European Journal of Operational Research 75 (1994) 441–461
North-Holland

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, USA

Candace Arai Yano

Department of Industrial & Operations Engineering, University of Michigan, Ann Arbor, MI 48109, USA

(Received October 1990; revised March 1993)

Abstract: This paper addresses the problem of determining a cyclic schedule for batch production on a flow line. We assume a constant supply of raw materials and a constant demand for all finished goods. Material that 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 sequences, one for each machine. Two heuristics are presented for finding near-optimal sequences: one is applicable to the special case of a two-machine flow line; the others are applicable to an m-machine line and focus on different aspects of the problem (e.g., cycle stock or work-in-process inventory). From a computational study, we have observed that: 1) permutation schedules, i.e., schedules with the 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. (The parts needs not be processed in the same sequence on all machines, however.) 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 that 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

Correspondence to: Prof. G. Dobson, William E. Simon Graduate School of Business Administration, University of Rochester, Rochester, NY 14627, USA.

SSDI 0377-2217(93)E0206-D

average cost per unit time of the sum of finished goods, WIP, and raw materials inventory. By setting all inventory holding costs equal to 1, we can also minimize average inventory (in units), and by setting the finished goods inventory cost to zero, we can minimize average WIP inventory.

Setup costs also can be included. However, just as in single-machine cyclic scheduling problems, the introduction of setup costs leads to longer cycle durations, and consequently more idle time in the schedule. As will be apparent, later, in our problem additional idle time makes scheduling easier. This is one reason why we chose to ignore setup costs. Another reason is that, in many instances, out-of-pocket setup costs are negligible, and the manufacturing facility may use setup cost as a surrogate for setup time. Since we explicitly include setup times, such surrogate costs need not be included.

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 manufacturing systems in which inter-machine buffer space has been reduced, presumably to force inventory reductions, and thus necessitating more careful scheduling.

The first portion of this paper deals with the case of two machines, but most of the proposed heuristics are applicable to larger numbers of machines. 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 that 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 (to be selected) 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. The objective is to minimize the average cost per unit time of finished goods, WIP, and raw materials inventory. Raw material arrives at a constant rate to the first processing step. (Other assumptions regarding raw material arrival can be handled with minor modifications.) 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.

Jensen and Khan (1972) model a single-product version of our problem in which the cycle times are allowed to differ across stages. The objective is to minimize the average setup and inventory holding cost per unit time. They present a dynamic programming formulation and an associated solution procedure for this problem.

Taha and Skeith (1970) consider a variation of this problem in which backlogging is allowed, and lot sizes are constrained to be an integer multiple of the lot size at the successor stage. Szendrovits (1975) analyzes a special case of the problem in which the same lot size is used at all stages and no backlogging is allowed. In all three of these papers, setup times are assumed to be zero.

Hsu and El-Najdawi (1990) analyze a multi-product, multi-stage version of our problem in which the (pure rotation) production sequence is pre-specified, the cycle duration is the same at all stages and for all products, and the transfer batch size is assumed to be equal to the production batch size. Setup costs are included. For this version of the problem, the optimal cycle duration can be obtained in closed form.

El-Najdawi and Kleindorfer (1990) study a very similar problem to that of Hsu and El-Najdawi, with the only differences being in the assumptions about when the incremental holding cost associated with the value added for a given stage starts to be incurred. They show that the objective function, constrained to the optimal schedule for each value of the cycle duration, is convex in the cycle duration. They develop a solution procedure which combines a search (for the cycle duration) and a linear program to find the optimal schedule for a given cycle duration.

Unlike the Hsu and El-Najdawi and the El-Najdawi and Kleindorder papers, the focus of our paper is on the sequencing aspect of the problem. Note that even with a pure rotation (common cycle) schedule, one may choose to have different sequences on the various machines. This introduces some additional combinatorial issues that are ignored in the existing literature. We explain these issues by way of an example.

There are two machines $j = 1, 2$ and seven parts indexed $i = 1, \ldots, 7$. 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 we assume without loss of generality that the WIP holding cost for these parts is \$1.00 per unit per unit time. 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 time (s_{ij}) and processing time (p_{ij}) per unit of each the parts are given in Table 1.

Table 1
Example data

i	s_{i1} (hours)	s_{i2} (hours)	p_{i1}^{-1} (hours/unit)	p_{i2}^{-1} (hours/unit)	h_{i1} (\$/unit/hour)	h_{i2} (\$/unit/hour)	d_i (units/hour)
1	2	0	2	2	1	M [a]	1/18
2	2	2	0	1	1	M	1/18
3	1	1	1	1	1	M	1/18
4	3	0	1	1	1	M	1/18
5	1	1	0	3	1	M	1/18
6	1	1	1	1	ϵ [a]	M	1/18
7	1	1	2	2	ϵ	M	1/18

[a] ϵ is a very small number and M is a very large number.

Assuming that only one unit of each part is processed in each cycle, the sum of setup and 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}. If we require a job processed on machine 1 in a particular cycle to be processed on the second machine in the cycle with the same index, the resulting schedule is as shown in Figure 1. In the figure, s indicates a setup, t indicates processing, and the numerical index denotes the part. This schedule has inventory holding costs of 31/18 per unit time.

On the other hand, by processing parts 6 and 7 on machine 1 in one cycle, then on machine 2 in the next cycle, we could have obtained the schedule shown in Figure 2. 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. This example demonstrates the potential desirability of using different sequences on the various machines, and the benefits of considering wrapping. Little of the existing literature has considered these issues explicitly.

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).

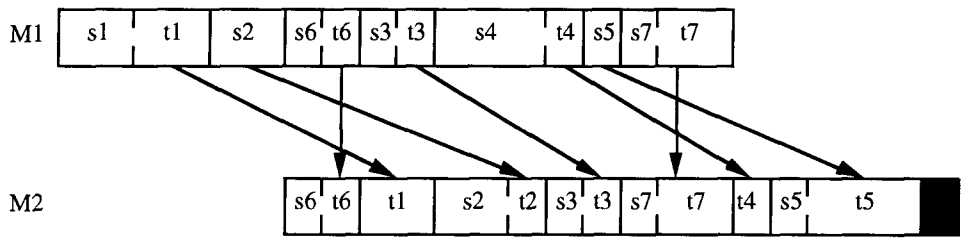

Fig. 1. Gantt chart with constraints on cycle indices (arrows indicate material transfers)

Jones and Inman (1989) have found that pure rotation schedules perform nearly as well as more complicated schedules in single-machine problems. It is useful to point out, however, that since all of the research on the single-machine problem 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 A.

Some recent work has been done on cyclic scheduling, where the 'jobs' and hence also the processing durations, are defined in advance, and where the entire job constitutes the transfer batch. These models do not incorporate setup times, and with only a few exceptions, the objective is to maximize throughput or simply to achieve a target throughput for a specified product mix. Examples include papers by Matsuo (1990), Roundy (1988), and McCormick et al. (1990). Our problem is more complicated because we must (i) specify the cycle duration (thus also the batch sizes); (ii) consider both setup times and processing times in specifying the detailed schedule; and (iii) consider schedule timing constraints that arise because of differences between the processing times of a part at two adjacent stages. (If the latter stage is faster, it is necessary to accumulate inventory between the stages and to delay the start of processing at the latter stage so as to avoid idle time in the middle of the production run at that stage.) Moreover, since our objective is to minimize the total inventory cost per unit time, the differences in the relative inventory holding cost rates will influence the quality of the sequence, whereas in throughput-based models, such relative weights need not be considered.

3. Problem formulations

In this section, we develop a two formulations of the 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

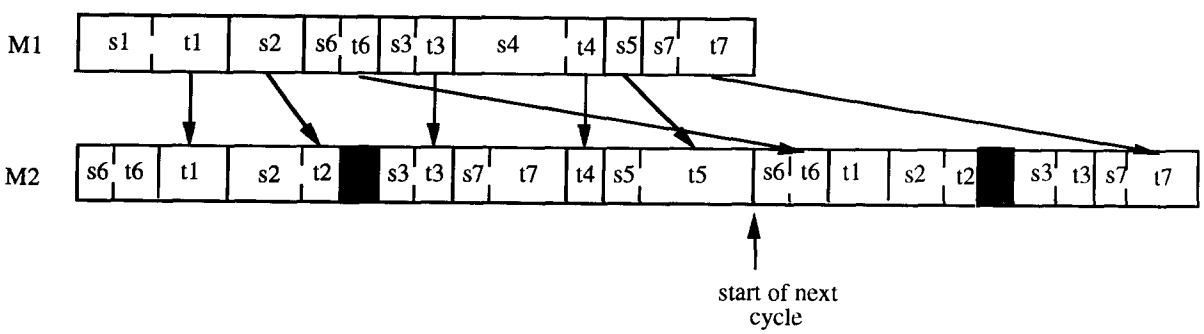

Fig. 2. Gantt chart with parts 6 and 7 'wrapped' (arrows indicate material transfers)

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 in Table 2.

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

$$f_{ij+1} = f_{ij} + v_{ij} + \delta_{ij} \quad \forall i, j, \tag{1}$$

$$f_{ij} = f_{i-1j} + u_{ij} + s_{ij} + t_{ij} \quad \forall i, j, \tag{2}$$

$$f_{0j} = f_{nj} - T \quad \forall j, \tag{3}$$

$$u_{ij}, v_{ij} \geq 0 \quad \forall i, j. \tag{4}$$

Constraints (1) ensure that the finishing time of part i on machine $j+1$ is no earlier than the finishing time of part i on machine j, accounting for the differences in the processing rates reflected in δ_{ij}. The variable v_{ij} reflects the delay of the start (finish) of processing of part i on machine $j+1$ 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

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

The minimum cycle length is

$$T_{\min} = \operatorname*{Max}_j \left\{ \sum_{i=1}^{n} s_{ij} / \left(1 - \sum_{i=1}^{n} \rho_{ij} \right) \right\}.$$

The expression in braces is the minimum cycle duration that ensures all setups and production can be completed on machine j. Since there are multiple machines, the minimum cycle duration for the entire system is the maximum of these values. We refer to the machine with the largest minimum cycle duration as the *bottleneck* machine.

Table 2
Notation

Data:	
i	Index for the positions in the sequence. For formulations in which the sequence is fixed and the same sequence is used on all 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 machine 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.
Decision variables:	
T	Cycle length.
f_{ij}	Finishing time of part i on machine j.
Auxiliary variables (implied by the decision variables):	
t_{ij}	$= Td_i / p_{ij}$; processing duration of part i on machine j.
δ_{ij}	$= (t_{ij} - t_{ij})^+$, i.e., minimum required delay (given T) between the finishing time of part i on machine $j+1$ and the finishing time of part i on machine j. This value is positive only when machine j is faster than machine $j+1$ in processing part i.
v_{ij}	Additional delay (beyond δ_{ij}) of the finishing time of part i on machine $j+1$.
u_{ij}	Idle time on machine j before setup of part i.

The derivation of finished goods and WIP inventory parallels that given in Jensen and Khan (1972). For the sake of completeness, we provide details in Appendix B.

We 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

$$\rho_{i0} = \rho_{im+1} = 1 \quad \text{for all } i.$$

The optimization problem is a linear program:

(P1)

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

subject to

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

$$f_{ij} = f_{i-1j} + u_{ij} + s_{ij} + T\rho_{ij} \quad \forall i, j, \tag{7}$$

$$f_{0j} = f_{nj} - T \quad \forall j,$$

$$f_{ij}, u_{ij}, v_{ij} \geq 0 \quad \forall i, j.$$

Problem (P1) is a very constrained version of the problem. Not only is the sequence specified, but the constraints (6) and (7) require that the units processed on machine j in $[f_{0j}, f_{0j} + T]$ be processed on machine $j+1$ in $[f_{0j+1}, f_{0j+1} + T]$. As mentioned earlier, there are instances where allowing earlier processing of parts (i.e., in the previous cycle) on machine j might be advantageous. 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

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

where $x_{ij} = 1$ if part i wraps between machines j and $j+1$, 0 otherwise. Note that when wrapping is allowed, the optimal schedules and objective values can differ considerably depending upon which parts 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 parts 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.

It is useful to mention that allowing wrapping may be very important in the context of cyclic schedules where the objective is to maximize throughput. Such wrapping allows one to reduce idle time that might otherwise be unavoidable. On the other hand, in our problem, where the objective is to minimize the total inventory holding cost per unit time and the demand rates are given, wrapping generally has adverse effects on WIP inventory with little, if any, offsetting benefits of reduced finished goods inventory. In Section 6, we report computational results that support this conjecture.

Observe that for any given value of T, the two-machine Gantt chart for a given job with minimum delay (hence also minimum WIP) between machines can take on one of only six different 'shapes', which are shown in Figure 3. 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. For fixed T, each shape can be defined by two parameters:

a = Difference between start of setup on machine 2 and start of setup on machine 1.

b = Difference between completion of processing on machine 2 and completion of processing on machine 1.

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

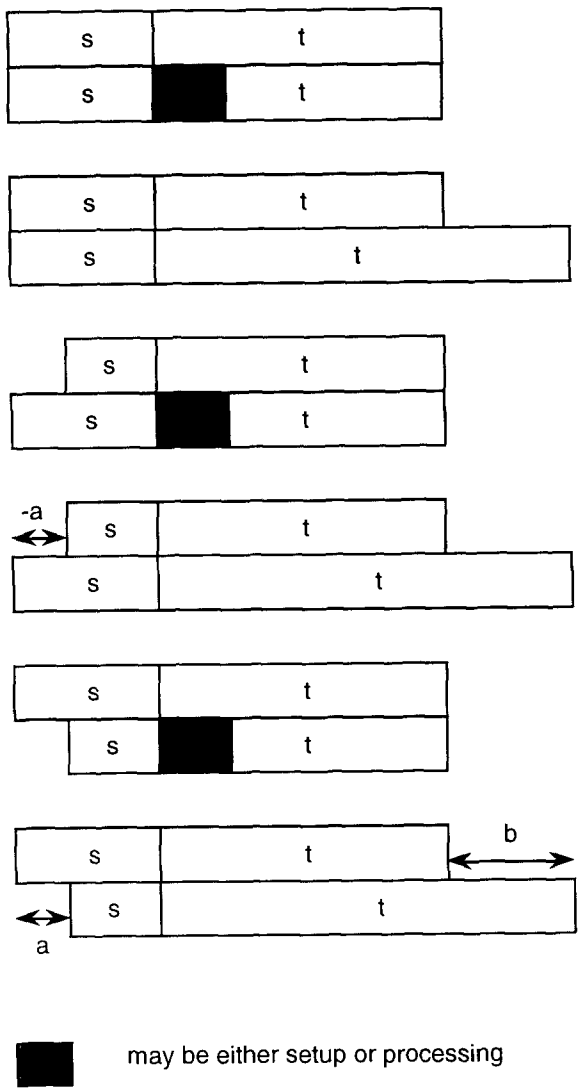

Fig. 3. Six different 'shapes' when $v_{ij} = 0$

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.

Although we have defined the 'shape' parameters only for the two-machine case, it should be evident that similar ideas can be used when there are three or more machines. Indeed, for the case of m machines; the shape of the zero-delay schedule can be defined with $2(m-1)$ parameters. We later explain how these ideas might be used in generalizing one of our heuristic procedures.

We can express both a and b as functions of T and the input data, as follows:

$a_{ij} = (s_{ij} - s_{ij+1}) + T(\rho_{ij} - \rho_{ij+1})^+$: i.e., the difference between the start of setup of part i on machine $j+1$ and the start of setup on machine j when i is scheduled so there is minimum delay between the two machines.

$b_{ij} = T(\rho_{ij+1} - \rho_{ij})^+$: i.e., the difference between the end of processing of part i on machine $j+1$ and the end of processing on machine j when i is scheduled so there is minimum delay between the two machines.

These relations arise from material flow constraints to ensure continuous processing of all units within a batch, and only require information on setups and processing intervals. This leads to the second formulation for a given sequence σ, which is the same on all machines:

(P2)

$$\text{Min } T \cdot H + \sum_{i=1}^{n} \sum_{j=1}^{m-1} h_{ij} d_i v_{ij}$$

subject to

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

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

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

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

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

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

$$T \geq 0, \tag{14}$$

where

$$H = \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=0}^{m} h_{ij} d_i \, | \rho_{ij} - \rho_{ij+1} |.$$

Constraints (8) ensure that time delays and idle times are properly accounted for at points where the schedules of two successive parts 'mesh' together. They play the same role as constraints (6) and (7) in the previous formulation. For a fixed T, (P2) is a highly structured linear program that can be solved easily.

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

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

where

$$c_k = s_{kj+1} + T\rho_{kj+1} + u_{kj+1}.$$

To clarify this, consider two consecutive parts on machine j, $\sigma_j(i)$ and $\sigma_j(i+1)$. For this pair of parts, we find the locations of the same parts on machine $j+1$. Define B_{ij} as the set of parts that appear between i and $i+1$ in the sequence on machine $j+1$. For Figure 5, we have $B_{ij} = \{x_1, \ldots, x_k\}$.

For each part that is 'wrapped', we must subtract T from the corresponding v_{ij} terms. In particular if part 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 part i wraps between machines j and $j+1$. The constraint (2') is now

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

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.

We end this section by deriving a lower bound on the optimal objective value, namely HT_{min}, which is needed in our worst-case analysis of the heuristics as well as the computational work. This is obtained by replacing T by T_{min} and setting $v_{ij} = 0$ for all i and j. This is clearly a lower bound for a given (set of) sequence(s) and since its value is independent of the (set of) sequence(s), it is a lower bound for all (sets of) sequences.

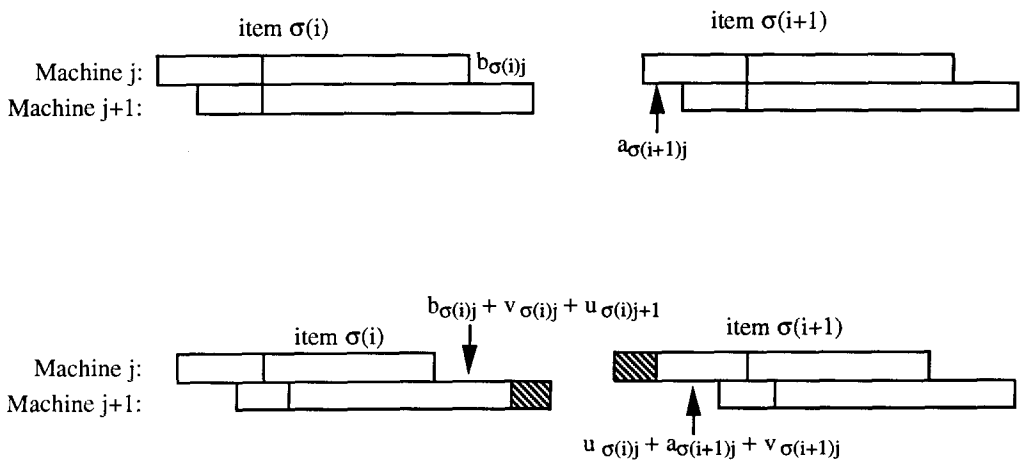

Fig. 4. Diagram for Equation (8) (shaded regions represent idle time)

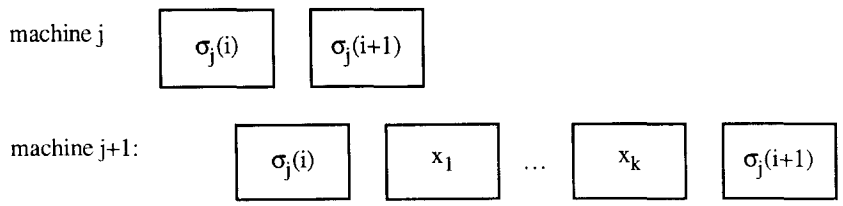

Fig. 5. Illustration of B_{ij}

4. Solution properties and conjectures

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

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 $j+1,\ldots,m$ 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 solution procedures, since it does not help to specify good sequences. One of the difficulties in developing a heuristic sequencing procedure for our problem is not knowing in advance what the optimal value of T, 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 two 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 (by allowing the schedules to 'fit together' better). 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 controllable WIP inventory. (Some of the WIP inventory arises because of differences in processing times across machines, and is therefore not controllable.) 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 enumeration procedures

This section describes the various procedures used in the computational study, including the two proposed heuristics for generating sequences, a routine for evaluating permutation schedules and a routine for computing optimal solutions for two-machine problems. We first describe the enumeration procedures for permutation schedules (same sequence on all machines and no wrapping) and for more general schedules.

Because permutation 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).

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 of 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 $((n-1)!)^m$ 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 i-th 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 $(0,\ldots,0)$, i.e., no parts wrap, and the solution with wrap vector $(1,\ldots,1)$, i.e., all parts wrap, are identical solutions. Thus for an m-machine n-part problem there are at most $((n-1)!)^m(2^n-1)^{m-1}$ 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 6.

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 schedule the lot of part 2 next to 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 $h_{ij}d_i$ 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. In Section 7, we describe some generalizations of these heuristics that incorporate the economic factors, but at the expense of significantly greater computation.

5.1. Heuristic 1

The first heuristic applies only to a two-machine problem. It starts by computing $\{a_i\}_{i=1}^n$ and $\{b_i\}_{i=1}^n$. It picks the largest b, say b_p and then it picks the largest a, say a_k, that corresponds to a different part, $k \neq p$. The partial sequence where p immediately precedes k is formed. Parts p and k are deleted, and they are replaced by a new part with

$$a = a_p + \max(0,\ a_k - b_p),$$

$$b = b_k + \max(0,\ b_p - a_k).$$

In other words, the schedules for parts p and k are merged to form a new part with no idle time between the parts. This is accomplished by either shifting part p's schedule on machine 1 earlier if

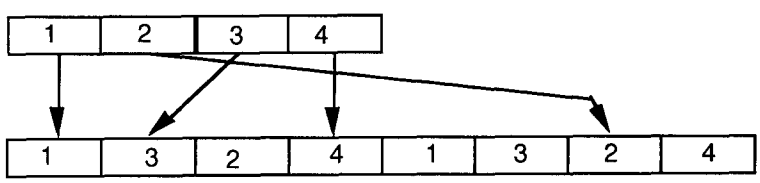

Fig. 6. Example of a dominated solution

$b_p < a_k$ or shifting part k's schedule on machine 1 later if $a_k < b_p$. 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 where reducing idle time on both machines would be advantageous.

5.2. Heuristic 2

The second heuristic takes a different approach. Rather than myopically matching the parts that appear to 'fit' together, we approximate the cost of placing k after p 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_{pk}. We then find the sequence that minimizes the total idle time added to the schedule of the bottleneck machine by solving the (asymmetric) travelling salesman problem (TSP) defined by $\{c_{pk}\}$.

The c_{pk}'s are computed as follows. Suppose that machine J is the bottleneck, and observe that the partial sequence in which p immediately precedes k on machine $J-1$ may cause idle time on machine J if $a_{kJ-1} > b_{pJ-1}$ and the amount will be $a_{kJ-1} - b_{pJ-1}$. Similarly machine $J-2$ may cause idle time if $a_{kJ-2} + a_{kJ-1} > b_{pJ-2} + b_{pJ-1}$. The latter situation is shown in Figure 7.

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

$$c_{pk}^{\text{before}} \equiv \underset{j<J}{\text{Max}} \left\{ \left(\sum_{l=j}^{J-1} (a_{kl} - b_{pl}) \right)^{+} \right\}.$$

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

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

Thus

$$c_{pk} \equiv \max\left\{c_{p\cdot}^{\text{before}}\ c_{pk}^{\text{after}}\right\}.$$

One advantage of this heuristic over the previous one is that it is defined for problems with more than two machines, whereas the first heuristic does not have a natural extension to a problem with more than two 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.

We note that this heuristic, while similar in spirit to that in McCormick et al. (1989), treats the problem with much greater accuracy. In the McCormick et al. paper, jobs are sequenced based on the total idle time incurred on all machines, not just the bottleneck. Our heuristic analyzes the effects of processing on other stations in computing the resultant idle time on the bottleneck. The McCormick et al. heuristic does not explicitly account for the cyclic nature of the schedule in determining the sequence, whereas ours does.

In some special circumstances the heuristic is guaranteed to find the optimal solution. An example of such a case is given in the next proposition.

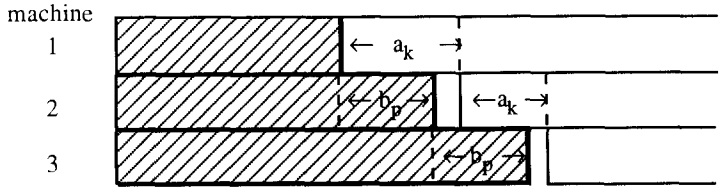

Fig. 7. Computation of c_{pk} for Heuristic 2

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 $\mathrm{HT}_{\min}$. Since this is a known lower bound to the problem the solution must be optimal. □

5.3. Worst case bounds for permutation schedules

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 the solution obtained by using this sequence on every machine, setting the cycle length* $T = T_{\min}$ *and optimizing. Then*

(i) *if* $h_{ij} = h_i$, $\forall j$,

$$z_{\mathrm{UB}} \leq m\mathrm{TH},$$

and

(ii) *if* h_{ij} *is increasing in j,*

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

Since TH *is a lower bound on the optimal value,* z^*, it follow that

(i) *if* $h_{ij} = h_i$, $\forall j$,

$$z_{\mathrm{UB}}/z' \leq m,$$

and

(ii) *if* h_{ij} *is increasing in j,*

$$z_{\mathrm{UB}}/z' \leq 2m - 1.$$

Proof. See Appendix C.

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 and without wrapping. 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.

6.1. Generation of random problems

Seven 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 time.
$\rho_{\min}$ = The minimum machine utilization.
$\rho_{\max}$ = The maximum machine utilization.
T_{equal} = True if the minimum cycle times of the machines are equal.

For the experiments described below, we decided to test the limits of the heuristics by generating problems that would be difficult to solve. In our problem, two particular characteristics generally make a problem difficult: (i) diversity of the items in terms of costs, setup times, and production rates, and (ii) equal utilization (percentage of time spent actually producing, exclusive of setups) across the machines. Diversity in terms of costs contributes to the difficulty of sequencing and wrapping decisions, while the other factors make it difficult to 'fit together' the schedules of the various parts without extending the cycle beyond the minimum cycle duration. We generated sets of problems with different levels of diversity of both the setup times and the holding costs, controlling these factors by setting the $h_{\max}$ and $s_{\max}$ parameters, respectively. For problems with equal utilizations across machines, we initially generated problem parameters without this constraint, then scaled the ρ_{ij}'s to make all utilizations equal to that of the bottleneck machine (as detailed below).

More formally, given the parameters, a problem was constructed as follows:

$$h_{ij} = U[1, h_{\max}]$$

for $i = 1, \ldots, n$, and $j = 0, \ldots, m$,

$s_{ij} = U[1, s_{\max}]$

for $i = 1,\ldots,n$, and $j = 1,\ldots,m$,

$\rho_{ij} = U[\rho_{\min}/n, \rho_{\max}/n]$

for $i = 1,\ldots,n$, and $j = 1,\ldots,m$,

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

We wanted the utilizations of the machines to be reasonably high, since problems with substantial idle time are easy to solve. Thus we generated each ρ_{ij} randomly between $\rho_{\min}/n$ and $\rho_{\max}/n$ so that the sum would be between $\rho_{\min}$ and $\rho_{\max}$.

If T_{equal}, i.e., the minimum cycle lengths were constrained to be equal, then the ρ_{ij}'s were scaled so that $T_j = \sum_i s_{ij}/(1 - \sum_i \rho_{ij}) = T_1$ for $j = 2,\ldots,m$. The required scale factor ω_j satisfies

$$T_1 = T_j' \equiv \sum_i s_{ij} \Big/ \left(1 - \omega_j \sum_i \rho_{ij}\right).$$

Solving for ω_j we obtain

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

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

6.2. The experiments

All of the computations were performed using Think Pascal Version 3 on a Macintosh II workstation. 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 $\{v_j\}$.

$z^{\#}$ = 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.

$z_{\text{LB}} = \text{HT}_{\min}$, a lower bound on z^{**}.

6.3. Performance of permutation schedules on two machines

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 four-part problems and ten five-part problems 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: $m = 2$, $h_{\max} = 5$, $s_{\max} = 5$, $\rho_{\min} = 0.5$, $\rho_{\max} = 0.9$, and $T_{\text{equal}} = \text{false}$. For all 20 problems $z^{**} = z^{*}$. While we cannot guarantee that this would hold for all realistic problems, the results did confirm our intuition that limiting our search to permutation schedules would be more than adequate in practice.

We were also interested in ascertaining how well arbitrary sequences would perform. For the problems described above, we also calculated $z^{\#\#}/z^{**}$ to provide an indicator of the range of costs across all sequences. Summary statistics are reported in Table 3. The results suggest that the

Table 3
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

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

Number of parts	Maximum setup time	Maximum holding cost	z_1/z^*		z_2/z^*		$z_1 < z_2$ (%)
			Mean	Maximum	Mean	Maximum	
4	5	5	1.0000	1.0004	1.0026	1.0255	10
		50	1.0006	1.0044	1.0003	1.0020	10
	50	5	1.0002	1.0010	1.0000	1.0000	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

worst sequence is significantly worse than the best sequence.

6.4. 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, the maximum holding cost was either 5 or 50, and either the minimum cycle lengths were constrained to be equal or allowed to be unequal. For each combination of number of parts, maximum setup time, maximum holding cost, and cycle length rule, 10 problems were generated, giving a total of 240 problems.

Table 4 presents the results for the cases with unequal minimum cycle times, i.e., one machine was more of a bottleneck than the other. For

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

Number of parts	Maximum setup time	Maximum holding cost	z_1/z^*		z_2/z^*		$z_1 < z_2$ (%)
			Mean	Maximum	Mean	Maximum	
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

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

Number of parts	z_1/z^*		z_2/z^*		$z_1 < z_2$ (%)
	Mean	Maximum	Mean	Maximum	
4	1.0053	1.0995	1.0067	1.0861	26.26
5	1.0046	1.0732	1.0068	1.0658	37.50
6	1.0056	1.1080	1.0086	1.0637	50.00

Table 7
Performance of two heuristics relative to range of possible values

Number of parts	Mean (%)	
	$\frac{z_1 - z^*}{z^\# - z^*}$	$\frac{z_2 - z^*}{z^\# - z^*}$
4	20	25
5	12	15
6	8	15

these problems the error from the heuristic only occasionally exceeded 1% and rarely exceeded 2% of the objective value of the best permutation schedule. The average performance was well within 1%. Table 5 presents the same results when the cycle times for the two machines were forced to be equal. These problems were more difficult, as we had anticipated. We expected these problems to be more difficult since if $T = T_{\min}$ there can be no idle time on either machine. This necessitates longer 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, and the average deviation from optimality was within 2.4% for all parameter combinations.

A summary of the results appears in Table 6. Each row in Table 6 reflects the average over the 8 combinations of parameters 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 find 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 account for a majority of the costs.

Because $z^\#$ was generally very close to z^* we tabulated the statistic $(z_i - z^*)/(z^\# - z^*)$ 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 7 summarizes these results.

Table 8
Experiments with 8, 10 or 12 parts and 2 machines with unequal minimum 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	1.0120	1.0293	70
12	5	5	1.0070	1.0177	1.0090	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 9
Experiments with 8, 10 or 12 parts and 2 machines with equal minimum 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	1.0163	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

We also generated several hundred problems with more homogeneous parts. Generally, the results for these problems (not reported here) were even better, as we had anticipated.

6.5. Experiments with larger problems

The purpose of the next two experiments was to determine how well the heuristics performed on problems with more parts or 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, and we used the lower bound (z_{LB}) rather than z^* as the benchmark. Recall that 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. For these larger problems, we compared heuristics solutions with the lower bounds rather than with optimal solutions. Yet, the gaps, summarized in Tables 8 and 9, were of the same order of magnitude. 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. Thus, these problems appear to become easier as the number of parts increases.

Using results from all experiments in which both heuristics were tested, we compared the performance of the two heuristics. In the last column of Tables 8 and 9, labeled $z_1 < z_2$, we report the percentage of time that Heuristic 1 outperformed Heuristic 2. We observed that Heuristic 1 improved relative to Heuristic 2 as the number of parts increased, and as the minimum cycle lengths became more equal across machines.

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 the lower bound, $z_{LB} = HT_{min}$. The results appear in Tables 10 and 11. Again, the results are quite good. The average gap between the heuristic solution and the best permutation schedule

Table 10
Experiments with either 3 or 5 machines and 4, 5 or 6 parts

Number of machines	Number of parts	z_2/z_{Lb}	
		Mean	Maximum
3	4	1.0104	1.1020
	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 11
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	1.0860
	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

was about 1%. For the problems with 8, 10, or 12 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%.

In summary, the experiments have demonstrated that the controllable WIP rarely adds significantly to the cost of a solution. Thus greater cost savings may be achieved if by 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 sequencing issues in a multi-machine flow shop with the objective of minimizing total inventory holding costs. 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.

The heuristics can be generalized in a variety of ways. We discuss two possible extensions here. The heuristic designed for the two-machine problem can be applied directly to multiple machines by considering the 'fit' of the schedule profiles on the bottleneck machine and its immediate successor. Such a procedure might perform well when only one machine is highly utilized. The heuristic based on the asymmetric traveling salesman procedure has greater prospects for generalization. In particular, for the case of permutation schedules, the 'cost' of scheduling one part before another can be generalized to include the cost of cycle stock due to idle time on the bottleneck machine and/or WIP costs due to schedule delays that occur when the idle time is eliminated. A heuristic of the latter type may be beneficial when a savings of a few percent would be worth the additional computational effort.

Appendix A

In this appendix, we present two examples. The notation is defined in Table 2. The first

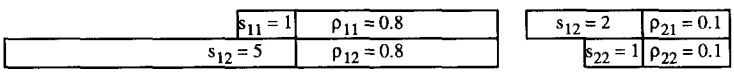

Fig. A.1. Gantt chart for two parts without shifting or wrapping

example 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 are common to both examples. There are two parts on two machines. Gantt charts for each part, when scheduled independently and with the minimum delay between machines, are given in Figure A.1, along with setup and utilization data.

The minimum cycle lengths for the two machines are

$$T_1 = \frac{1+2}{1-0.8-0.1} = 30$$

and

$$T_2 = \frac{5+1}{1-0.8-0.1} = 60,$$

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

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

Let us 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

$$[60(0.2) + 60(0.9)\phi + 56\phi = 12 + 100\phi.]$$

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 B

In this appendix we derive the inventory holding cost per unit time. 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. Since the raw materials arrive at a constant rate, d_i, for part i, the average cost per unit time 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

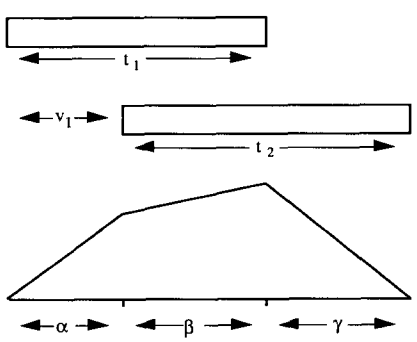

Fig. B.1a. Inventory for Case 1

machines 1 and 2 and suppress the subscript for the part i. There are three cases.

Case 1. t_1, t_2 overlap and $t_2 \geq t_1$ (see Figure B.1a):

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

where $\alpha = v_1$, $\beta = t_1 - v_1$, and $\gamma = t_2 + v_1 - t_1$. Thus

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

Case 2. t_1, t_2 overlap and $t_2 \leq t_1$: The inventory diagram is similar to that of Case 1, except in the middle section (β) the inventory decreases rather than increases.

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

where $\alpha = t_1 + v_1 - t_2$, $\beta = t_2 - v_1$, and $\gamma = v_1 1$.

Thus

$$I = \tfrac{1}{2}((t_1 + v_1 - t_2)p_1(t_1) + v_1 p_2(t_2))$$
$$= \tfrac{1}{4}v_1(t_1 p_1 + t_2 p_2) + \tfrac{1}{2}(t_1 - t_2)p_1 t_1,$$
$$I/T = v_1 d + \tfrac{1}{2}Td(\rho_1 - \rho_2).$$

Case 3. t_1 and t_2 do not overlap: Note $p_1 t_1 = p_2 t_2$, and refer to Figure B.1b. We have

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

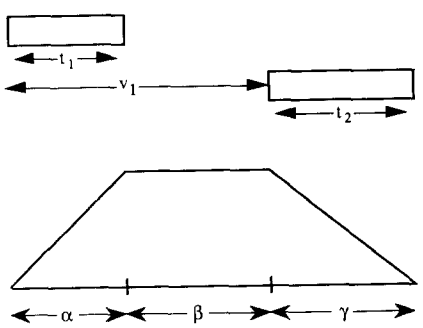

Fig. B.1b. Inventory for Case 3

where $\alpha = t_1$, $\beta = v_1 - t_1$, and $\gamma = t_2$.

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

and so forth.

The derivations above allow us to express the average inventory per unit time for any case as

$$I/T = v_1 d + \tfrac{1}{2}Td\,|\,\rho_2 - \rho_1\,|.$$

The average cost per unit time of the WIP is thus

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

Appendix C

Proof of Proposition 3. 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 part 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

$$\sum_{j=1}^{m-1}\sum_{i} h_i d_i v_{ij} \leq (m-1)TH, \tag{C.1}$$

since then we have that

$$z_{\mathrm{UB}} \leq TH + (m-1)TH = mTH = mz_{\mathrm{LB}}.$$

The above inequality is obtained by observing that

$$v_{ij} \leq T(1 - \rho_{ij}) \leq T\left(1 - \min_j\{\rho_{ij}\}\right).$$

Thus

$$\sum_{j=0}^{m-1} \sum_i h_i d_i v_{ij} \le T(m-1) \sum_i h_i d_i \Big(1 - \min_j \{\rho_{ij}\}\Big). \tag{C.2}$$

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

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

since

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

and

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

Thus

$$H \ge \sum_i h_i d_i \Big(1 - \min_j \{\rho_{ij}\}\Big) \tag{C.3}$$

Inequalities (C.2) and (C.3) give us (C.1)

For case 2

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

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

$$\ge \sum_i h_{ik} d_i \sum_{j=k}^{m} |\rho_{ij+1} - \rho_{ij}|$$

since h_{ij} is increasing in j

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

Thus

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

$$\le T \sum_{j=1}^{m-1} 2H \le 2HT(m-1).$$

So we have

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

$$\le TH + TH2(m-1) \le TH(2m-1)$$

$$\le (2m-1) z_{LB}.$$

For $m = 2$ the bounds become

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

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 aproach to lot-size scheduling for multistage, multiproduct production processes", Unpublished Ph.D. Dissertation, The Wharton School, University of Pennsylvania, Philadelphia, PA.

El-Najdawi, M.K., and Kleindorfer, P.R. (1990), "Common cycle scheduling for multi-product, multi-stage production", Working Paper, College of Commerce and Finance, Villanova University, Villanova, 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.

Hsu, J.I.S., and El-Najdiwa, M. (1990), "Common cycle scheduling in a multistage production process", *Engineering Costs and Production Economics* 20, 73–80.

Jensen, P.A., and Khan, H.A. (1972), "Scheduling in a multistage production system with set-up and inventory costs", *AIIE Transactions* 4, 126–133.

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

McCormick, S.T., Pinedo, M.L., Shenker, S., and Wolf, B.

(1989), "Sequencing in an assembly line with blocking to minimize cycle time", *Operations Research* 37, 925–935.

Matsuo, H. (1990), "Cyclic scheduling problems in the two-machine flow shop: complexity, worst-case and average case analysis", *Naval Research Logistics* 37/5, 679–694.

Roundy, R. (1991), "Cyclic schedules for job shops with identical jobs", *Mathematics of Operation Research*, to appear.

Szendrovits, A.Z. (1975), "Manufacturing cycle time determination for a multi-stage economic production quantity model", *Management Science* 22, 298–308.

Singh, D. (1990), "Using CFM as a competitive edge", *Automation* 37, 64–65.

Taha, H.A., and Skeith, R.W. (1970), "The economic lot sizes in multi-stage production systems", *AIIE Transactions* 2, 157–162.