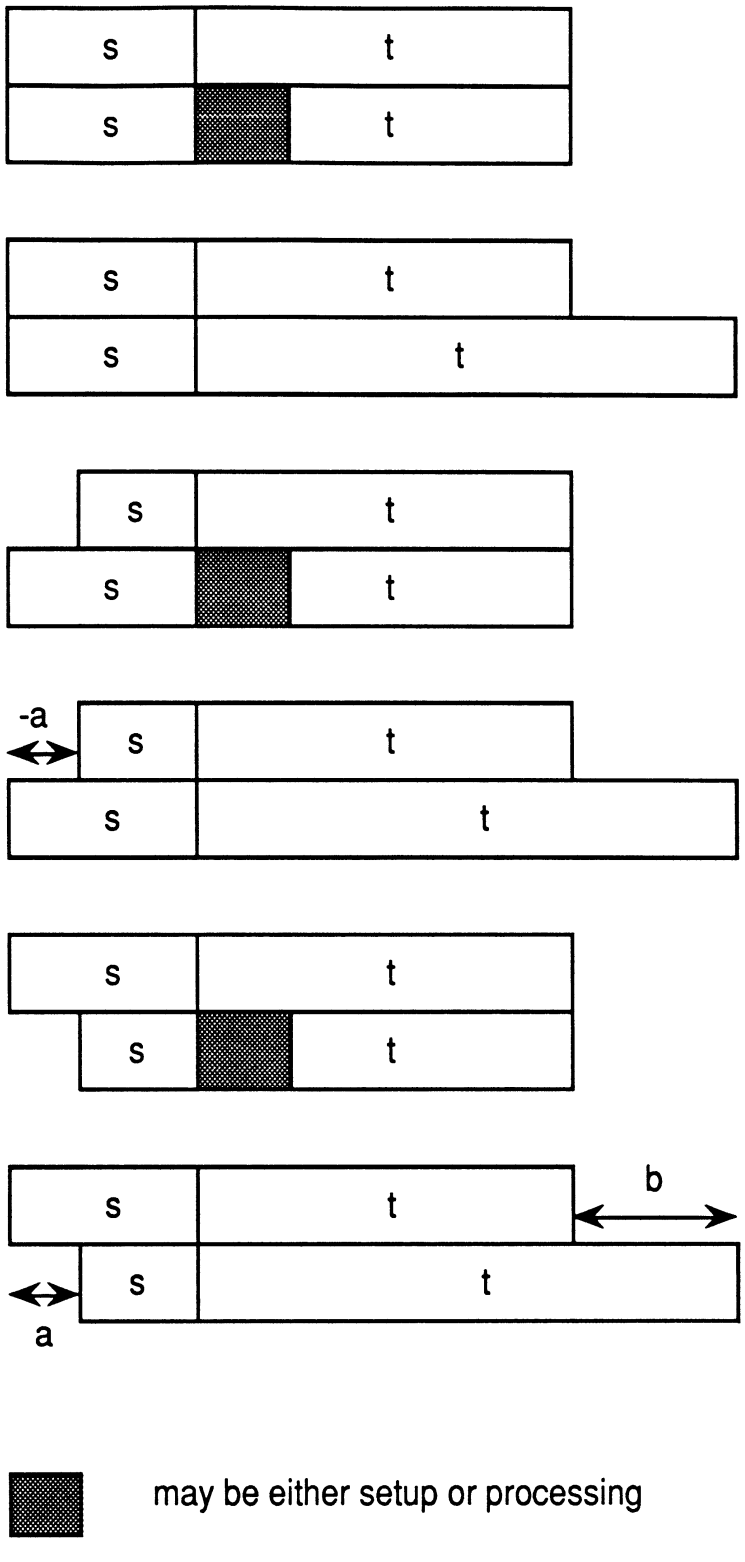
We say that parts so scheduled are "wrapped," since their production schedules wrap around from one cycle to the next. Note that this alternate schedule has negligible inventory holding costs. It turns out, however, that the decision of whether to allow a part to "wrap" is a binary decision, which adds another set of combinatorial decisions, above and beyond the sequencing issue. To accommodate wrapping, constraint (6) would be replaced by

$$f_{ij} = f_{ij-1} + T(\rho_{ij} - \rho_{ij-1})^+ + v_{ij-1} - Tx_{ij-1} \qquad \forall\ i,j \qquad (6')$$

where $x_{ij-1} = 1$ if part i wraps between machines j-1 and j, 0 otherwise. Note that when wrapping is allowed, the optimal schedules and objective values can differ considerably depending upon which items are "wrapped," even when the same sequence is used on both machines.

Although "wrapping" may provide a lower cost than a solution with no wrapping, it appeared to us that such a solution might be difficult to administer in practice. For instance, in flow lines with automated material handling between machines, it would be necessary to set aside the WIP of the wrapped items for later use. This could require considerable storage space, as well as extra equipment and/or labor to set aside, then later retrieve, the WIP. The same problem occurs if different sequences are used on the two machines. For this reason, we decided to confine our development of heuristic procedures to "unwrapped" permutation schedules. Later in the paper, we provide worst case error bounds for schedules of this type, as well as empirical results on their performance relative to the optimal solution.

Observe that for any given value of T, the two-machine Gantt chart for a given job with minimum delay between machines can take on one of only six different "shapes," which are shown



may be either setup or processing

Figure 4: Six Different "Shapes" when $v_{ij} = 0$

in Figure 4. Note that in the "shapes" with simultaneous completion of processing on the two machines, the setup on machine 2 may conclude strictly later than the setup on machine 1 if the processing rate on the second machine is higher than on the first. Upon further observation, we realized that, for fixed T, each shape could be defined by two parameters: a = difference between start of setup on machine 1 and start of setup on machine 2, and b = difference between completion of processing on machine 1 and completion of processing on machine 2.

Note also that delaying the processing on machine 2 (i.e., increasing the variable v) effectively allows us to change the "shape" at a cost. Thus, if T were given, the problem becomes one of deciding whether and to what extent to alter each of the shapes, and simultaneously selecting a permutation sequence, with the constraint that the overall cycle duration is equal to T.

Further analysis allowed us to express both a and b as functions of T and the input data, as follows:

$$a_{ij} = (s_{ij} - s_{ij+1}) + T(\rho_{ij} - \rho_{ij+1})^+ \text{ and}$$

$$b_{ij} = T(\rho_{ij} - \rho_{ij+1})^- \qquad \text{for all i and j.}$$

All of these relations arise from the basic precedence constraints among the various setups and processing intervals. This leads to the second formulation for a given sequence σ, which is the same on all machines:

(P2) Min $$T \cdot K + \sum_{i=1}^{n}\sum_{j=1}^{m-1} h_{ij}d_i v_{ij} \qquad (7)$$

subject to $$a_{\sigma(i+1)j} + v_{\sigma(i+1)j} + u_{\sigma(i)j} = b_{\sigma(i)j} + v_{\sigma(i)j} + u_{\sigma(i)j+1} \quad \text{for } i=1,\ldots,n;\ j=1,\ldots,m-1, \quad (8)$$

$$a_{ij} = (s_{ij} - s_{ij+1}) + T(\rho_{ij} - \rho_{ij+1})^+ \quad \text{for } i=1,\ldots,n;\ j=1,\ldots,m-1, \quad (9)$$

$$b_{ij} = T(\rho_{ij} - \rho_{ij+1})^- \quad \text{for } i=1,\ldots,n;\ j=1,\ldots,m-1, \quad (10)$$

$$(1 - \sum_{i=1}^{n}\rho_{ij})T = \sum_{i=1}^{n} s_{ij} + \sum_{i=1}^{n} u_{ij} \quad \text{for } j=1,\ldots,m, \quad (11)$$

$$v_{ij} \geq 0 \quad \text{for } i=1,\ldots,n;\ j=1,\ldots,m-1, \quad (12)$$

$$u_{ij} \geq 0 \quad \text{for } i=1,\ldots,n;\ j=1,\ldots,m, \quad (13)$$

$$T \geq 0, \quad (14)$$

where $K=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=0}^{m} h_{ij}d_i \mid \rho_{ij} - \rho_{ij+1} \mid$.

For a fixed T this is a minimum cost network flow problem. The network for the 4-part, 2-machine problem with sequence {1,2,3,4} appears in Figure 5.

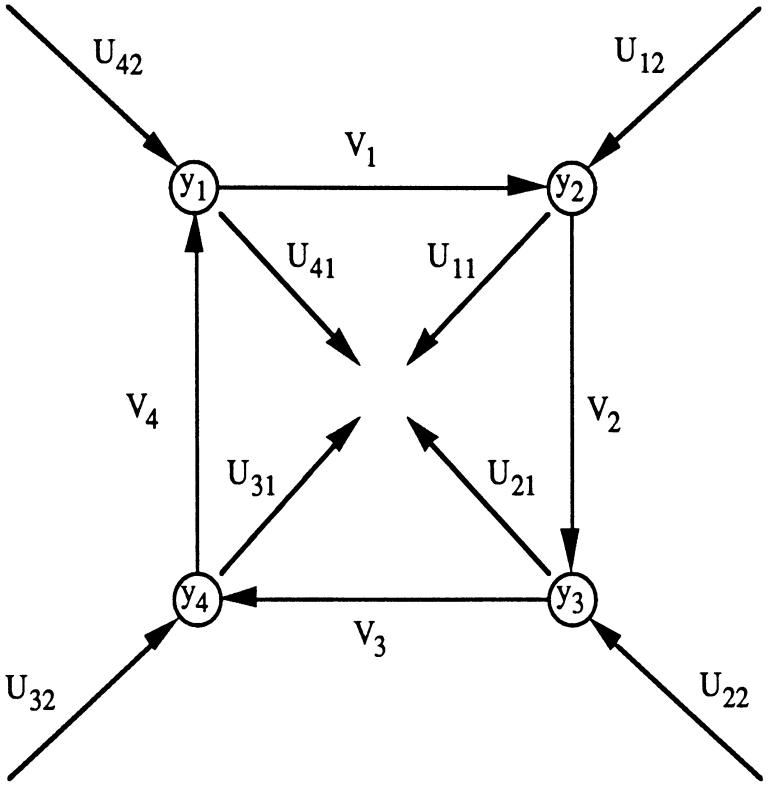


Figure 5: Network Flow Representation

The nodes can be viewed as demand nodes, where the demand, y_i, is equal to $a_i - b_{i-1}$. Thus, in some cases, y_i may be negative. The flows around the cycle are the v_{ij}'s, and the cost of each unit of flow on these arcs is $h_{ij}d_i$, respectively. The u's are the excess supply or demand at each node that are needed to make the problem feasible. Each additional machine adds another cycle around the perimeter of the network, and exterior inward arcs at the corners.

The formulation for a fixed set of sequences $\{\sigma_j\}_{j=1}^{m}$, one for each machine, requires that we modify constraint (8) to

$$a_{\sigma_j(i+1)j} + v_{\sigma_j(i+1)j} + u_{\sigma_j(i)j} = b_{\sigma_j(i)j} + v_{\sigma_j(i)j} + u_{\sigma_j(i)j+1} + \sum_{k\in B_{ij}} c_k \quad i=1,\ldots,n \;\; j=1,\ldots,m-1 \qquad (8')$$

where $c_k = s_{kj+1} + T\rho_{kj+1} + u_{kj+1}$. To clarify this, consider two items in sequence on machine j, $\sigma_j(i)$ and $\sigma_j(i+1)$. For a given pair of items, $\sigma_j(i)$ and $\sigma_j(i+1)$, on machine j, we find the locations of the same items on machine j+1. Define B_{ij} as the set of items that appear between these two in the sequence on machine j+1. For Figure 6, we have $B_{ij} = \{x_1,\ldots,x_k\}$.

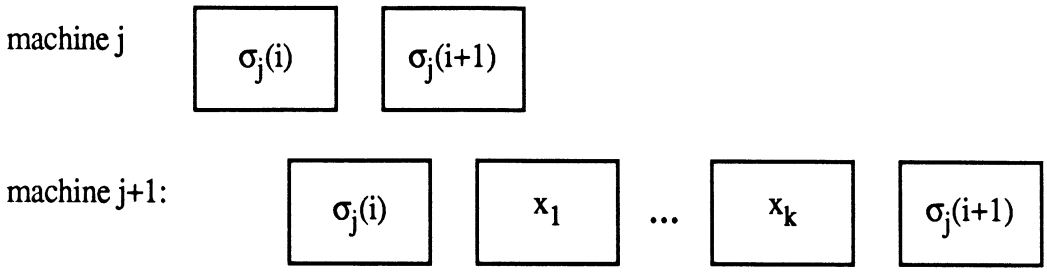


Figure 6: Illustration of B_{ij}

For each item that is "wrapped," we must subtract T from the corresponding v_{ij} terms. In particular if item i wraps between machines j and j+1 then replace v_{ij} by v_{ij}-T wherever it appears in the right hand side of constraints (2′). Let $x_{ij} = 1$ if item i wraps between machines j and j+1. The constraint (2′) is now

$$a_{\sigma_j(i+1)j} + v_{\sigma_j(i+1)j} + u_{\sigma_j(i)j} = b_{\sigma_j(i)j} + v_{\sigma_j(i)j} - Tx_{\sigma_j(i)j+1} + u_{\sigma_j(i)j+1} + \sum_{k\in B_{ij}} c_k.$$

We now have a general formulation of the problem. We should note that in each formulation, it is easy to add constraints on the v_{ij}'s to reflect WIP storage limitations.

4. Solution Properties and Conjectures

In this section we discuss a property of the optimal solution, and conjectures on properties of good sequences and the optimal cycle duration.

Proposition 1. In the optimal solution, $\min_i \{v_{ij}\} = 0$ for all j.

Proposition 1 states that there is at least one part on each machine which has zero delay. We sketch the proof as follows. If the proposition were not true, it would be possible to modify the

schedules on machines j, j+1,...,m so that each event begins $\min_i \{v_{ij}\}$ earlier. The schedule is still feasible, and the WIP inventory is reduced between machines j and j-1. This property, while intuitively appealing, is not very useful in developing heuristic solutions.

One of the difficulties in developing a heuristic sequencing procedure for our problem is not knowing in advance what the value of T^* will be, and therefore not knowing the processing times. In the example problem, we knew that $T^* = T_{min}$ because of economic considerations, but it is not clear that such a relationship holds in general. To investigate this issue, we generated and solved 54 two-machine problems. We chose to examine problems with the same, arbitrarily selected, sequence on both machines and no wrapping, since allowing different sequences and wrapping would have relaxed some precedence constraints, making it easier to obtain a more compressed schedule. These problems were constructed by hand with the intent of ensuring significant differences among the problems. In 52 out of 54 cases, we found that $T^* = T_{min}$. In the remaining 2 cases T^* was within 2% of T_{min}. On reflection, this result is not surprising. Note from (1) that increasing T above T_{min} increases finished goods and cycle WIP inventory proportionally for all parts. On the other hand, it is unlikely to substantially decrease the delay WIP for more than a few parts. On the basis of these preliminary results, we conjecture that T^* is either equal to, or very close to, T_{min}.

We also conjectured that using the same sequence on all machines would provide good results in most instances with realistic costs. Although this conjecture is based largely on intuition, we observed that the solutions for the two-machine problems mentioned earlier had relatively little delay WIP inventory. Moreover, the processing delays for the individual parts can be selected (by the LP) to minimize the total inventory cost. Thus, the main factors in obtaining a good solution appeared to be selecting a sequence that would permit a schedule with $T = T_{min}$, then secondarily avoiding long processing delays between machines.

5. Heuristics and Enumerations Procedures

This section describes the various procedures used in the computational work, including the two proposed heuristics for generating sequences, a routine for evaluating permutation schedules and a routine for computing optimal solutions for 2-machine problems. We describe the two enumeration procedures first and then the two heuristics.

We define a permutation schedule as a cyclic schedule with the same sequence for every machine and without any "wrapping." Because sequences are cyclic, if there are n parts, we need to examine only (n–1)! different sequences. Thus the procedure to enumerate permutation schedules first fixes part n as the last part, then generates the (n–1)! different sequences for the first n–1 parts and evaluates each sequence via the LP formulation (P2). We collected statistics on the best and the worst permutation sequence.

Finding the optimal solution, even for a 2-machine problem, is substantially more difficult than determining the best permutation schedule. For m machines, one must consider all possible sequences on each the m machines. Furthermore one must account for all possible "wraps". For each machine, there are (n–1)! permutation sequences. For m machines there are $\left((n-1)!\right)^m$ combinations of sequences. The set of all wraps appears to generate 2^n possibilities for each pair of sequences on adjacent machines. Each possibility can be represented as a 0–1 n-vector in which the ith element is 1 if part i wraps in the solution. However, not all of these possibilities are distinct. For example, the solution with wrap vector (0,...,0), i.e., no parts wrap, and the solution with wrap vector (1,...,1), i.e., all parts wrap, are identical solutions. Thus for a m-machine n-part problem there are $((n-1)!)^m(2^n - 1)^{m-1}$ possibilities. Clearly some of these are dominated and can be eliminated without solving the LP. As an example, consider the sequence 1,2,3,4 on machine 1, the sequence 1,3,2,4 on machine 2 and the wrap vector (0,1,0,0). This is illustrated in Figure 7.

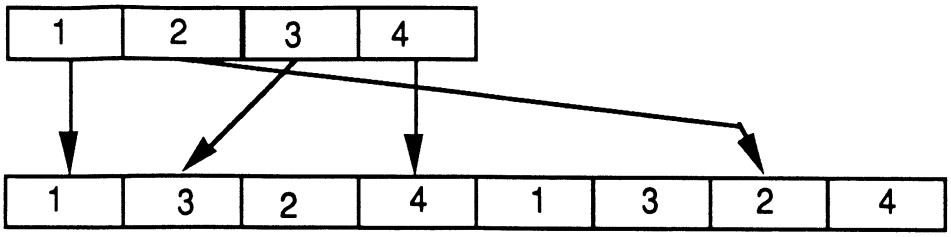


Figure 7 : Example of a solution which is dominated

Notice that since part 3 is not wrapped, the first lot of part 2 on machine 2 must be completed after the lot of part 2 on machine 1. In this situation it is not logical to wrap part 2 and connect the lot of part 2 with the second lot of part 2 on machine 2 as is shown in the diagram. As the alternatives are enumerated, those with clearly dominated wrap decisions are eliminated and the corresponding LPs are not solved.

We now describe two heuristic procedures, both of which were motivated by the conjectures in Section 4. Both deal with the "jigsaw puzzle" aspect of the sequencing problem rather than accounting for the relative economic factors. That is, the heuristics concentrate on finding a sequence which minimizes either the cycle length or the processing delays, and they do not consider the relative values of $h_i d_i$ in determining the sequence. This was based on our intuition that for most problems, keeping $T = T_{min}$ would lead to good solutions, and the remaining costs would be minimized by reducing the total processing delay.

The first heuristic applies only to a 2-machine problem. It starts by computing $\{a_i\}_{i=1}^n$ and $\{b_i\}_{i=1}^n$. It picks the largest b, say b_i and then it picks the largest a, say a_k that corresponds to a different part, $k \neq i$. The partial sequence (i,k) is formed. Parts i and k are deleted, and they are replaced by a new part with $a = a_i + \max(0, a_k - b_i)$, $b = b_k + \max(0, b_i - a_k)$. In other words, the schedules for parts i and k are merged to form a new part with no idle time between the parts. This is accomplished by either shifting part i's schedule on machine 1 earlier if $b_i < a_k$ or shifting part k's schedule on machine 1 later if $a_k < b_i$. The process is repeated until there is only a single part. The resulting sequence is then scheduled and evaluated using the LP. Since this heuristic concentrates on reducing idle time, we anticipated that it would perform well when the utilization

levels of the two machines are similar, and hence reducing idle time on both machines would be advantageous.

The second heuristic takes a different approach. Rather than myopically matching the parts that appear to "fit" together, this heuristic approximates the cost of placing k after i by measuring the idle time that would be created on the bottleneck machine if no avoidable production delays between machines were allowed (i.e., $v_{ij} = 0$ for all i and j). This time is defined as c_{ik}. The heuristic then finds the sequence which minimizes the total idle time added to the schedule of the bottleneck machine. It does this by solving the (asymmetric) travelling salesman problem (TSP) defined by $\{c_{ik}\}$.

The c_{ik}'s are computed as follows. Suppose that machine J is the bottleneck, and observe that for the sequence (i,k), machine J–1 may cause idle time on machine J if $a_{kJ-1} > b_{iJ-1}$ and the amount will be $a_{kJ-1} - b_{iJ-1}$. Similarly machine J–2 may cause idle time if $a_{kJ-2} + a_{kJ-1} > b_{iJ-2} + b_{iJ-1}$. The latter situation is shown in Figure 8.

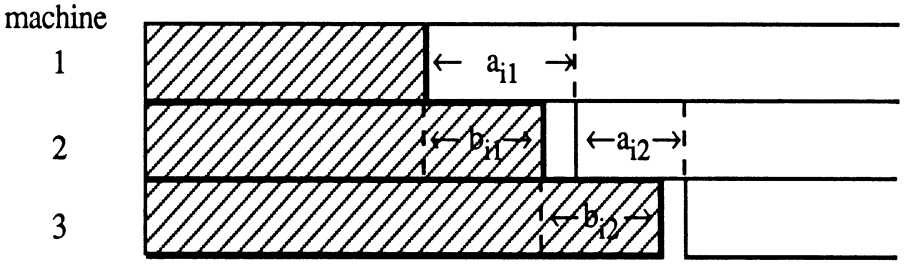


Figure 8: Computation of c_{ik} for Heuristic 2.

In general, the amount of idle time between parts i and k on machine J caused by machines $j < J$ is

$$c_{ik}^{\text{before}} \equiv \max_{j<J} \left\{ \left(\sum_{l=j}^{J-1} (a_{kl} - b_{il}) \right)^{+} \right\}.$$

An analogous argument shows that the amount of idle time between i and k on machine J caused by machines $j > J$ is

$$c_{ik}^{after} \equiv \underset{j>J}{Max} \left\{ \left(\sum_{l=J}^{i-1} (b_{il} - a_{kl}) \right)^{+} \right\}.$$

Thus

$$c_{ik} \equiv \max\left\{c_{ik}^{before}, c_{ik}^{after}\right\}.$$

One advantage of this heuristic over the previous one is that it is defined for problems with more than 2 machines, whereas the first heuristic does not have a natural extension to a problem with more than 2 machines. A second advantage is that it provides a more global assessment of whether two pieces fit together. The first heuristic is rather myopic in this respect. The disadvantage, of course, is that it requires solving a TSP, but for a small number of parts (≤12) this does not present a significant computational burden. In some special circumstances it is guaranteed to find the optimal solution. An example of such a case is given in the next proposition.

Proposition 2. Suppose machine j is the bottleneck machine and that $a_{ij} \geq 0$ for all i. Suppose further that heuristic 2 finds a solution to the TSP with value 0. Then, that sequence is optimal.

Proof: By definition of the TSP, the sequence found by the heuristic can be executed without any shifting (v = 0) with a cycle length of T_{min} + objective value of the TSP. Since the value of the TSP solution is 0 by hypothesis, and v = 0 this yields a solution with objective value equal to KT_{min}. Since this is a known lower bound to the problem the solution must be optimal. ■

Before closing this section, we develop worst-case error bounds for permutation schedules in instances where they are not provably optimal.

Proposition 3. For an arbitrary sequence σ, let z_{UB} be the value of solution obtained by using this sequence on every machine, setting the cycle length $T=T_{min}$ and optimizing. Then

(i) if $h_{ij} = h_i \; \forall_j$, $z_{UB} \leq mTK$; and

(ii) if h_{ij} is increasing in j, $z_{UB} \leq (2m-1)\,TK$.

Since TK is a lower bound on the optimal value, z*, then

(i) if $h_{ij} = h_i \; \forall_j$, $\frac{z_{UB}}{z^*} \leq m$, and

(ii) if h_{ij} is increasing in j, $\frac{z_{UB}}{z^*} \leq 2m\text{-}1$.

Proof: First we show that the largest value any v_{ij} can obtain is $T(1 - \rho_{ij+1})$. Consider a schedule (a portion of a cycle) where the facility produces every part but i and then it produces part i. Let the starting time for the production of part i on machine j be time 0. The claim is that the finishing time of part i on machine j+1 is at most T, and thus the starting time is at most $T(1 - \rho_{ij+1})$. Suppose that we schedule item i on machine j+1 from $T(1 - \rho_{ij+1})$ to T. This leaves from time 0 to time $T(1 - \rho_{ij+1})$ to produce the remaining parts and execute all the setups, including the setup for i. This is clearly feasible since every part $k \neq i$ has completed on machine j by time 0.

For case 1 we need to show that

$$\sum_{j=1}^{m-1} \sum_{i} h_i d_i v_{ij} \leq (m-1)TK \tag{9}$$

since then we have that

$$z_{UB} \leq TK + (m-1)TK = mTK = mz_{LB}.$$

To see this replace v_{ij} by $T(1-\rho_{ij}) \leq T(1-\min_j \{\rho_{ij}\})$.

$$\text{Thus} \sum_{j=0}^{m-1} \sum_{i} h_i d_i v_{ij} \leq T(m-1) \sum_{i} h_i d_i (1- \min_j \{\rho_{ij}\}). \tag{10}$$

On the other hand, if $k = \text{argmin}_j \rho_{ij}$, then

$$\sum_{j=0}^{m} |\rho_{ij+1} - \rho_{ij}| \geq 2(1-\rho_{ik})$$

since

$$\sum_{j=0}^{k-1} |\rho_{ij+1} - \rho_{ij}| \geq \left| \sum_{j=0}^{k-1} (\rho_{ij+1} - \rho_{ij}) \right| = (1 - \rho_{ik})$$

and

$$\sum_{j=k}^{m} |\rho_{ij+1} - \rho_{ij}| \geq \left| \sum_{j=k}^{m} (\rho_{ij+1} - \rho_{ij}) \right| = (1- \rho_{ik}).$$

Thus $$K \geq \sum_i h_i d_i (1 - \min_j \{\rho_{ij}\}) \qquad (11)$$

Inequalities (10) and (11) give us (9).

For case 2,

$$2K = \sum_i \sum_{j=0}^{m} h_{ij} d_i \mid \rho_{ij+1} - \rho_{ij} \mid$$

$$\geq \sum_i \sum_{j=k}^{m} h_{ij} d_i \mid \rho_{ij+1} - \rho_{ij} \mid$$

$$\geq \sum_i h_{ik} d_i \sum_{j=k}^{m} \mid \rho_{ij+1} - \rho_{ij} \mid \quad \text{since } h_{ij} \text{ is increasing in } j$$

$$\geq \sum_i h_{ik} d_i (1 - \rho_{ik}).$$

Thus

$$\sum_{j=1}^{m-1} \sum_i h_{ij} d_i v_{ij} \leq T \sum_{j=1}^{m-1} \sum_i h_{ij} d_i (1 - \rho_{ij})$$

$$\leq T \sum_{j=1}^{m-1} 2K$$

$$\leq 2KT\,(m-1).$$

So we have

$$z_{UB} = TK + \sum_{j=1}^{m-1} \sum_i h_{ij} d_i v_{ij}$$

$$\leq TK + TK\,2(m-1)$$

$$\leq TK(2m-1)$$

$$\leq (2m-1)\, z_{UB}.$$

For m = 2 the bounds become

$$z_{UB} \leq 2 z_{LB} \text{ and } z_{UB} \leq 3 z_{LB}.$$

6. Computational Experiments and Results

There were three purposes of the computational experiments. First, we wanted to determine whether the heuristics could obtain optimal or near optimal solutions by using the same sequence on all machines. Second, we were interested in determining how important the sequence was in generating good solutions. Third we wanted to evaluate the two heuristics suggested in Section 5 by comparing their objective values to either the optimal solution value or a lower bound.

Generation of Random Problems

Nine parameters were used to generate random problems for the experiments. They are:

n	the number of parts,
m	the number of machines,
h_{max}	the maximum holding cost,
s_{max}	the maximum setup cost,
ρ_{min}	the minimum utilization,
ρ_{max}	the maximum utilization,
h_{equal}	=true if the holding costs for a part were equal across the stages of production,
s_{equal}	=true if the setup times for a part were equal across the machines,
T_{equal}	=true if the cycle times of the machines are equal.

Given the parameters, a problem was randomly generated by computing

$h_{ij} = U[1,h_{max}]$ for i=1,...,n and j=0,...,m,

$s_{ij} = U[1,s_{max}]$ for i=1,...,n and j=1,...,m,

$\rho_{ij} = U[\rho_{min}/n, \rho_{max}/n]$ for i=1,...,n and j=1,...,m,

where U[a,b] is a random variable with a uniform distribution on [a,b]. We set d_i=1 for i=1,...,n without loss of generality. Given this, the ρ_{ij}'s determine the production rates, $\{p_{ij}\}$.

If h_{equal} then $h_{ij} = h_{i0}$ for j=1,...,m.

If s_{equal} then $s_{ij} = s_{i1}$ for j=2,...,m.

If T_{equal} then the ρ_{ij}'s were scaled so that $T_j = \left(\sum_i s_{ij}\right) / \left(1 - \sum_i \rho_{ij}\right) =$ T_1 for j=2,...,m. The required scale factor ω_j satisfies

$$T_1 = T_j' \equiv \frac{\sum_i s_{ij}}{1 - \omega_j \sum_i \rho_{ij}}.$$

Solving for ω_j we obtain

$$\omega_j = \frac{\left(1 - \frac{\sum_i s_{ij}}{T_1}\right)}{\sum_i \rho_{ij}}.$$

With ω_j known, set $\rho_{ij}' = \rho_{ij}\omega_j$, for i=1,...,n j=2,...,m.

The Experiments

We use the following notation to represent the solution values for the various procedures:

z^{**} the optimal objective value.

z^{*} the objective value of the best permutation schedule.

$z^{\#\#}$ the worst objective value from the sequence and wrap enumeration given that the LP (P2) is solved for the best T and $\{v_{ij}\}$.

$z^{\#}$ the worst permutation schedule objective value in the same sense as $z^{\#\#}$ but the enumeration is only over permutation schedules with no wrapping.

z_1 the solution value from Heuristic 1.

z_2 the solution value from Heuristic 2.

z_{LB} = KT_{min}, a lower bound on z^{**}.

Determining if permutation schedules are near optimal.

The purpose of the first set of experiments was to determine whether allowing parts to wrap or allowing different sequences on each machine would provide a better solution than the best permutation schedule. The examples in Appendix 1 demonstrated that there are instances for which the optimal solution has a part that wraps or has a different sequence on machine 2. Thus the goal here is to demonstrate that for reasonable problems these considerations will not improve the solution significantly. Because the number of permutations and wraps that need to be enumerated grows exponentially, it was possible to consider only small problems. Ten problems with four parts and ten problems with five parts were generated. For each of the 5-part problems, $(4!)^2\, 2^5 = 18432$ sequence/wrap combinations were examined, and an LP was solved for non-dominated combinations. The parameters of the problems generated were ($m = 2$, $h_{max} = 5$, $s_{max} = 5$, $\rho_{min} = 0.5$, $\rho_{max} = 0.9$, h_{equal} = false, s_{equal} = false, T_{equal} = false). For all 20 problems $z^{**} = z^{*}$. Although, because of the small sample sizes and small problem sizes, we cannot guarantee that this would hold for all realistic problems, it did confirm our intuition that limiting our search to permutation schedules would be more than adequate in practice.

For these problems we also calculated $z^{\#\#}/z^{**}$. Summary statistics are reported in Table 2. The results suggest that the worst sequence is significantly worse than the best sequence.

Table 2: Ratio of solution values of worst sequence choice to best sequence choice.

Number of Parts	$z^{\#\#}/z^{**}$	
	mean	maximum
4	2.38	2.98
5	2.56	3.10

Evaluating the two heuristics

The second set of experiments considered a much broader set of problems. Since the first set confirmed that permutation schedules are likely to provide near-optimal solutions, the best permutation schedule (z^*) was used as a benchmark for these problems. The trials included problems with 4, 5 or 6 parts. We limited the size of the problems because of the number of alternatives that had to be evaluated via an LP, and because we wanted to consider many parameter combinations. For each problem the maximum setup time was either 5 or 50 and the maximum holding cost was either 5 or 50. Problems were generated for which the minimum cycle length for machine j, $T_j \equiv \left(\sum_i s_{ij}\right) / \left(1 - \sum_i \rho_{ij}\right)$ was equal to T_{min} and problems were generated for which $T_j > T_{min}$ for every machine j that was not the bottleneck. For each category, 10 problems were generated for a total 240 problems.

Table 3 presents the results for the cases with unequal minimum cycle times, i.e., one machine was more of a bottleneck than the other. Table 4 presents the same results when the cycle times for the two machines were forced to be equal. The results are quite striking. For the first set, the heuristic objective value only occasionally exceeded 1% and rarely exceeded 2% of the objective value of the best permutation schedule. The average performance was well within 1%. The second set of problems was more difficult as we anticipated. We expected these problems to be more difficult since if $T = T_{min}$ there can be no idle time on either machine. This implies more production delays between machines, i.e., higher v_{ij} values. Nonetheless, even here the worst performance by either heuristic across all 120 problems was within 10% of the value of the best permutation schedule. The average performance in each category never exceeded 2% above that of the best permutation schedule.

Table 3: Experiments with 2 machines, 4, 5 or 6 parts and unequal cycle lengths

Number of Parts	Maximum Setup Time	Maximum Holding Cost	z_1/z^* Mean	z_1/z^* Maximum	z_2/z^* Mean	z_2/z^* Maximum	$z_1<z_2$
4	5	5	1	1.0004	1.0026	1.0255	10%
		50	1.0006	1.0044	1.0003	1.0020	10%
	50	5	1.0002	1.0010	1	1	0%
		50	1.0001	1.0013	1.0061	1.0516	20%
5	5	5	1.0004	1.0027	1.0015	1.0119	10%
		50	1.0006	1.0027	1.0012	1.0063	20%
	50	5	1.0013	1.0081	1.0042	1.0159	30%
		50	1.0006	1.0043	1.0035	1.0117	40%
6	5	5	1.0015	1.0095	1.0032	1.0119	50%
		50	1.0004	1.0022	1.0024	1.0090	40%
	50	5	1.0016	1.0079	1.0086	1.0440	50%
		50	1.0040	1.0192	1.0026	1.0154	20%

Table 4: Experiments with 2 machines, 4, 5 or 6 parts and equal cycle lengths

Number of Parts	Maximum Setup Time	Maximum Holding Cost	z_1/z^* Mean	z_1/z^* Maximum	z_2/z^* Mean	z_2/z^* Maximum	$z_1<z_2$
4	5	5	1.0020	1.0178	1.0040	1.0258	40%
		50	1.0067	1.0196	1.0091	1.0408	40%
	50	5	1.0093	1.0511	1.0149	1.0581	50%
		50	1.0233	1.0995	1.0163	1.0861	40%
5	5	5	1.0040	1.0268	1.0032	1.0172	20%
		50	1.0069	1.0191	1.0059	1.0207	50%
	50	5	1.0067	1.0523	1.0175	1.0658	80%
		50	1.0159	1.0732	1.0174	1.0562	50%
6	5	5	1.0032	1.0112	1.0111	1.0388	80%
		50	1.0016	1.0064	1.0087	1.0357	70%
	50	5	1.0101	1.0548	1.0114	1.0265	50%
		50	1.0219	1.1080	1.0208	1.0637	40%

A summary of the results appears in Table 5. Each line in Table 5 averages the results over the 8 categories for the given number of parts. These results are somewhat less surprising when viewed in light of the $z^{\#}/z^{*}$ statistic, i.e., the ratio of the objective value of the worst permutation schedule to the objective value of the best permutation schedule (details not shown here). This value averaged about 1.04 and its maximum value over 240 problems was 1.15. Thus we can conclude that in absolute terms, the heuristics finds near-optimal solutions, but for permutation schedules, the sequence does not have a large impact on the solution value. One reason for this is that the unavoidable inventory costs due to the rotation cycle account for a majority of the costs.

Table 5: Results for problems with 2 machines and 4, 5 or 6 parts, summarized over all parameter settings.

	z_1/z^*		z_2/z^*		$z_1<z_2$
Number of Parts	Mean	Maximum	Mean	Maximum	
4	1.0053	1.0995	1.0067	1.0861	26.25%
5	1.0046	1.0732	1.0068	1.0658	37.5%
6	1.0056	1.1080	1.0086	1.0637	50%

Because $z^{\#}$ was generally very close to z^{*} we tabulated the statistic $\frac{z_i - z^{*}}{z^{\#} - z^{*}}$ which gives the percentage by which Heuristic i's objective value exceeded the best permutation schedule relative only to the range between $z^{\#}$ and z^{*}. Table 6 summarizes these results.

Table 6: Performance of two heuristics relative to range of possible values.

Number of Parts	$\frac{z_1-z^*}{z^\#-z^*}$ mean	$\frac{z_2-z^*}{z^\#-z^*}$ mean
4	20%	25%
5	12%	15%
6	8%	15%

The results reported above correspond to the parameter settings h_{equal} = false, s_{equal} = false. We actually generated problems for all possible parameter settings which resulted in a total of 960 problems. Generally, the results for these other problems (not reported here) were even better. We also anticipated this since the other parameter settings would generate problems with less diversity.

Experiments with more parts and more than two machines.

The purpose of the next two experiments was to determine how well the heuristics performed on problems with more parts and problems with more machines. The same combinations of parameter settings were used. For the first of these experiments, the number of parts was either 8, 10 or 12. The only difference here is that the lower bound (z_{LB}) is used as the benchmark. For the problems with 4, 5 and 6 parts, the gap between z^* and z_{LB} was small. The value of z^*/z_{LB} was 1.02 on average and its maximum over the 240 problems was 1.11. Thus, as the results summarized in Tables 7 and 8 indicate, these problems become easier with more parts. The average gap between z_i for i=1,2 and z_{LB} was under 1.2% for problems where the minimum cycle lengths differed and under 5% for problems where the cycle lengths were equal.

For the last experiment the number of machines was increased to either 3 or 5. For these problems only Heuristic 2 could be applied. For the trials with 4, 5 or 6 parts, the benchmark was

Table 7: Experiments with 2 machines, 8, 10 or 12 parts and unequal cycle lengths

Number of Parts	Maximum Setup Time	Maximum Holding Cost	z_1/z_{LB}		z_2/z_{LB}		$z_1<z_2$
			Mean	Maximum	Mean	Maximum	
8	5	5	1.0047	1.0156	1.0052	1.0194	50%
		50	1.0037	1.0094	1.0037	1.0101	30%
	50	5	1.0065	1.0142	1.0092	1.0204	40%
		50	1.0056	1.0334	1.0053	1.0306	10%
10	5	5	1.0055	1.0131	1.0064	1.0177	60%
		50	1.0047	1.0109	1.0057	1.0114	40%
	50	5	1.0087	1.0201	1.0091	1.0175	60%
		50	1.0083	1.0154	1.012	1.0293	70%
12	5	5	1.0070	1.0177	1.009	1.0392	40%
		50	1.0040	1.0125	1.0051	1.0138	70%
	50	5	1.0039	1.0071	1.0061	1.0152	60%
		50	1.0063	1.0148	1.0084	1.0290	50%

Table 8: Experiments with 2 machines, 8, 10 or 12 parts and equal cycle lengths.

Number of Parts	Maximum Setup Time	Maximum Holding Cost	z_1/z_{LB} Mean	z_1/z_{LB} Maximum	z_2/z_{LB} Mean	z_2/z_{LB} Maximum	$z_1<z_2$
8	5	5	1.0163	1.0351	1.0215	1.0459	90%
		50	1.0278	1.0577	1.0283	1.0491	70%
	50	5	1.0194	1.0536	1.0316	1.0613	90%
		50	1.0305	1.0658	1.0312	1.0527	70%
10	5	5	1.0215	1.0307	1.0277	1.0359	80%
		50	1.0193	1.0399	1.0228	1.0449	80%
	50	5	1.0193	1.0291	1.0281	1.0554	90%
		50	1.0326	1.1081	1.0467	1.1075	90%
12	5	5	1.0155	1.0311	1.0212	1.0503	70%
		50	1.0162	1.0284	1.0202	1.0389	60%
	50	5	1.0233	1.0478	1.0293	1.0603	70%
		50	1.0285	1.1005	1.0362	1.1089	80%

the lower bound, $z_{LB} = T_{min}K$. The results appear in Tables 9 and 10. Again, the results are quite good. The average gap between the heuristic solution and the best permutation schedule was about 1%. For the problems with a larger number of parts, the 3-machine problems had an average gap, relative to the lower bound, of 2.5%. For the 5-machine problems the average gap was under 7%.

Table 9: Experiments with either 3 or 5 machines and 4, 5 or 6 parts.

Number of Machines	Number of Parts	z_2/z^*	
		Mean	Maximum
3	4	1.0104	1.102
	5	1.0113	1.0388
	6	1.0138	1.0937
5	4	1.0099	1.0468
	5	1.0155	1.0653
	6	1.0131	1.0572

Table 10: Experiments with either 3 or 5 machines and 8, 10 or 12 parts

Number of Machines	Number of Parts	z_2/z_{LB}	
		Mean	Maximum
3	8	1.0263	1.086
	10	1.0286	1.0895
	12	1.0262	1.0612
5	8	1.0678	1.2364
	10	1.0732	1.2339
	12	1.0653	1.1579

In summary the experiments have demonstrated that the controllable WIP rarely adds significantly to the cost of a solution. Thus greater cost savings are achievable if via some engineering choice one can make both the s_{ij}'s and ρ_{ij}'s more similar across machines for each i, and by reducing the setup times in absolute terms. Yet, for a given situation the heuristics presented here perform quite well in minimizing the controllable WIP cost across a broad range of problems.

7. Summary and Discussion

In this paper we have investigated a very general version of the problem of finding a cyclic, pure rotation schedule for a multi-machine flow shop to minimize total inventory holding costs. One major distinction between our problem and those investigated before is that we allow parts produced on a machine in one cycle to be processed by the subsequent machine in the next cycle. We developed a formulation of the problem that led us to conclude that (a) minimizing the overall cycle duration is important, and (b) using the same sequence on all machines is likely to produce good solutions. On this basis, we developed two heuristic procedures. One applies to only two-machine problems, and focuses on minimizing the cycle duration. The other can be applied to any number of machines and is based upon an approximate representation of our problem as a travelling salesman problem. We also developed worst-case error bounds for these heuristics.

We solved a large number of problems, both optimally and using the heuristic procedures. The results indicate that the optimal cycle duration is equal to or very close to the minimum cycle duration. In addition, permutation sequences (same sequence on all machines) perform quite well in comparison to all possible sequences, and worst permutation sequence is not substantially worse than the best. Consequently, the heuristics produce optimal or very near optimal solutions.

We are currently investigating extensions of this model to consider non-serial production systems and more general (non-rotation) sequences.

Acknowledgements

We wish to thank Karla E. Bourland for introducing us to this problem.

This research was partially supported by a gift from the Ford Motor Company to the University of Michigan and by the Center for Manufacturing and Operations Management, William E. Simon Graduate School of Business Administration, University of Rochester

References

Dobson, G. (1987), "The Economic Lot Scheduling Problem: Achieving Feasibility Using Time-Varying Lot Sizes," *Operations Research*, **35** (5), 764-771.

El-Najdawi, M.K. (1989), Common Cycle Approach to Lot-Size Scheduling for Multistage, Multiproduct Production Processes, Unpublished Ph.D. Dissertation, Wharton School, University of Pennsylvania, Philadelphia, PA.

Gallego, G. (1990), "An Extension to the Class of Easy Economic Lot Scheduling Problems," *IIE Transactions*, **22** (2), 189–190.

Hanssmann, F. (1962), *Operations Research in Production and Inventory Control*, Wiley, New York.

Jones, P.C. and R.R. Inman (1989), "When is the Economic Lot Scheduling Problem Easy?", *IIE Transactions*, **21** (1), 11-20.

Matsuo, H. (1987), "Cyclic Scheduling Problems in the Two-Machine Flow Shop: Complexity, Worst-Case and Average Case Analysis," Working Paper 87-12-4, Graduate School of Business, University of Texas at Austin, Austin, TX.

Roundy, R. (1988), "Cyclic Schedules for Job Shops with Identical Jobs," Technical Report #766, School of OR and IE, Cornell University, Ithaca, NY.

Singh, D. (1990), "Using CFM as a Competitive Edge," *Automation*, **37**, 64-5.

Appendix 1

In this section we present two examples. The first is one in which $T>T_{min}$ for the optimal solution. The second example has one part wrapped in the optimal solution. The following data is common to both examples. There are two parts on two machines. The setups and utilizations are given in Figure A1.1.

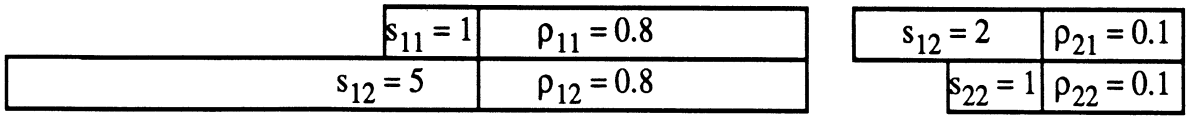


Figure A1.1: Gantt chart for two parts without shifting or wrapping.

The minimum cycle lengths for the two machines are

$$T_1 = \frac{1+2}{1-0.8-0.1} = 30 \quad \text{and} \quad T_2 = \frac{5+1}{1-0.8-0.1} = 60$$

so machine 2 is the bottleneck. Let $d_1 = d_2 = 1$, $h_{10} = h_{11} = h_{12} = 1$, and $h_{20} = h_{21} = h_{22} = \phi$. Since there are only two parts there is only one sequence to consider, (1,2). there are three possible solutions corresponding to three wrap vectors, (0,0), (1,0), (0,1). Note that wrap vector (1,1) yields the same solution as (0,0). The objective value for this problem is

$$\begin{aligned} &0.5T\,[\,(h_{10}d_1(1-0.8) + h_{11}d_1(0.8-0.8) + h_{12}d_1(1-0.8) \\ &\quad + h_{20}d_2(1-0.1) + h_{21}d_2(0.1-0.1) + h_{22}d_2(1-0.1)] \\ &\quad + h_{11}d_1v_1 + h_{21}d_2v_2 \\ &= T(0.2 + \phi(0.9)) + v_1 + \phi v_2 \end{aligned}$$

Let's consider the objective value of the three possible solutions. The first solution corresponds to the case for which production of neither part has been delayed, i.e. $v_1 = v_2 = 0$. In this case $T = 61$ and the objective value is $61(0.2) + \phi(61)(0.9) = 12.2 + 54.9\phi$. The second solution is one in which the production of part 1 on machine 1 is done early in the cycle so $v_1=1$ but $T = T_{min} = 60$. The objective value is $60(0.2) + 60(0.9)\phi+1 = 13 + 54\phi$. The third solution

is the one in which the production of part 2 on machine 1 is delayed so that part 2 is "wrapped". In this case T = 60, $v_1 = 0$ but $v_2 = 56$. The objective value is $60(0.2) + 60(0.9) + 56\phi = 12 + 100\phi$. It is easy to verify that the third solution is the minimum for $\phi \in [0, \frac{2}{451}]$. This provides an example where wrapping a part is optimal. The first solution is the minimum for $\phi \in [\frac{2}{451}, \frac{8}{9}]$ and this provides the example where $T > T_{min}$ in the optimal solution. The second solution is the minimum for $\phi \in [\frac{8}{9}, \infty)$.

Appendix 2

This appendix contains derivations of average WIP for the four cases that were not presented in Section 2.

Case 1: t_1, t_2 overlap and $t_2 \geq t_1$

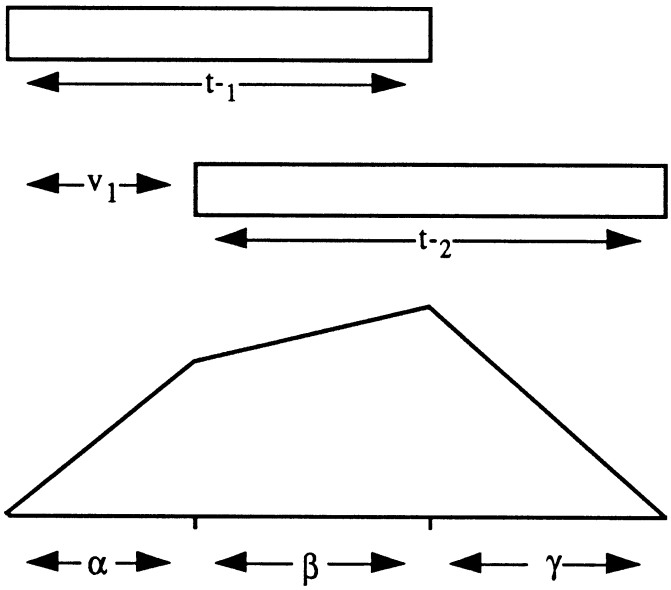


Figure A2.1: Inventory for Case 1

$$I = \frac{1}{2}\alpha^2 p_1 + \frac{1}{2}\gamma^2 p_2 + \frac{1}{2}(\alpha p_1 + \gamma p_2)\beta$$

$$= \frac{1}{2}(\alpha p_1(\alpha + \beta) + \gamma p_2(\gamma + \beta))$$

where

$$\alpha = v_1$$
$$\beta = t_1 - v_1$$
$$\gamma = t_2 + v_1 - t_1$$

Thus

$$I = \frac{1}{2}\,(v_1 p_1(t_1) + (t_2 + v_1 - t_1)p_2\,(t_2))$$
$$= \frac{1}{2} v_1\,(t_1 p_1 + t_2 p_2) + \frac{1}{2} p_2 t_2\,(t_2 - t_1)$$
$$\frac{I}{T} = v_1 d + \frac{T}{2} d\,(\rho_2 - \rho_1).$$

Case 2: t_1, t_2 overlap and $t_2 \leq t_1$

The inventory diagram is similar except in the middle section (β) the inventory decreases rather than increases.

$$I = \frac{1}{2}\,(\alpha p_1(\alpha + \beta) + \gamma p_2(\gamma + \beta))$$

where

$$\alpha = t_1 + v_1 - t_2$$
$$\beta = t_2 - v_1$$
$$\gamma = v_1$$

Thus

$$I = \frac{1}{2}\,((t_1 + v_1 - t_2)p_1\,(t_1) + v_1 p_2\,(t_2))$$
$$= \frac{1}{2} v_1\,(t_1 p_1 + t_2 p_2) + \frac{1}{2}\,(t_1 - t_2)\,p_1 t_1$$
$$\frac{I}{T} = v_1 d + \frac{T}{2} d\,(\rho_1 - \rho_2).$$

Case 3: t_1 and t_2 do not overlap and $t_1 < t_2$

Note $p_1 t_1 = p_2 t_2$.

$$I = \frac{1}{2}\alpha^2 p_1 + \frac{1}{2}\gamma^2 p_2 + \frac{1}{2}(\gamma p_1 + \gamma p_2)\beta$$

$$= \frac{1}{2}(\alpha p_1 (\alpha + \beta) + \gamma p_2 (\gamma + \beta))$$

where

$\alpha = t_1$

$\beta = -a$

$\gamma = t_2$

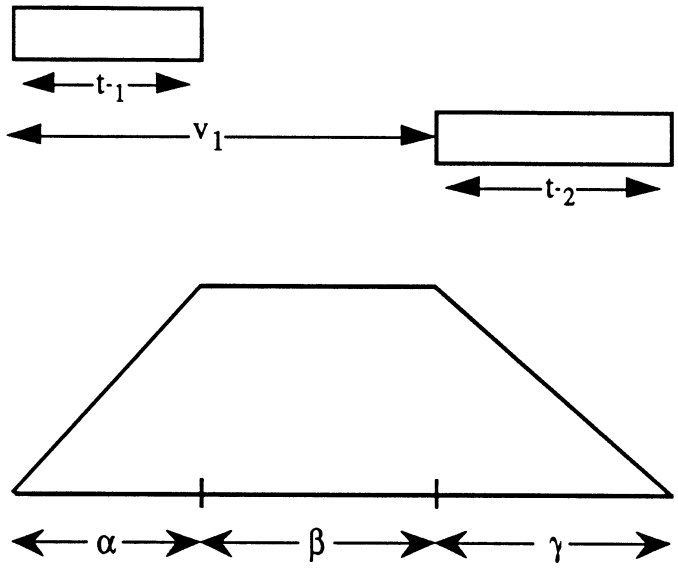


Figure A2.2: Inventory for Case 3

$$I = \frac{1}{2}(t_1 p_1 v_1 + t_2 p_2 (t_2 + v_1 - t_1))$$

$$= \frac{1}{2} v_1 (t_1 p_1 + t_2 p_2) + \frac{1}{2} t_2 p_2 (t_2 - t_1)$$

and so forth.

Case 4: t_1 and t_2 do not overlap and $t_1 \geq t_2$

Same as case 3 except $\beta = v_1 - \chi$.

$$I = \frac{1}{2}(t_1 p_1 (t_1 + v_1 - t_2) + t_2 p_2 v_1)$$

and so forth.