January, 1979 LIDS-R-879

SCHEDULING OF FLEXIBLE FLOW SHOPS

by

K.L. Hitz*

This research was partially supported by National Science Foundation Grants NSF/RANN APR76-12036 and NSF DAR78-17826, "Systems Aspects of Flexible Manufacturing Systems".

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

*On leave from Dept. of Mechanical Engineering, University of Newcastle, NSW, Australia.

ABSTRACT

The report discusses the scheduling of production in a deterministic Flexible Flow Shop. This is a serial arrangement of buffered multipurpose machines connected by a conveyor system which allows fully mechanized parts handling. Machines can be bypassed, and simultaneous processing of parts with different machine routes is possible. The problem considered is that of producing a range of part types in specified ratios, with known and fixed transport and processing times. Optimal schedules are characterized as periodic loading squences which minimize idle time on bottleneck machines once they have begun operating, while satisfying constraints on buffer capacity. The computation of optimal schedules by an implicit enumeration algorithm is discussed. Early computational results indicate that this formulation of the scheduling problem leads to significant savings when compared to the minimization of makespan or mean weighted finish time customary in the scheduling literature.

ACKNOWLEDGEMENTS

I am grateful to my own University, and to Prof. M. Athans, Director of the Laboratory for Information and Decision Systems at M.I.T., for the opportunity of a sabbatical leave at MIT.

The research reported here is one of the results. It has been helped along by many stimulating discussions with the members of the Manufacturing Group at LIDS. Special thanks are due to Prof. Athans, Dr. S.B. Gershwin, Mr. J. Ward and M. J. Kimemia. I am also indebted to Mr. R. Hildebrand of the C.S. Draper Laboratories, Cambridge, Mass., who independently, and some time before me, arrived at the notion of a minimal part set.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Flexible manufacturing systems [1] are networks of multipurpose, automatic, often computer-controlled machines linked by an automated transport system. Such systems are rapidly gaining interest in manufacturing industry. They offer one way of improving productivity in those areas of manufacturing where each plant typically produces a range of similar items in volumes too low to warrant the use of product-specific mass-production machinery such as transfer lines.

The work described in this report deals with some aspects of the problem of scheduling production in a simple type of flexible manufacturing system that might be called a Flexible Flow Shop (FFS). This is a serial arrangement of multipurpose machines connected by a fixed conveyor system. The conveyors are equipped with appropriate sensors and switching devices so that the transport of parts from machine to machine is fully mechanized. The first and last machines in the FFS are loading and unloading stations where parts are loaded onto, or removed from, fixtures or pallets necessary for alignment in the machines or on the conveyor. Each internal machine of the system is assumed to be capable of performing a range of similar operations, or sequences of operations. Moreover, it will be assumed that a changeover from one operation to another can be performed automatically and at negligible cost. However, only one part at a time can be processed at each machine. Typical examples of such machines might be NC machine tools or computer-controlled assembly robots. At each internal machine of the FFS, the conveyor system has by-pass links so that it is possible for parts to visit only some of the machines in their passage through the system. For additional system flexibility, each of the machines has an entry buffer of specified capacity; in the sequel, these buffers will be assumed to operate with a first-in first-out discipline. A schematic diagram of a typical FFS is shown in Fig. 1.

Suppose now that such a system is to be used to produce a specific range of part types in a prescribed constant ratio, and in sufficient

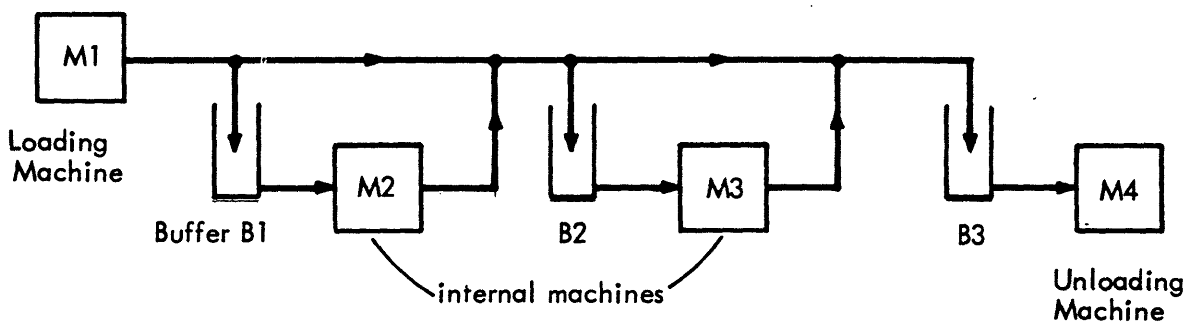


Fig. 1 - A 4-Machine Flexible Flowshop

volume to keep the system occupied for a reasonably long time. Each part type requires a prescribed sequence of operations, and we shall assume that this can be expressed by specifying the sequence of machines to be visited by each type, and the processing time required on each machine. This means, in particular, that whenever an operation can be carried out on more than one machine, an actual part type has been split into a number of "artificial" ones, each of them associated with a distinct feasible machine sequence, and that appropriate production ratios for the artificial part types have been determined independently by considering the balance of work among machines (see [2] for one method of doing this). In the sequel, only flow shops, or serial arrangements of machines, will be discussed. It will always be understood, therefore, that the machine sequence specified for each part type is compatible with the actual serial arrangement of machines.

In the literature on production scheduling, it is usually assumed that a setup cost is incurred in a changeover of a machine or production line from one part type to another. The production scheduling problem is then to determine the sizes and sequence of the batches in which the part types should be made so as to strike the best balance between setup costs and costs of holding in-process and finished inventory. In the flexible manufacturing systems considered here, however, setup or changeover costs are negligible so that an optimal mode of production will minimize work in progress by producing the required part types simultaneously rather than in a sequence of batches. The buffers in the system have the purpose of reducing machine idle time due to unequal processing times, and of providing a cushion against short-term machine breakdowns; their function is not to reduce the number of setups required.

We are thus led to the following production scheduling problem:

> Given a description of the system (number of machines, buffer capacities, travel times between machines) and of the desired production (number of part types, machine sequence for each part type, processing times on the machines, production ratios required), determine in what sequence and at what intervals parts should be loaded into the system (i.e., the first machine) so that

(i) the output of finished parts is as large as possible, subject to the constraint that part types are produced in the prescribed ratio, and

(ii) The steady state of maximum production in the prescribed ratios is reached as quickly as possible after startup or a momentary disturbance.

In this report, a deterministic version of this problem will be considered. All processing and travel times are assumed known and fixed, and all machines are considered completely reliable. For simplicity, we shall also assume that processing and travel times are integral multiples of some fundamental time step.

With these assumptions, the problem can be formulated as a special type of jobshop scheduling problem. There is an extensive literature on this problem; excellent recent discussions of methods for scheduling both flowshops (e.g., transfer lines) and general jobshops can be found in [3], [4], [5], together with extensive references to earlier work. Except in a very few simple cases, both flowshop and jobshop scheduling are well known to belong to the category of hard-to-solve, "NP-complete" combinatorial problems [3], [6], [7] which seem to be characterized by such a dearth of structure that the only feasible solution methods found so far are implicit enumeration type tree searches. Particularly in the case of the jobshop problem, these searches often require a massive computational effort for problems of even moderate size. This seems at least in part due to the choice of optimization criterion. In all studies known to the author, the scheduling problem is stated as finding either the shortest total time (makespan) or the shortest average time weighted by jobs, in which a flowshop or jobshop can process a given fixed set of jobs.

For the systems considered in this report, this seems an unnecessarily strict optimization criterion. We shall confine attention to situations where the total number of parts to be produced by the FFS in one run is so large that the system can be considered to be in an optimal state of operation whenever it produces parts in the prescribed ratio and at a steady maximum rate, both averaged over suitable time intervals. The additional time saving possible by using schedules which minimize makespan or average weighted finishing time, for the total set of parts in the run, will usually be very small.

The remainder of the report focuses on the problem of characterizing and computing loading sequences, or schedules, for the FFS which result in optimal steady state output, subject to constraints on the duration of startup transients, and on available buffer storages. It will be shown that substantial computational savings can be obtained by replacing the minimization of makespan (or mean weighted finishing time) with the requirement that the bottleneck machine or machines be fully occupied once they have started working.

A precise definition of what is meant by an optimal loading sequence is given in Chapter 2. There we will also establish the simple but basic result that loading sequences which lead to optimal steady state production, without constraints on buffer capacities or on the duration of start-up transients, form a very large class and are trivially easy to compute.

In Chapter 3, we shall introduce constraints on the length of transients as well as buffer storage, and describe an implicit enumeration algorithm for computing optimal loading sequences. Some initial computational results will be reported for the particular case of unit buffer capacity at all machines.

Finally, in Chapter 4 we shall briefly discuss how the ideas presented earlier might be extended and, in particular, how they might be incorporated in a closed-loop system of controlling production in a flexible manufacturing system.

2. MAXIMAL STEADY-STATE PRODUCTION IN AN FFS

Consider an FFS with K machines linked by a conveyor system such that the travel time $\tau_{j\ell}$ from the machine j to the input buffer of machine ℓ is known and fixed for all $1 \leq j < \ell \leq K$. Suppose that M part types are to be produced. Let p_{ij} denote the known fixed processing time required for a part of type i on machine j; $p_{ij} = 0$ means that a part of type i bypasses machine j. Suppose further that the part types are to be produced in the ratios r_i, $i = 1,2,\ldots M$: of the total number α of parts produced since startup, $r_i\alpha$ parts are required to be of type i. We shall assume throughout that the r_i are rational fractions. In some cases, particularly where the various part types are required for assembly into a larger unit, a tighter constraint may be appropriate: of every N consecutive parts produced by the system, n_i are required to be of type i, where $n_i = r_iN$, and where N is a reasonably small number. This leads to the notion of a "Minimal Part Set" (MPS).

Definition: A Minimal Part Set is a set of integers $\{n_1,n_2,\ldots n_M\}$ such that

$$n_i = r_i \sum_{i=1}^{M} n_i \overset{\Delta}{=} r_iN, \quad i = 1,2,\ldots M \tag{1}$$

and

$$\text{g.c.d.} \{n_1,n_2,\ldots n_M\} = 1 \tag{2}$$

Clearly, the system satisfies the production ratio constraints if in every set of N consecutive completed parts, n_i are of type i, $i = 1,2,\ldots M$. Moreover, it is impossible to be certain that the constraints are satisfied without checking at least the last N parts produced. Thus the minimum time required to produce the N parts is, in a sense, the shortest possible response time of the system. Ideally, this will be equal to the time T required by the bottleneck (or most heavily loaded) machine or machines to process their share of work in an MPS. This time will be called a Period; it is given by

$$T = \max_{1\leq j\leq K} \left\{ T_j = \sum_{i=1}^{M} n_i p_{ij} \right\} \qquad (3)$$

The scheduling problem studied here is to determine a loading sequence $\{(\sigma_1,t_1), (\sigma_2,t_2),\ldots\}$, (where (σ_ℓ,t_ℓ) means that a part of type σ_ℓ is loaded into machine 1 at time t_ℓ), which will cause the system to produce parts in the prescribed ratio and at the maximum possible rate. Clearly, the best steady state operation obtains when the system completes a minimal part set in every period T. This suggests the use of a periodic loading sequence of the form

$$\{(\sigma_1,t_1), (\sigma_2,t_2),\ldots(\sigma_N,t_N),(\sigma_1,t_1+T),\ldots(\sigma_N,t_N+T),(\sigma_1,t_1+2T),\ldots\}$$

where $t_1<t_2<\ldots<t_N$, $t_N-t_1<T$ and where $\{\sigma_1,\sigma_2,\ldots\sigma_N\}$ is some permutation of the items in a minimal part set. Any such sequence will be called <u>maximal periodic</u>. It will be convenient to use the same phrase for sequences in which the strict inequalities $t_i<t_{i+1}$ are relaxed to $t_i \leq t_{i+1}$; such sequences can occur when subsequences merge at a conveyor junction inside the system (see F2 below). In that case, the t_i will not be instants at which the part enters the system but instants at which it passes the point where the sequence is observed.

Suppose now that the FFS has the following features:

F1: Each machine has a FIFO buffer of unlimited capacity.

F2: The conveyor system can carry an arbitrary number of pieces on each position. This means that no part emerging from a machine will be delayed in its journey to the next machine by traffic on the conveyor. It also implies that several parts may arrive at a machine buffer simultaneously. In order to maintain deterministic behaviour throughout the system, it is assumed that parts sharing a conveyor position and destined for the same machine leave the conveyor and enter the machine's buffer according to some arbitrary but fixed rule, e.g, LIFO.

For a system with these features, any one of a very large class of loading sequences will result in optimal steady-state production, as the following theorem shows.

Theorem: In a flexible flow shop with features F1 and F2, any maximal periodic loading sequence results, after a finite interval of time, in an output sequence which is itself maximal periodic.

To prove the theorem, we first establish a preliminary result on the response of a single machine.

Lemma: Suppose that parts arrive at the buffer of a machine in a periodic sequence such that the sum of processing times of the parts arriving in one period does not exceed the capacity of the machine. Then, after a finite time interval, the sequence of parts leaving the machine wil be periodic with the same period as the arrival sequence.

Proof: Let w be the sum of processing times of the parts arriving in one period, let τ denote the length of the period, and let s_1 be the instant at which the first part of the periodic input sequence arrives. Denote by x(t) the time required by the machine to finish the part being worked on and to process the parts waiting in the buffer just prior to the instant t; x(t) can be regarded as a backlog of unfinished work in the machine and its buffer. Define η as the total time the machine would be idle in the first period, i.e., in the first τ time steps after s_1, if the machine and buffer were empty just prior to the arrival of the first part at time s_1:

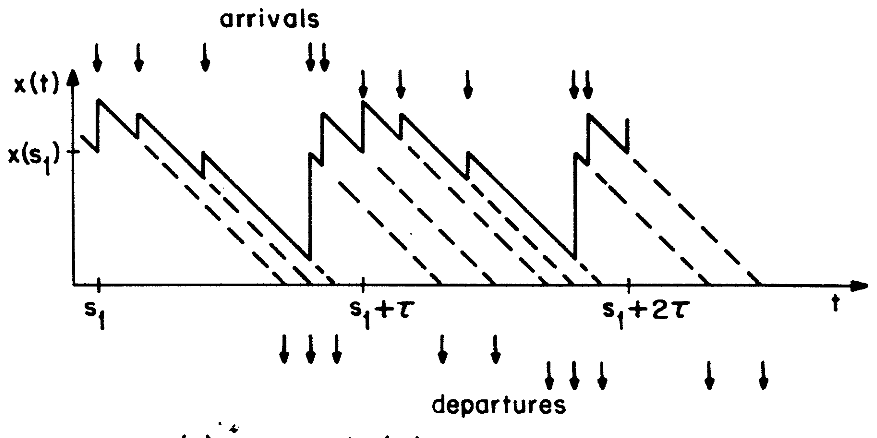
$$\eta = \max_{1<\ell\leq n+1} \left\{ (s_\ell - s_1) - \sum_{j=1}^{\ell-1} \pi_j \right\}$$

where n is the number of parts arriving in a period, and where s_ℓ, $\overline{\pi}_\ell$ are the arrival instants and processing times, respectively, of the ℓ-th part to arrive.

For example, in Fig. 2(b), η equals the sum of the intervals labelled "a" and "b".

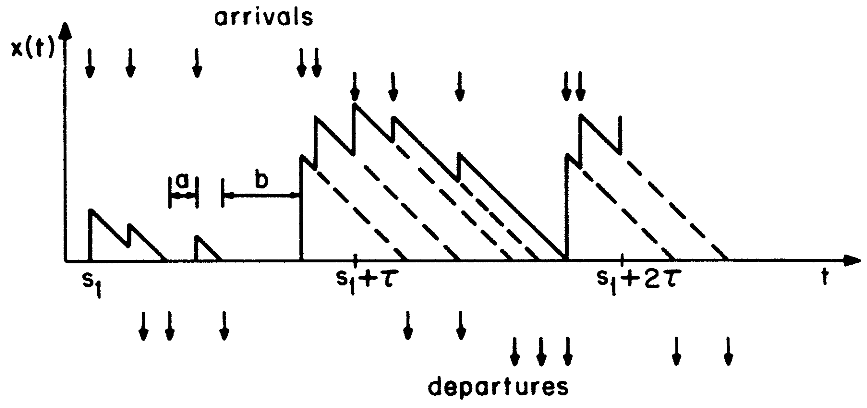
We distinguish two possibilities:

Case 1: The backlog of unfinished work never becomes zero, i.e., $x(t) > 0$ for all $t > s_1$. Clearly, this can happen only if the machine



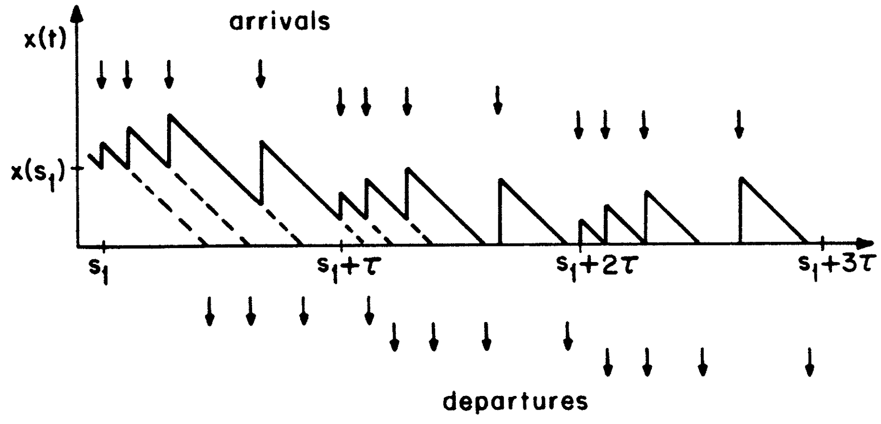
(a) $\omega = \tau$ and $x(s_1) > \eta$

(η equals a+b on fig.(b))



(b) $\omega = \tau$ and $x(s_1) = 0$

(The input sequence is the same as in(a))
$\eta = a + b$



(c) $\omega < \tau$

87049AW037

Fig. 2: Unfinished Work in a Single Machine and Its Buffer with a Periodic Sequence of Arriving Parts

ω = total work arriving in one period
τ = period of input sequence
τ_1= beginning of periodic input sequence
vertical jumps are the processing times of arriving parts

needs a whole period to process the parts arriving in a period, i.e., if $w = \tau$. In addition, the initial backlog $x(s_1)$ needs to be large enough to prevent the machine from clearing its backlog sometime in the first period: $x(s_1) > \eta$. This case is illustrated in Fig. 2(a). Under these conditions, it is obvious that for $t > s_1$, $x(t)$ is a periodic function with period τ, and this implies that the sequence of parts leaving the machine is similarly periodic, beginning at the instant of departure $s_1 + x(s_1) + \pi_1$ of the part arriving at s_1.

Case 2: The backlog of unfinished work becomes zero, i.e., the machine and its buffer become empty, at least once after time s_1: $x(t) = 0$ for at least one $t > s_1$. If $w = \tau$, this will happen during the first period if the initial backlog $x(s_1)$ is small enough, i.e., if $x(s_1) \leq \eta$. See Fig. 2(b). On the other hand, if $w < \tau$, so that the amount of work arriving in each period is strictly less than the capacity of the machine, then no matter how large the initial backlog $x(s_1)$, an instant must arrive at which the machine has cleared its backlog, and begins an idle period of finite length. This is shown in Fig. 2(c).

Let $\bar{t}$ be the smallest $t > s_1$, such that $x(\bar{t}) = 0$, and define ℓ by that $\ell\tau + s_1 < \bar{t} < (\ell+1)\tau + s_1$. Then from time $\bar{t}$ onwards, the amount of unfinished work in the machine and its buffer depend only on the input sequence, and not on any initial backlog $x(s_1)$. This in turn implies that $x(t)$ begins being periodic at some instant during the ℓ-th period, and certainly no later than at time $s_1 + \ell\tau$. However, a periodic backlog function implies a periodic output sequence. This proves the lemma.

Proof of Theorem: The first machine in the system is a loading station without an input buffer. Clearly, the sequence of parts leaving machine 1 will be maximal periodic from the instant of departure of the first part. Now consider machine 2. The sequence of parts arriving there is a subsequence of the maximal periodic sequence leaving machine 1, delayed by the constant travel time from the loading machine. It follows from the lemma that after some finite time, say τ_2, the sequence of parts leaving machine 2 is also periodic with period T. This sequence merges with the sequence of parts which bypass machine 2. Since both have the same period T, we conclude that from time τ_2 onward, a maximal periodic sequence enters the conveyor linking machines 2 and 3. The theorem now follows from a simple repetition of this argument for each subsequent machine.

The theorem assures us that if the system has unlimited buffer capacity at each machine, and if only steady-state output is of interest, the scheduling problem has a simple solution: any maximal periodic loading sequence will do. After a finite time, an initially empty system will produce a minimal part set in every period.

Of course, a poor choice of loading sequence may result in a long transient. During this transient, the system will produce parts at less than the maximum possible rate, and since the sequence of parts will usually be permuted in its passage though the system, there is no assurance that an integral number of minimal part sets will be produced during the transient. Thus the output will in general satisfy the production ratio constraints only if the production during the transient is ignored.

It is possible to overcome this difficulty, as well as reduce the duration of the transient phase, by a partial preprocessing of parts which fills the machine buffers to an appropriate level prior to starting the maximal periodic loading sequence. This yields a heuristic scheduling algorithm which will be discussed in more detail in Chapter 4.

First, we turn to the problem of characterizing and computing maximal periodic loading sequences which are optimal in the sense that they minimize undesirable start-up transients, and satisfy explicit constraints on available buffer storage.

Definition: A maximal periodic loading sequence will be called optimal if, when it is applied to an initially empty FFS, the following hold:

(i) specified constraints on buffer capacity are satisfied

(ii) the output sequence of completed parts is maximal periodic from the instant at which the first part leaves the system.

We shall use the term "loading schedule", or simply schedule, to denote the first N components of a maximal periodic loading sequence, i.e., the ordered set of pairs

$$\underline{\sigma} = \{(\sigma, t_1), (\sigma_2, t_2), \ldots, (\sigma_N, t_N)\}$$

where, as before, $t_i < t_{i+1}$, $i = 1,2,\ldots N-1$, where $t_N - t_1 < T$ and where $\{\sigma_1, \sigma_2, \ldots \sigma_N\}$ is a permutation of the MPS.

Suppose now that parts are loaded into an initially empty FFS according to a given schedule $\underline{\sigma}$. Let S(1,j) denote the starting time, on machine j, of the first part of $\underline{\sigma}$ to arrive at machine j, and let C(N,j) be the completion time, on machine j, of the last part of $\underline{\sigma}$ to be processed by machine j.* Then the condition

$$C(N,j) - S(1,j) \leq T, \quad j = 1,2,\ldots K \tag{4}$$

is clearly sufficient for the schedule $\underline{\sigma}$ to yield an optimal loading sequence: on every machine, and for all k = 1,2,..., processing of all the parts in the first k minimal part sets is completed before the first part of the (k+1)-st MPS arrives. Hence the processing sequence on every machine, including the last, is periodic from the instant at which the first part reaches that machine.

An example of an FFS scheduling problem is given in Fig. 3, together with an optimal schedule which satisfies the constraint of unit buffer capacity at each machine. On the chart, two-digit numbers are used to identify individual parts. For example, "52" refers to the second part of type 5. This particular part is loaded at time 16, leaves machine 1 at time 18, bypasses machine 2 to arrive at the buffer of machine 3 at time 28, waits there for 3 time steps while part 31 is processed, leaves machine 3 at time 36, and so on. It leaves the system at time 74.

Note that condition (4) is not necessary for optimality of a schedule. Suppose that C(N,j) - S(1,j) > T for some schedule $\underline{\sigma}$ and some machine

---

*Note, however, that this notation is not meant to imply that each machine is visited by all N parts in an MPS.

(a) Problem Data:

No of machines K = 6
No of part types M = 5
Minimal Part Set = {2,2,1,1,3}; Period T = 42

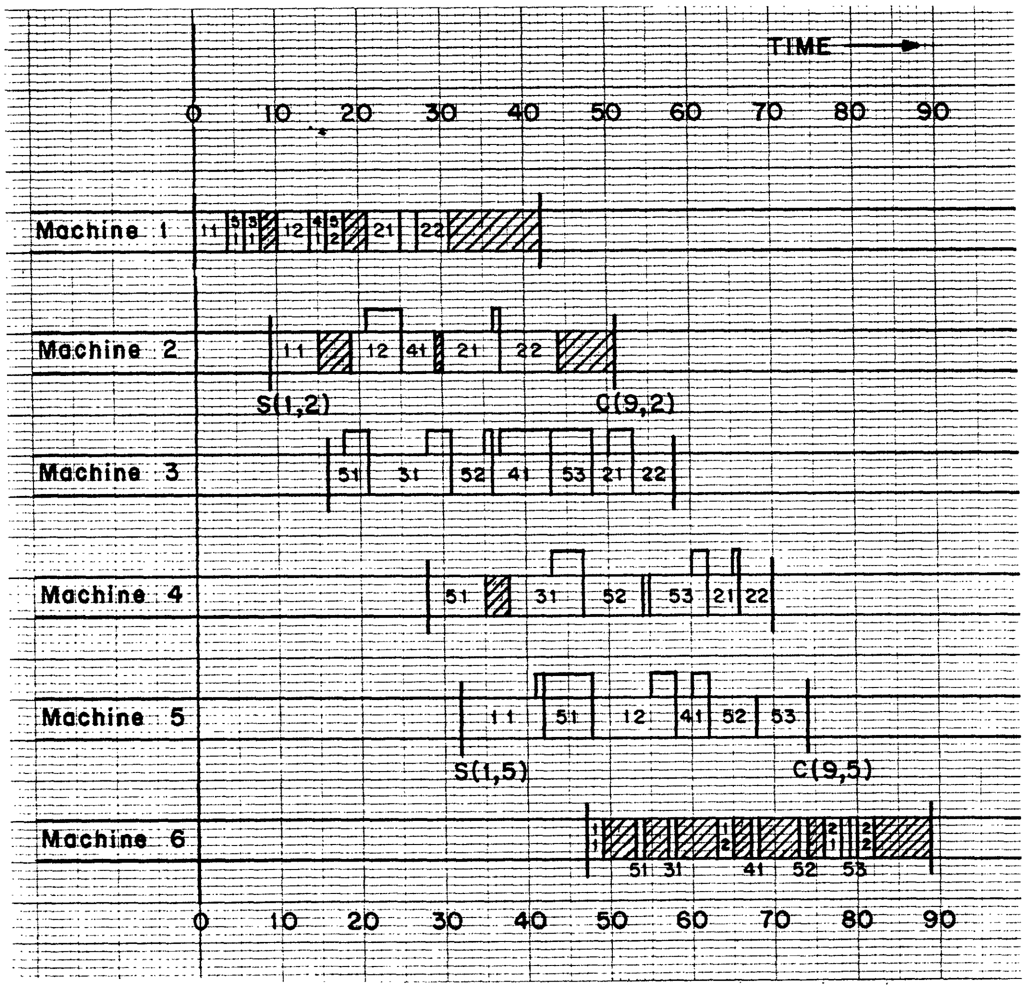
Travel times $\tau_{j\ell}$

		j				
		1	2	3	4	5
ℓ	2	5				
	3	10	6			
	4	16	12	7		
	5	21	17	12	6	
	6	25	21	16	10	5

Operation times

		Machine					
		1	2	3	4	5	6
Part type	1	4	6	0	0	10	2
	2	4	7	5	4	0	2
	3	2	0	10	9	0	1
	4	2	4	7	0	4	1
	5	2	0	5	7	6	1
Total		26	30	42	38	42	13

(b) Ghannt Chart of Optimal Schedule



Optimal schedule $\underline{\sigma}$ = {(1,0),(5,4),(3,6),(1,10),(4,14),(5,16),(2,21),(5,25),(2,27)}. Rectangular blocks above machine bars indicate occupied buffer shaded areas are idle times in one period.

j. Then $j < K$ since otherwise the output sequence could not have a minimal period. It may happen that the processing on machine j of the last few items in the first MPS in the interval $[S(1,j) + T, C(N,j)]$ does not interfere with the processing of the first items of the next MPS in the same interval. If this is so for all machines for which $C(N,j) - S(1,j) > T$, then clearly the schedule can be repeated indefinitely; the processing of the k-th MPS will not affect the timing of the operations for th (k+1)-st MPS, and the system output has the same dead beat transient response as if (4) were satisfied.

However, from now on, we shall identify optimal schedules with those that satisfy (4), treating as negligible the possibility that the only optimal schedules for a problem violate (4). We do this because (4) leads to a fairly simple and efficient algorithm for computing optimal schedules; permitting some overlap in the processing of successive minimal part sets would make the algorithm very much more complex and computationally expensive.

3. COMPUTING OPTIMAL FFS SCHEDULES

In this chapter, we discuss the computation of schedules which satisfy condition (4), by means of an implicit enumeration algorithm. This algorithm can be regarded as a formal statement of the procedure one would naturally adopt in a manual construction of optimal schedules using Ghantt charts [3]. To simplify the exposition, we shall first restrict the discussion to so-called permutation schedules [4]. In such schedules, processing of different parts on common machines must always occur in the order in which the parts entered the system. For example, the schedule in Fig. 3 is not a permutation schedule: part 21 enters the system before part 53, yet is processed on machines 3 and 4 after part 53.

Let $T_j = \sum_{i=1}^{M} n_i p_{ij}$ denote the total time required by machine j to process its share of work in a minimal part set. Then $U(0,j) \overset{\Delta}{=} T - T_j$ is the maximum amount of time that machine j can be idle in the interval $[S(1,j), C(N,j)]$ if condition (4) is to be satisfied by the schedule $\underline{\sigma}$. Thus, an alternative way of stipulating (4) is to require that in an optimal schedule, the sum of the gaps, or idle periods, between successive work periods must not exceed $U(0,j)$ for $j = 1,2,\ldots K$. Of course, $U(0,j) = 0$ on a bottleneck machine. Let $\underline{\sigma}_k$ denote a "partial schedule":

$$\underline{\sigma}_k = \{(\sigma_1, t_1), (\sigma_2, t_2), \ldots, (\sigma_k, t_k)\}$$

where the σ_i form a proper subset of the items in an MPS. For convenience, we usually take $t_1 = 0$. Let $\hat{U}(k,j)$ be the total idle time on machine j between the work periods necessary to process $\underline{\sigma}_k$; if only one, or no, part of $\underline{\sigma}_k$ requires processing on machine j, set $\hat{U}(k,j) = 0$. Then $U(k,j) = U(0,j) - \hat{U}(k,j)$ is the amount of idle time available on machine j, and not yet "used up" by $\underline{\sigma}_k$. We shall call $\underline{\sigma}_k$ a feasible partial schedule if

(i) at no machine, the maximum number of buffer positions required in processing $\underline{\sigma}_k$ exceeds the buffer capacity B_j,

(ii) $U(k,j) \geq 0$ for all $j = 1,2,\ldots K$.

Suppose that in the search for an optimal schedule for a particular problem, a feasible partial schedule $\underline{\sigma}_k$ has been found. We now attempt to find a feasible $\underline{\sigma}_{k+1}$. Let $C(k,j)$ be the finishing time on machine j of the last item of $\underline{\sigma}_k$ to be processed by machine j. Let $A(k,j)$ denote the earliest time at which a new part from $\underline{\sigma} - \underline{\sigma}_k$ can arrive at the buffer of machine j, subject to the constraints that the buffer cannot be full at time $A(k,j)$ and that processing of the new part on machine j cannot begin before time $C(k,j)$. (Recall that only permutation schedules are allowed for the moment.)

The search algorithm consists of the following:

Step 1: Let $\mathcal{F}_1 = \{1,2,\ldots M\}$; for $k > 1$, $\mathcal{F}_k$ is the set of part types which may still be possible in the k-th place of an optimal schedule having $\underline{\sigma}_{k-1}$ in its first k-1 places.

Step 2: If $\mathcal{F}_1$ is empty, stop; an optimal schedule does not exist. Otherwise, select a part type σ_1 from $\mathcal{F}_1$, set $t_1 = 0$, and for all $j = 1,2,\ldots K$, set $U(1,j) = U(0,j)$ and compute $A(1,j)$ and $C(1,j)$. Set $k = 1$.

Step 3: Compute the set $\mathcal{F}_{k+1}$ of part types for which $n_i - m_i > 0$, where n_i and m_i are the numbers of parts of type i in the MPS and in $\underline{\sigma}_k$, respectively.

Step 4: If $\mathcal{F}_{k+1}$ is empty, go to step (7). Otherwise, select a part type σ_{k+1} from $\mathcal{F}_{k+1}$ according to some priority rule. Set $S(k+1,j) = 0$ for $j = 1,2,\ldots K$.

Step 5: Compute the earliest possible times $S(k+1,j)$ at which a part of type σ_{k+1} can begin being worked on by each of the machines j in its route, subject to the joint constraints imposed by the $A(k,j)$. If $S(k+1,j) - C(k,j) > U(k,j)$ for any machine j visited by part type σ_{k+1}, delete part type σ_{k+1} from $\mathcal{F}_{k+1}$, and return to step 4.

Step 6: A feasible "continuation" of $\underline{\sigma}_k$ has been found:

$\underline{\sigma}_{k+1} = \{(\sigma_1,0),\ldots, (\sigma_k, t_k), (\sigma_{k+1}, S(k+1,1))\}$.

If $k = N-1$, stop; $\underline{\sigma}_{k+1}$ is an optimal schedule. Otherwise, compute and store $U(k+1,j)$, $C(k+1,j)$, $A(k+1,j)$, record F_{k+1} for possible future reference, replace k by k+1 and return to step 3.

Step 7: A feasible continuation of $\underline{\sigma}_k$ does not exist, i.e., no optimal permutation schedule exists which has $\underline{\sigma}_k$ as its first k entries. Delete part type σ_k from F_k. If $k = 1$, go to step 2. Otherwise, replace k by k-1 and go to step 4.

The major computational effort in the algorithm is required by the computation of the earliest possible starting times in step 6. This (possibly iterative) computation can be accomplished by the following block of Fortran code (LEV $\equiv$ k, i $\equiv \sigma_{k+1}$, TRTIME(J,L) $\equiv \tau_{j\ell}$):

```
      S(LEV+1,1) = C(LEV,1)
 1Ø   MC = 1
      DO 2Ø MCN = 2,K
         IF(P(i,MCN).EQ.Ø)GO TO 2Ø
        TEMP = S(LEV+1,MC) + P(i,MC) + TRTIME(MC,MCN)
         IF(TEMP.GE.A(LEV,MCN)) GO TO 3Ø
         S(LEV+1,1) = S(LEV+1,1) + A(LEV,MCN) - TEMP
         GO TO 1Ø
 3Ø      IF(TEMP.GT.C(LEV,MCN)) GO TO 4Ø
         S(LEV+1,MCN) = C(LEV,MCN)
         GO TO 5Ø
 4Ø      S(LEV+1,MCN) = TEMP
 5Ø      MC = MCN
 2Ø   CONTINUE
```

The algorithm just described can be regarded as a fairly simple member of the general class of branch-and-bound, or tree search, algorithms for the solution of combinatorial scheduling problems (see Ch. 6 of [3] for a formal survey): a partial feasible schedule $\underline{\sigma}_k$ is a node in the search tree, and feasible continuations $\underline{\sigma}_{k+1}$ are its descendants or successor nodes. Interpreted in the branch-and-bound framework, the algorithm has two distinctive features:

(i) Minimization of a single-valued objective function (e.g., makespan) is replaced by searching for schedules satisfying a set of constraints: the sums of idle times between busy periods on each machine must not exceed the known limits U(0,j).

(ii) During the search, the idle time on machine j of the partial schedule $\{\sigma_k, (\sigma_{k+1}, S(k+1,1))\}$ is used as a lower bound on the idle time on machine j of any permutation schedule of the form $\{\underline{\sigma}_k, (\sigma_{k+1}, S(k+1,1)),\ldots(\sigma_N, t_N)\}$.

The first of these features by-passes the difficulty in many branch-and-bound schemes of obtaining a tight upper bound early in the search. In our case, perfectly tight upper bounds are known from the outset. The second feature yields lower bounds which are easy to compute, at least in comparison to the effort required to obtain good lower bounds in the branch-and-bound solution of the classic permutation flowshop problem [4].

It was hoped that these features would combine to yield a computationally efficient algorithm. This turned out to be the case, at least in regard to speed of computation. A program was written in Fortran IV for the special case of unit buffer capacities at all machines, and tested on an IBM 370/168 with a number of sizeable problems derived from those described in [1]. The search tree generated was usually quite large; for problems with K = 6, M = 5 and N = 25, several thousand nodes were usually required to find an optimal permutation schedule, or to establish that none existed. However, the computation time per node was only 0.7 msec on average.

It also became apparent that the restriction to permutation schedules is too drastic in an FFS scheduling problem involving different processing times and part types which do not require processing on every machine. In such cases, permutation schedules often fail to take advantage of the key feature which can make the FFS more efficient than a transfer line: the ability of parts to reach a downstream machine and begin being processed there while an upstream machine is still busy on a part which entered the system earlier. The example in Fig. 3 is a case in point: no optimal permutation schedule exists (the program concluded this after generating 32 nodes in the search tree).

It seems that the simplest way of removing the restriction to permutation schedules is to enable the algorithm to look for feasible continuations of a partial schedule not just by adding a single part, but by adding a group of several parts having "interlocked processing profiles" on a Ghantt chart, i.e., which overtake each other in their passage through the system. For example, in the schedule of Fig. 3, parts 41 and 52 form a pair of interlocked parts, or a "composite part". Together they yield a feasible continuation of the partial schedule $\underline{\sigma}_4 = \{(1,0),(5,4),(3,6),(1,10)\}$.

Individually they do not: neither $\{\underline{\sigma}_4, (4,t_5)\}$ nor $\{\underline{\sigma}_4, (5,t_5)\}$ are feasible for the smallest possible values of t_5. The former continuation would force machine 3 to be idle between time steps 31 and 35, while with the latter, machine 5 would be idle between times 58 and 60.

Naturally, an algorithm which can cope with such "composite parts" is considerably more complex. It seems that the simplest program structure is obtained when composite parts are treated as different part types. The set of possible composites can be pre-computed. However, the timing of the operations required to process the composite, and thus the amount of idle time between these operations, depend on the partial schedule to which the composite is joined and cannot be computed beforehand. For large composites, this computation is quite complex and time-consuming.

The Fortran program in the Appendix is a first attempt at developing a code which can handle composite parts. Only the special case of unit buffer capacity is considered, and composites can consist of only two parts. In other words, the schedules generated are restricted to those in which no more than two successive parts can overtake each other (once, or more frequently), and any such "crossed pair" cannot itself be interlocked with its neighbors.

The program was tested on one small and four substantial problems; see the Appendix for problem data and printouts. The results are summarized below:

Problem No.	K	M	N	T	U(0,j)	Result	Nodes Generated in Search
1 (Fig. 3)	6	5	9	42	(16,12,0,4,0,29)	Opt. Soln.	20
2	6	5	25	119	(49,46,0,2,1,84)	Opt. Soln.	370
3	6	5	25	117	(57,44,48,0,5,82)	Opt. Soln.	26961
4	6	5	21	143	(75,5,38,0,25,109)	Opt. Soln.	8812
5	6	5	21	143	(75,5,38,0,1,109)	No Opt. Soln. exists	1667

On an IBM 370/168, the average computing time per node was 1.6 msec (up from 0.7 for the algorithm for permutation schedules), so that a computation time of the order of several seconds can be expected on a powerful computer for problems of reasonable size ($K \leq 6$, $N \leq 20$).

It is of interest to compare this with the computational results reported in [4] for the minimum-makespan permutation flow shop problem with no buffer constraints. There it is stated that "fewer than half of the 20/5 problems [20 jobs, 5 machines] could be solved within one minute" of CPU time on a CDC Cyber 73-28 computer.

Thus, while the set of problems solved so far is too small to allow an accurate assessment of the performance of the algorithm, the results are sufficiently encouraging to warrant further development.

It seems that the difficulty of solving a problem, apart from sheer size, depends on

(i) the range of processing times among different parts in an MPS,

(ii) the balance of work among machines in an MPS,

(iii) the buffer capacity.

The first of these determines the amount of overtaking possible, and hence the size of composite parts which may be necessary in constructing optimal schedules. The results obtained so far indicate that unless the processing times of different parts on different machines differ by whole orders of magnitude, it may not be necessary to consider composites of more than three parts. Allowing pairwise overtaking already gave a dramatic improvement in the success rate of the algorithm, even in cases where the ratio of maximum to minimum processing times on internal machines was as high as 5 (see problem 5 in the appendix). In networks of NC machine tools, at least, this value can be regarded as a reasonable upper limit on the range of processing times.

The second factor, work balance among machines, affects the tightness of the constraints. If there is a high degree of unbalance, i.e., a large percentage of idle time in an optimal schedule on all machines except one bottleneck one, then in general there will exist a large number of optimal schedules, and only a very "narrow" portion of the search tree needs to be explored to find one. At the other extreme, when many machines are nearly fully occupied in an optimal schedule, the search will also be very brief: because of the tightness of the idle time constraints, even small feasible partial schedules will in general have very few feasible descendants (see problems 1,2,5). The most difficult problems seem to be those where the number of optimal schedules is not very large, but where the idle time constraints are not very tight either, at least early in the search (see problem 3). In that case, pruning of the search tree does not begin until a substantial depth is reached, so that many branches must be explored to a considerable depth. For this case, it may be desirable to augment the simple feasiblity tests by calculating a more elaborate but tighter lower bound on the idle time of any completion of a partial schedule.

The third factor, buffer capacity, is, of course, an external constraint. However, unpublished simulation studies at MIT and the C.S. Draper Laboratories [8] seem to indicate that the number of optimal schedules (and hence the speed with which a search procedure can find one) increases dramatically as the buffer capacity at internal machines is increased. It is of interest to explore this conjecture further.

An algorithm which removes the major limitations of the program in the Appendix is currently being developed.

The objectives in the revision are to

(i) allow buffer capacities of arbitrary size,

(ii) extend the size of composite parts to three,

(iii) avoid multiple occupancy of conveyor positions. The present program merely avoid simultaneous arrivals of parts at a machine buffer.

(iv) minimize program complexity and overhead during the search. The present program is not very elegant or efficient; a considerable amount of unnecessary overhead is incurred in the storage of partial schedules, and in their modification during backtracking steps.

(v) incorporate a number of options if an optimal schedule does not exist or cannot be found in a specified amount of computing time. There are several possible alternatives: (a) to compute a good completion of the longest feasible partial schedule found, (b) to repeat the search with a larger value of the period T, i.e., with an identical increase in the allowable idle times U(0,j) of all machines, and (c) to repeat the search with greater buffer capacity.

Results of this revision will be presented in a sequel to this report.

4. EXTENSIONS AND CONCLUSIONS

In this chapter, we shall briefly discuss a number of desirable extensions of the ideas presented earlier, and then present some conclusions.

4.1 Closed-Loop Control of a Flexible Flow Shop

The computation of optimal schedules described so far in this report is based on the assumption of a perfectly deterministic system model: transport and processing times are known and fixed, machines never fail and always produce perfect work, and parts and pallets are never in short supply at the loading station. However, even in systems where transport and processing times are accurately known, such as in networks using assembly robots, NC machine tools or automatic inspection equipment, breakdowns will occur, or the work done may deteriorate so that parts will be rejected. Thus, schedules computed on the assumption of a perfect system will be useful only to the extent that they can be incorporated in a closed-loop or feedback control arrangement which can detect deviations from optimal behavior and make appropriate corrections.

We shall briefly outline some of the problems which will need to be solved in deriving such closed-loop control systems.

Consider first the problem of handling machine failures in an otherwise deterministic system. In the case of transfer lines, the standard method of reducing the effect of breakdowns is to provide buffer space between machines [9], [10]. With such intermediate storage, the neighbors of a broken-down machine can continue operating until the buffer upstream of the stoppage fills up, or the downstream buffer is depleted. Thus the effects of short-term stoppages can be effectively localized.

It seems reasonable to expect that the provision of adequate buffer capacity will be just as important in an FFS as in a transfer line, at least in the case of failures of short to medium duration. Naturally, a schedule which is optimal for the case of full machine availability cannot in general be maintained during a breakdown. What is needed then is an algorithm which makes the appropriate changes in the loading sequence, on the basis of the current buffer levels and machine states, so as to drive the system back to optimal operation after a breakdown. It is tempting to speculate that if the diversity of part types and machine routes is properly exploited in such an algorithm, then a good insulation of the system against short-term individual machine breakdowns can be obtained with less buffer capacity than would be required in a transfer line.

The problem of breakdowns of long duration is more straightforward. If the disabled machine is essential for all part types being produced, then, of course, there is no alternative to shutting the system down once buffer stocks and storage places are exhausted. However, it may be possible to bypass the disabled machine and to have its function taken over by other machines. This will alter the routing for some or all of the part types. A new allocation of items to machine routes can then be computed (e.g., by the methods in [2]), and a new optimal schedule generated for use during the breakdown period. The "dead-beat" response to optimal schedules will ensure a rapid settling down to the new steady state.

It may not be possible to maintain the original production ratios with a breakdown in the system. In that case, it may be advantageous to produce whatever mix of parts can be made efficiently with the reduced system during the breakdown, followed by a "catch-up" phase in which the backlog of missing parts is cleared. The problem then is to determine what production ratios should be used in each of the two phases so that the total production loss due to the breakdown is minimized. How to formulate this problem in the simplest way (possibly as a small integer programming problem) and solve it efficently, requires further investigation.

Apart from machine breakdowns, an online feedback control system must also be able to handle occasional rejection of unfinished parts. This may occur as a result of outright machine failure, or by a gradual deterioration of work. As a result, the total part production plus the set of parts currently in the system may not add up to an integral number of minimal part sets. The scheduling algorithm must then determine how this backlog can be cleared with a minimum of idle time on bottleneck machines.

4.2 Fast Heuristics for Near-Optimal FFS Schedules

The implicit enumeration algorithm discussed in Chapter 3 for computing optimal schedules (and hence loading sequences) may require more computing time or computer capacity then is available for an on-line control system. It is therefore of interest to investigate alternative algorithms in which the tight constraints on dead-beat response, on buffer capacity, or an maximal steady-state production are relaxed.

One possible approach is to exploit the result of the theorem in Chapter 2, by initially ignoring constraints on buffer capacity. This yields the following procedure:

Step 1: Assume a reasonable schedule $\{(\sigma_1,t_1),\ldots(\sigma_N,t_N)\}$. One plausible scheme is to schedule parts of each type at roughly uniform intervals.

Step 2: Assuming unlimited buffer capacity, compute the steady-state response of the initially empty system to the maximal periodic loading sequence derived from the schedule. By steady-state response we mean the detailed processing sequence of a minimal part set at each machine once the system has settled down to steady state, i.e., once the output sequence has started being periodic with period T. According to our theorem, this must occur after a finite number of periods. This step essentially requires a simulation of the system.

Step 3: Scan the steady-state response obtained in Step 2 for the maximum buffer occupancy at each machine. If this does not exceed available buffer capacity, stop; a feasible schedule has been found with which a maximal period loading sequence can be formed. If the available buffer capacity is exceeded, attempt to modify the current schedule so as to reduce peak buffer occupancy and return to step 2. If no further changes are possible or apparent, go to step 4.

Step 4: Expand the current schedule by the minimum amount of time necessary so that the peak buffer occupancies are reduced to match available capacity. If this time is δ, say, then the resulting schedule can be used to form a periodic loading sequence of period $T+\delta$. This sequence will be feasible in that buffer constraints are satisfied, but since a minimal part set is produced only once every $T+\delta$ steps, the production rate may be less than the maximum possible.

An important feature of the optimal schedules generated by the implicit enumeration algorithm in Chapter 3 is that the first part to arrive at each machine does not wait in the buffer. This guarantees that when the corresponding loading sequence is applied to an empty system, the output sequence is immediately periodic. This is no longer true in general for the loading sequences obtained in steps 3 or 4 of the heuristic described above. A finite number of minimal part sets must pass through the system before the output becomes periodic. It may be possible to shorten this transient phase with the following trick.

Let ℓ be the number of minimal part sets processed in the transient phase starting at $t_1 = 0$. This is obtained as a byproduct of the computation in step 2. That computation also establishes for each machine

(i) the instant of arrival at the buffer of machine j of the first part of the (ℓ+1)-st minimal part set to be processed by machine j, say a_j,

(ii) the state of the buffer, i.e., the number of parts, and their type and order, waiting at time a_j.

Now suppose that it is possible to arrange a pre-processing phase in the system prior to applying the periodic loading sequence at time

$t_1 = 0$, so that the buffer of machine j has the state at time $a_j - a_1$ which it would reach at time a_j if the sequence were applied to an empty system. In that case, the system will settle down to a periodic steady state immediately after the pre-processing stage, and the output satisfies the production ratio constraints from the instant at which the first minimal part set is completed.

The heuristic described above is currently being developed in detail. The important features needed will be a fast procedure for computing the steady-state response to a periodic loading sequence (step 2), and (in step 3) effective rules for changing a schedule so as to reduce peak buffer requirements. The latter may well turn out to be less difficult than might be expected, at least for systems in which a reasonable amount of buffer capacity is available.

4.3 Conclusions

The system described in this report is the first step of an attempt to find out whether on-line production control in a flexible manufacturing system using deterministic scheduling theory is likely to be useful and computationally feasible. The current literature on flowshop and jobshop scheduling is certainly not encouraging: even in problems of moderate size, the minimization of total processing time for a fixed set of jobs is shown to require substantial computational resources.

The present study has examined a reformulation of the traditional scheduling goals, arguing that for the case of reasonably long production runs, schedules which quickly yield a maximum steady-state production rate are just as acceptable as those which minimize the total duration of a production run. At least for the flexible flow-shops considered, it appears that the reformulation described in Chapter 2 substantially reduces the computational burden.

The computation of such schedules by an implicit enumeration scheme was discussed in Chapter 3. The results obtained with a first version of this algorithm are encouraging. They indicate that in certain types of flexible flow shops, on-line control of production based on optimal solutions to deterministic scheduling problems may well be perfectly feasible, and may yield significant economic benefits. Typical examples of such systems are networks of N/C machine tools, which have fairly long but quite accurately known processing times and are in many cases already equipped with on-line computer facilities. Of course, the development of effective closed-loop control algorithms, using deterministic scheduling, requires much further research; some of the problems to be resolved were outlined earlier in this Chapter.

More research is also needed on how to extend the ideas presented here to systems other than flexible flow shops. Even in the case of an FFS, the formulation of the scheduling problem used here may not be the most natural or simple. An example might be a system having several identical machines. The requirement of the present model, that parts following different machine routes must be of different type, can then lead to an unduly large number of part types (many of them artifically different) and a large minimal part set. For such systems, a scheduling model is required which allows for alterative machine routes in a more direct fashion.

In an FFS, the part routes must be compatible with the actual sequence of machines. In a general flexible manufacturing system, arbitrary part routes are possible; a simple example is a set of machines connected by a loop conveyor. In such systems, the computation of optimal schedules (as defined here) is likely to be much more difficult. It is hard to see how the fairly simple algorithm of Chapter 3 can be extended to allow arbitrary routing of parts. A more promising approach might be to develop a heuristic scheduling procedure along the lines of the algorithm in Section 4.2. As a first step in this direction it will be necessary to generalize the theorem in Chapter 2 to systems which permit arbitrary part routes.

APPENDIX

A FORTRAN PROGRAM FOR COMPUTING OPTIMAL FFS SCHEDULES

A1: List of Parameters and Variables

(CPS $\equiv$ current partial schedule)

Input data:		
K	K	no of machines
M	M	no of part types
TRTIME (I,J)	τ_{ij}	travel time from machine i to entry buffer of machine j.
JOBNO (I)	n_i	no of parts of type i in a minimal part set
JOBTIM(I,J)	p_{ij}	processing time of a part of type i on machine j.
MAXNOD		maximum no of partial schedules examined for possible continuation before search is terminated.
Parameters defined from input data:		
N	N	no of parts in an MPS
MACH(I)		no of machines visited by part of type i
JOBRTE(I,L)		index of machine visited by a part of type i for its L-th operation.
PERIOD	T	time required to process MPS on bottle-neck machine without idle time.
MAXPRT(J)		no of parts in the MPS which must be processed on machine J.
CRSPR(I)		no of part types from which a crossed pair can be formed with a part of type i
OTAKE(J,I)		index of i-th part type which can form a crossed pair with, or overtake, a part of type J
REACH(I,J)		minimum time in which some part in the MPS can reach machine J after its arrival at machine i
Variables		
REMJOB(I)		no of parts of type i not included in the CPS (initially, REMJOB(I) = JOBNO(I))
SLACK(J)	$U(k,j)$	amount of permissible idle time on machine J not yet used up by the CPS.

Program Label	Symbol used in text	Description
LEVEL	k	no of parts in the CPS (=depth of search tree). If LEVEL = N, search terminates; an optimal schedule has been found.
PPERM(L)	σ_ℓ	index of part type of ℓ-th part in the CPS (ℓ = 1,2,...LEVEL)
XPERM(L)		=1 if ℓ-th part in the CPS either overtakes its successor or is overtaken by its predecessor =0 otherwise
NCOUNT		no of nodes (partial schedules) in search tree examined so far
BIGLEV		no of parts in largest feasible partial schedule found so far
MCPART(J)		no of parts processed by machine J in the CPS
MSCHED(I,J)		index of part type of the i-th part processed by machine J in the CPS.
START(I,J)		time at which machine J begins processing its i-th part in the current partial schedule
FINISH(I,J)		time at which machine J completes its i-th part in the CPS.
BUFF(I,J)		= if the i-th part visiting machine J in the CPS has to wait in the entry buffer of machine J = 0 otherwise
NODE(L,I)		= -1 if it has been discovered that the partial schedule {PPERM(1),...PPERM(L),I} is not feasible ($1 \leq I \leq M$) because <u>either</u> all parts of type I in the MPS are already included in the first L parts <u>or</u> because the earliest possible addition of I uses up more idle time on some machine than is left in a schedule starting with {PPERM(1),...PPERM(L)}, <u>or</u> because all feasible continuations of {PPERM(1),...PPERM(L),I} have been found to have no feasible continuations themselves. Thus NODE stores the search tree. If NODE (1,I) = -1 for all I=1,2,...M, the search is complete; an optimal schedule does not exist. = 0 otherwise

Program Label	Symbol used in text	Description
TLAST(J)	$C(k,j)$	time at which machine J completes the last operation in the CPS.
WLAST(J)		length of last operation on machine J in the CPS.
STRT1(J)		tentative starting time on machine J of a possible continuation J1 of the CPS
FIN1(J)		tentative finishing time on machine J of a possible continuation J1 of the CPS
STRT2(J)		tentative starting time on machine J of part J2 in a possible continuation of the CPS by a crossed pair J1, J2
FIN2(J)		tentative finishing time on machine J of part J2 in a possible continuation of the CPS by a crossed pair J1, J2
J1	σ_{k+1}	part type of possible continuation of the CPS
J2		part type of possible continuation of the CPS by a crossed pair J1, J2

A2: Listing of Fortran Program

```
G1  RELEASE 2.0                MAIN               DATE = 79017          15/06/17

      C ** FLEXIBLE FLOWSHOP - VERSION 2 **
      C
      C WRITTEN BY: K. L. HITZ      JULY 1978
      C
      C PROGRAM USES AN IMPLICIT ENUMERATION TO SEARCH FOR A SATURATED SCHEDULE
      C FOR STEADY-STATE PRODUCTION IN A FLOWSHOP OF AT MOST K = 9 MACHINES WITH
      C          (1) FIXED TRAVEL TIMES BETWEEN MACHINES(INCL ONE TIME STEP IN ENTRY
      C              BUFFER OF EACH MACHINE EXCEPT THE FIRST)
      C          (2) SINGLE-POSITION ENTRY BUFFERS
      C          (3) BY-PASS CONVEYORS AT ALL MACHINES EXCEPT THE FIRST AND LAST
      C              (WHICH ARE LOADING AND UNLOADING STATIONS VISITED BY ALL PARTS)
      C
      C THERE ARE UP TO M=9 ITEM TYPES TO BE PROCESSED, EACH WITH JOBNO(I) PARTS
      C IN A 'MINIMAL PART SET' (MPS) : A SMALLEST SET OF INTEGER PART NUMBERS
      C WHICH SATISFIES THE SPECIFIED PRODUCTION RATIOS.
      C THE TOTAL NO OF ITEMS IN AN MPS MUST NOT EXCEED 99
      C
      C A SATURATED SCHEDULE IS ONE WHICH PROCESSES THE MPS WITHOUT IDLE TIME
      C ON THE BOTTLENECK MACHINE(S) AND WITHIN A TIME ON ALL OTHER MACHINES
      C LESS THAN OR EQUAL TO THAT REQUIRED ON THE BOTTLENECK MACHINE. MOREOVER,
      C THE FIRST ITEM IN A SATURATED SCHEDULE DOES NOT WAIT IN ANY INPUT
      C BUFFER LONGER THAN THE MANDATORY ONE TIME STEP.
      C IF REPEATED OVER AND OVER, A SATURATED SCHEDULE THUS GIVES MAXIMUM
      C STEADY-STATE OUTPUT, AS WELL AS A DEADBEAT TRANSIENT RESPONSE:
      C THE OUTPUT IS PERIODIC FROM THE INSTANT AT WHICH THE FIRST PART LEAVES
      C THE SYSTEM.
      C THE FOLLOWING LIMITATIONS APPLY TO THE SCHEDULES GENERATED:
      C     (1) EACH ITEM CAN OVERTAKE (AND PERHAPS BE AGAIN OVERTAKEN BY) AT
      C         MOST ONE OTHER PIECE. OVERTAKING CAN OCCUR ONLY WHEN BY-PASSING
      C         A MACHINE; ALL BUFFERS ARE SINGLE-POSITION AND FIFO.
      C     (2) THE PROGRAM DOES NOT CHECK FOR POSSIBLE CLASHES ON THE CONVEYOR
      C         WHEN A PART LEAVES A MACHINE. HOWEVER, IT WILL AVOID SIMULTANEOUS
      C         ARRIVALS AT A MACHINE.
      C IF A SATURATED SCHEDULE, RESTRICTED AS DESCRIBED, DOES NOT EXIST OR IS NOT
      C FOUND AFTER GENERATING MAXNOD NODES IN THE SEARCH TREE, THE LARGEST
      C SATURATED PARTIAL SCHEDULE FOUND IS PRINTED.
      C
      C
      C
            IMPLICIT INTEGER (A-Z)
            REAL RATE,XR
            LOGICAL DESC,CROSS,TEST
            COMMON//K,M,BIGLEV,REMJOB(9),MCPART(9),MAXPRT(9),SLACK(9),
           1 MACH(9),CRESPR(9),OTAKE(9,9),REACH(9,9),JOBTIM(9,9),TRTIME(9,9),
           2JOBRTE(9,9),FIN1(9),FIN2(9),STRT1(9),STRT2(9),START(99,9),FINISH(
           399,9),BUFF(99,9),PPERM(99),XPERM(99),NODE(99,9),TLAST(9),WLAST(9),
           4 MSCHED(99,9),SBUFF(99,9),SSTART(99,9),SFIN(99,9),SCHED(99,9),
           5 SPART(9),SPPER(99)
            DIMENSION RAND(9),NPART(9),NSCHED(500,9),NBUFF(500,9),JOBNO(9)
            NREAD=5
            NWRIT=6
      C
      C DATA INPUT: K,M,MAXNOD; TRTIME; JOBNO; JOBTIM. DATA CARDS SHOULD BE FOLLOWED
```

```
G1  RELEASE 2.0             MAIN                DATE = 79017          15/06/17
        C BY BLANK CARD FOR GRACEFUL STOP.
        C
            1 READ(NREAD,900)K,M,MAXNOD
              IF(K.EQ.0)CALL EXIT
              K1=K-1
              DO 10 LR=1,K1
                 LT=LR+1
           10 READ(NREAD,900) (TRTIME(LR,J),J=LT,K)
              READ(NREAD,900) (JOBNO(I),I=1,M)
              DO 20 LR=1,M
           20 READ(NREAD,900) (JOBTIM(LR,J),J=1,K)
          900 FORMAT(10I5)
        C
        C DEFINE PROBLEM PARAMETERS: N,REMJOB,JOBRTE,MACH,PERIOD,SLACK,MAXPRT,CRSPR,
        C OTAKE, REACH
        C
              N=0
              DO 30 I=1,M
                 REMJOB(I)=JOBNO(I)
                 N=N+JOBNO(I)
                 L=0
                 DO 32 J=1,K
                    IF(JOBTIM(I,J).EQ.0)GO TO 32
                    L=L+1
                    JOBRTE(I,L)=J
           32    CONTINUE
           30 MACH(I)=L
        C
              PERIOD=0
              DO 34 MC=1,K
                 SLACK(MC)=0
                 MAXPRT(MC)=0
                 DO 36 PRT=1,M
                    IF(JOBTIM(PRT,MC).EQ.0)GO TO 36
                    SLACK(MC)=SLACK(MC)+JOBNO(PRT)*JOBTIM(PRT,MC)
                    MAXPRT(MC)=MAXPRT(MC)+JOBNO(PRT)
           36    CONTINUE
                 IF(SLACK(MC).GT.PERIOD)PERIOD=SLACK(MC)
           34 CONTINUE
              DO 37 MC=1,K
           37 SLACK(MC)=PERIOD-SLACK(MC)
        C
              DO 60 I=1,M
                 CRSPR(I)=0
                 DO 60 J=1,M
           60 OTAKE(J,I)=0
        C
              DO 62 J1=1,M
                 NT=MACH(J1)-1
                 STRT1(1)=0
                 FIN1(1)=JOBTIM(J1,1)
                 DO 63 I=1,NT
                    MC=JOBRTE(J1,I)
```

```
                  MCN=JOBRTE(J1,I+1)
                  STRT1(MCN)=FIN1(MC)+TRTIME(MC,MCN)
                  FIN1(MCN)=STRT1(MCN)+JOBTIM(J1,MCN)
   63         CONTINUE
              L=0
              DO 64 J2=1,M
                  IF(J2.EQ.J1) GO TO 64
                  NT=MACH(J2)-1
                  STRT2(1)=FIN1(1)
                  FIN2(1)=JOBTIM(J2,1)+STRT2(1)
                  DO 66 I=1,NT
                      MC=JOBRTE(J2,I)
                      MCN=JOBRTE(J2,I+1)
                      STRT2(MCN)=FIN2(MC)+TRTIME(MC,MCN)
                      FIN2(MCN)=STRT2(MCN)+JOBTIM(J2,MCN)
                      IF(JOBTIM(J1,MCN).EQ.0) GO TO 66
                      IF(STRT2(MCN).GE.STRT1(MCN))GO TO 68
                      L=L+1
                      OTAKE(J1,L)=J2
                      GO TO 64
   68                 IF(STRT2(MCN).GE.FIN1(MCN)) GO TO 66
                      STRT2(MCN)=FIN1(MCN)
                      FIN2(MCN)=STRT2(MCN)+JOBTIM(J2,MCN)
   66             CONTINUE
   64         CONTINUE
              CRSPR(J1)=L
   62 CONTINUE
C
          DO 70 I=1,K1
              I1=I+1
              DO 70 J=I1,K
   70 REACH(I,J)=10000
C
          DO 80 I=1,K1
              DO 82 TYPE=1,M
                  I1=I+1
                  TIME=JOBTIM(TYPE,I)
                  IF(TIME.EQ.0)GO TO 82
                  MCL=I
                  DO 84 J=I1,K
                      TMP=JOBTIM(TYPE,J)
                      IF(TMP.EQ.0)GO TO 84
                      TIME=TIME+TRTIME(MCL,J)
                      IF(TIME.LT.REACH(I,J)) REACH(I,J)=TIME
                      TIME=TIME+TMP
                      MCL=J
   84             CONTINUE
   82         CONTINUE
   80 CONTINUE
C
C OUTPUT PROBLEM DATA
C
          DO 72 I=1,K
```

```
      72 RAND(I)=I
         WRITE(NWRIT,902) K
     902 FORMAT(1H1,'PROBLEM DATA:'/1H0,5X,'NO OF MACHINES:',I2)
         K1=K-1
         WRITE(NWRIT,904) (RAND(I),I=1,K1)
     904 FORMAT(1H ,5X,'TRAVEL TIMES BETWEEN MACHINES:',15X,'FROM'/1H ,
        140X,'TO',1X,8I4)
         WRITE(NWRIT,905)
     905 FORMAT(1H )
         DO 74 I=2,K
         I1=I-1
      74 WRITE(NWRIT,906)I,(TRTIME(L,I),L=1,I1)
     906 FORMAT(1H ,40X,I2,1X,8I4)
         WRITE(NWRIT,908)M,(RAND(I),I=1,K)
     908 FORMAT(1H ,5X,'NO OF ITEM TYPES:',I2/1H0,22X,'ITEM TYPE',4X,
        1'NO REQD',3X,'PROC. TIMES ON MC'/1H ,43X,8I4/)
         WRITE(NWRIT,905)
         DO 76 I=1,M
      76 WRITE(NWRIT,910)I,JOBNO(I),(JOBTIM(I,J),J=1,K)
     910 FORMAT(1H ,27X,I1,9X,I2,4X,8I4)
         WRITE(NWRIT,915) PERIOD,(SLACK(J),J=1,K)
     915 FORMAT(1H0,16X,'PERIOD =',I4,6X,'SLACK(J)',1X,8I4)
         WRITE(NWRIT,911)
     911 FORMAT(1H0,'ITEM TYPE  CRSPR  OTAKE')
         DO 86 PRT=1,M
            I1=CRSPR(PRT)
      86 WRITE(NWRIT,912)PRT,I1,(OTAKE(PRT,L),L=1,I1)
     912 FORMAT(1H ,I6,I8,4X,9I2)
         WRITE(NWRIT,917)
     917 FORMAT(1H ,'REACH(MC,MCN):')
         DO 88 MC=1,K1
         MC1=MC+1
      88 WRITE(NWRIT,916)(REACH(MC,MCN),MCN=MC1,K)
     916 FORMAT(1H ,9I4)
C
C INITIALIZE SEARCH
C
         DO 40 PC=1,N
            PPERM(PC)=0
            YPERM(PC)=0
            DO 42 MC=1,K
               MSCHED(PC,MC)=0
               START(PC,MC)=0
               FINISH(PC,MC)=0
               BUFF(PC,MC)=0
      42    CONTINUE
            DO 40 PRT=1,M
               NODE(PC,PRT)=0
      40 CONTINUE
         DO 44 MC=1,K
            TLAST(MC)=0
            WLAST(MC)=0
            STRT1(MC)=0
         STPT2(MC)=0
         FIN1(MC)=0
         FIN2(MC)=0
   44 MCPART(MC)=0
C
      MCOUNT=1
      BIGLEV=0
      LEVEL=1
      J1=1
C
  100 CALL NEXT(J1,J2,DESC,CROSS,LEVEL)
C
      IF(DESC)GO TO 200
      NODE(LEVEL,J1)=-1
C
C FOR FIRST CHOICE OF NEXT ITEM TYPE, SELECT ONE MOST IN ARREARS RELATIVE TO
C UNIFORM SCHEDULING
C
  102 RATE=2.
      TEST=.FALSE.
      DO 104 PRT=1,M
         IF(NODE(LEVEL,PRT).EQ.(-1))GO TO 104
         IF(REMJOB(PRT)) 106,106,108
  106    NODE(LEVEL,PRT)=-1
         GO TO 104
  108    TEST=.TRUE.
         XR=1.-FLOAT(REMJOB(PRT))/FLOAT(JOBNO(PRT))
         IF(XR.GT.RATE)GO TO 104
         J1=PRT
         RATE=XR
  104 CONTINUE
C
      IF(.NOT.TEST)GO TO 117
      DO 114 I=1,M
         IF(REMJOB(I)) 115,115,116
  115    NODE(LEVEL+1,I)=(-1)
         GO TO 114
  116    NODE(LEVEL+1,I)=0
  114 CONTINUE
      GO TO 100
C
C NO CONTINUATION OF THE CURRENT PARTIAL SCHEDULE IS POSSIBLE. IF LEVEL=1,
C SEARCH IS COMPLETE; NO SATURATED SCHEDULE EXISTS. IF LEVEL>1, BACKTRACK IN
C SEARCH TREE
C
  117 IF(LEVEL.EQ.1)GO TO 1000
  110 LEVEL=LEVEL-1
      JT=PPERM(LEVEL)
      NODE(LEVEL,JT)=-1
C
      IF(XPERM(LEVEL).EQ.0)GO TO 112
C
C THE LAST TWO CHAINS IN THE PARTIAL SCHEDULE FORM A CROSSED PAIR. BOTH ARE
```

```
      C REMOVED FROM THE SCHEDULE, BUT THE FIRST IS CHOSEN AS CANDIDATE FOR A NEW
      C CONTINUATION SINCE IT MIGHT EITHER BE FEASIBLE BY ITSELF OR CAN BE CROSSED
      C WITH ANOTHER CHAIN NOT YET ELIMINATED
      C
              CROSS=.TRUE.
              CALL BACKTR(CROSS,LEVEL)
              LEVEL=LEVEL-1
              J1=PPERM(LEVEL)
              GO TO 100
      C
      C SIMPLE BACKTRACK STEP: ONLY ONE CHAIN IS REMOVED
      C
        112 CROSS=.FALSE.
              CALL BACKTR(CROSS,LEVEL)
              GO TO 102
      C
      C CHAIN J1, OR, IF CROSS=TRUE, THE PAIR OF CROSSED CHAINS (J1,J2), FORM A
      C FEASIBLE CONTINUATION OF THE PARTIAL SCHEDULE
      C
        200 IF(CROSS)GO TO 202
              PPERM(LEVEL)=J1
              XPERM(LEVEL)=0
              LEVEL=LEVEL+1
              GO TO 204
        202 PPERM(LEVEL)=J1
              PPERM(LEVEL+1)=J2
              XPERM(LEVEL)=1
              XPERM(LEVEL+1)=1
              LEVEL=LEVEL+2
        204 CALL DSCND(J1,J2,CROSS,LEVEL)
      C
      C TEST IF OPTIMAL SOLUTION HAS BEEN FOUND
      C
              IF(LEVEL.GT.N)GO TO 2000
      C
              NCOUNT=NCOUNT+1
              IF(NCOUNT.GT.MAXNOD)GO TO 3000
      C
      C SEARCH IS NOT COMPLETE. BEFORE ATTEMPTING THE DETAILED CALCULATIONS OF
      C SEARCHING FOR A FEASIBLE CONTINUATION CHAIN, CHECK WHETHER IN THE CURRENT
      C PARTIAL SCHEDULE, EACH MACHINE CAN BE REACHED BY AT LEAST ONE PART TYPE
      C (NOT YET FULLY SCHEDULED), FROM THE IMMEDIATELY PRECEDING MACHINE FOR THAT
      C PART, WITHOUT EXCEEDING THE REMAINING AVAILABLE IDLE TIME. IF NOT, THE
      C PARTIAL SCHEDULE CANNOT HAVE A SATURATED CONTINUATION; BACKTRACK IN SEARCH
      C TREE (STMNT 110)
      C
              DO 210 MCN=2,K
                 IF((MCPART(MCN).EQ.MAXPRT(MCN)).OR.(TLAST(MCN).EQ.0))GO TO 210
                 DO 212 PRT=1,M
                    IF((JOBTIM(PRT,MCN).EQ.0).OR.(REMJOB(PRT).EQ.0))GO TO 212
                    MC=MCN-1
        215         TMP=JOBTIM(PRT,MC)
                    IF(TMP.GT.0)GO TO 214
```

```
C
      SUBROUTINE NEXT(J1,J2,TEST,CROSS,LEVEL)
C
C SUBROUTINE TESTS WHETHER THE CURRENT PARTIAL PERMUTATION (CPP):
C      <PPERM(1),...PPERM(LEVEL)>
C HAS FEASIBLE SATURATED CONTINUATIONS, OF THE FORMS
C         <PPERM(1),...PPERM(LEVEL),J1,J2>                      (A)
C WHERE J1 AND J2 FORM A 'CROSSED PAIR', OR
C         <PPERM(1),...PPERM(LEVEL),J1>                         (B)
C WHERE J1 IS ADDED TO CPP BY ITSELF. GIVEN J1, THE ROUTINE FIRST ATTEMPTS
C TO FIND J2 SUCH THAT (A) IS FEASIBLE. IF SUCCESSFUL, IT RETURNS TEST=.T.,
C CROSS=.T., J2 AND THE APPROPRIATE STARTING AND FINISHING TIMES OF CHAINS
C J1 AND J2 ON THE VARIOUS MACHINES IN ARRAYS STRT1,FIN1,STRT2,FIN2.
C IF NO J2 EXISTS FOR WHICH (A) IS FEASIBLE, THE ROUTINE TESTS WHETHER
C (B) IS FEASIBLE; IF SO,IT RETURNS TEST=.T., CROSS=.F. AND THE STARTING
C AND FINISHING TIMES OF CHAIN J1 IN STRT1,FIN1. IF NEITHER (A) NOR (B)
C ARE FEASIBLE, ROUTINE RETURNS TEST=.F.
C
C
      IMPLICIT INTEGER (A-Z)
      LOGICAL TEST,CROSS
      COMMON//K,M,BIGLEV,REMJOB(9),MCPART(9),MAXPRT(9),SLACK(9),
     1 MACH(9),CRSPR(9),OTAKE(9,9),REACH(9,9),JOBTIM(9,9),TRTIME(9,9),
     2JOBRTE(9,9),FIN1(9),FIN2(9),STRT1(9),STRT2(9),START(99,9),FINISH(
     399,9),BUFF(99,9),PPERM(99),XPERM(99),NODE(99,9),TLAST(9),WLAST(9),
     4 MSCHED(99,9),SBUFF(99,9),SSTART(99,9),SFIN(99,9),SCHED(99,9),
     5 SPART(9),SPPERM(99)
      DIMENSION GAP(9)
C
C DETERMINE IF THERE IS A CHAIN J2 WHICH OVERTAKES J1
C
      IF(CRSPR(J1).EQ.0)GO TO 130
      JXM=CRSPR(J1)
  100 DO 120 JX=1,JXM
         J2=OTAKE(J1,JX)
         IF(NODE(LEVEL+1,J2).EQ.(-1))GO TO 120
         IF(REMJOB(J2).GT.0)GO TO 200
         NODE(LEVEL+1,J2)=(-1)
  120 CONTINUE
C
C CHAIN J1 CANNOT BE PAIRED WITH AN OVERTAKING CHAIN; TEST WHETHER J1 CAN BE
C JOINED TO PARTIAL SCHEDULE BY ITSELF
C
  130 J2=0
  200 DO 101 I=1,K
         STRT1(I)=0
         FIN1(I)=0
         STRT2(I)=0
         FIN2(I)=0
  101 GAP(I)=0
      STRT1(1)=TLAST(1)
  198 FIN1(1)=STRT1(1)+JOBTIM(J1,1)
      IF(J2.EQ.0)GO TO 197
```

```
                 MC=MC-1
                 GO TO 215
   214           IF((TLAST(MC)+TMP+TRTIME(MC,MCN)-TLAST(MCN)).LE.SLACK(MCN))
     1                                                             GO TO 210
   212        CONTINUE
              GO TO 110
   210   CONTINUE
         GO TO 102
C
C OUTPUT ROUTINE
C
 1000 WRITE(NWRIT,919)NCOUNT
  919 FORMAT(1H0,' NO SATURATED SCHEDULE EXISTS'/1H ,I4,' NODES WERE GEN
     1ERATED')
 1001 WRITE(NWRIT,920)BIGLEV,(SPPER(J),J=1,BIGLEV)
  920 FORMAT(1H0,' LONGEST SATURATED PARTIAL SCHEDULE HAS',I5,' ITEMS:'/
     11H ,40I3/)
      N1=BIGLEV
      DO 1002 J=1,N1
      PPERM(J)=SPPER(J)
      DO 1002 I=1,K
      START(J,I)=SSTART(J,I)
      BUFF(J,I)=SBUFF(J,I)
      MSCHED(J,I)=SCHED(J,I)
 1002 FINISH(J,I)=SFIN(J,I)
      DO 1004 I=1,K
 1004 MCPART(I)=SPART(I)
      GO TO 2010
C
 3000 WRITE(NWRIT,940)NCOUNT
  940 FORMAT(1H0,' SEARCH EXCEEDS MAXIMUM NODE COUNT OF',I5)
      LEVEL=LEVEL-1
      IF(BIGLEV.GT.LEVEL)GO TO 1001
      WRITE(NWRIT,920)LEVEL,(PPERM(J),J=1,LEVEL)
      N1=LEVEL
      GO TO 2010
C
 2000 N1=N
      WRITE(NWRIT,922)NCOUNT,(PPERM(J),J=1,N)
  922 FORMAT(1H0,' FIRST OPTIMAL SCHEDULE FOUND AFTER GENERATING',I5,
     1' NODES:'/1H0,40I3)
C
 2010 NLAST=FINISH(N1,K)
      DO 2020 MC=1,K
         DO 2012 TYPE=1,M
 2012    NPART(TYPE)=0
         DO 2014 TIM=1,NLAST
            MSCHED(TIM,MC)=0
 2014    NBUFF(TIM,MC)=0
         LIM=MCPART(MC)
         DO 2016 PRT=1,LIM
            TYPE=MSCHED(PRT,MC)
            NPART(TYPE)=NPART(TYPE)+1
```

```
              X=100*TYPE+NPART(TYPE)
              MIN=START(PLT,MC)+1
              MAX=FINISH(PLT,MC)
              DO 2018 TIM=MIN,MAX
 2018         NSCHED(TIM,MC)=X
              IF(MC.EQ.1)GO TO 2016
              IF(BUFF(PLT,MC).EQ.0)GO TO 2016
              MAX=START(PLT,MC)
              MIN=MAX-BUFF(PLT,MC)+1
              DO 2019 TIM=MIN,MAX
 2019         NBUFF(TIM,MC)=X
 2016      CONTINUE
 2020 CONTINUE
C
      DO 2030 I=1,K
 2030 RAND(I)=I
      WRITE(NWFIT,930) (RAND(I),I=1,K)
  930 FORMAT(1H0,12X,'TIME',8(7X,'MC',I1))
      WRITE(NWFIT,932)
  932 FORMAT(1H0)
      DO 2032 I=1,NLAST
 2032 WRITE(NWFIT,934)I,(NBUFF(I,J),NSCHED(I,J),J=1,K)
  934 FORMAT(1H ,12X,I3,3X,8(3X,I3,1X,I3))
      GO TO 1
      END
```

```
G1  RELEASE 2.0              NEXT                DATE = 79017             15/05/17
          STRT2(1)=FIN1(1)
      199 FIN2(1)=STRT2(1)+JOBTIM(J2,1)
    C
      197 CROSS = .FALSE.
          M1L=1
          M2L=1
    C
          DO 999 MCN=2,K
             NMC=MCPART(MCN)
             TJOB1=JOBTIM(J1,MCN)
             IF(J2.GT.0)GO TO 201
             IF(TJOB1)999,999,203
      201    TJOB2=JOBTIM(J2,MCN)
             IF(TJOB1.GT.0)GO TO 202
             IF(TJOB2.EQ.0)GO TO 999
    C
    C ONLY CHAIN J2 VISITS MACHINE MCN
    C
             IF(TLAST(MCN).EQ.0)GO TO 204
             GAP(MCN)=FIN2(M2L)+TRTIME(M2L,MCN)-TLAST(MCN)
             IF(GAP(MCN).GT.0)GO TO 206
             DIFF=IABS(GAP(MCN))-WLAST(MCN)
             IF(DIFF.LT.0)GO TO 208
             IF(BUFF(NMC,MCN).EQ.0)GO TO 210
             IF(DIFF.EQ.0)GO TO 208
      212    STRT2(1)=STRT2(1)+DIFF
             GAP(1)=GAP(1)+DIFF
             GO TO 214
      210    STRT2(1)=STRT2(1)+1+DIFF
             GAP(1)=GAP(1)+1+DIFF
      214    IF(GAP(1)-SLACK(1))199,199,290
    C
      206    IF(GAP(MCN).GT.SLACK(MCN))GO TO 290
      204    STRT2(MCN)=FIN2(M2L)+TRTIME(M2L,MCN)
             GO TO 216
      208    STRT2(MCN)=TLAST(MCN)
      216    FIN2(MCN)=STRT2(MCN)+TJOB2
             M2L=MCN
             GO TO 999
    C
      202    IF(TJOB2.GT.0)GO TO 300
    C
    C ONLY CHAIN J1 VISITS MCN
    C
      203    IF(TLAST(MCN).EQ.0)GO TO 224
             GAP(MCN)=FIN1(M1L)+TRTIME(M1L,MCN)-TLAST(MCN)
             IF(GAP(MCN).GT.0)GO TO 226
             DIFF=IABS(GAP(MCN))-WLAST(MCN)
             IF(DIFF.LT.0)GO TO 228
             IF(BUFF(NMC,MCN).EQ.0)GO TO 230
             IF(DIFF.EQ.0)GO TO 228
      232    STRT1(1)=STRT1(1)+DIFF
             GAP(1)=GAP(1)+DIFF
```

```
            GO TO 234
      232   STRT1(1)=STRT1(1)+1+DIFF
            GAP(1)=GAP(1)+1+DIFF
      234   IF(GAP(1)-SLACK(1)) 198,198,290
C
      226   IF(GAP(MCN).GT.SLACK(MCN))GO TO 290
      224   STRT1(MCN)=FIN1(M1L)+TRTIME(M1L,MCN)
            GO TO 236
      228   STRT1(MCN)=TLAST(MCN)
      236   FIN1(MCN)=STRT1(MCN)+TJOB1
            M1L=MCN
            GO TO 999
C
C CHAINS J1 AND J2 ARE BOTH PROCESSED ON MACHINE MCN
C
      300   STRT1(MCN)=FIN1(M1L)+TRTIME(M1L,MCN)
            FIN1(MCN)=STRT1(MCN)+TJOB1
            STRT2(MCN)=FIN2(M2L)+TRTIME(M2L,MCN)
            FIN2(MCN)=STRT2(MCN)+TJOB2
C
            IF(STRT2(MCN).GE.STRT1(MCN))GO TO 400
C
C CHAIN J2 REACHES MACHINE MCN BEFORE J1
C
            CROSS=.TRUE.
            IF(STRT1(MCN).GE.TLAST(MCN))GO TO 302
C
C BOTH J2 AND J1 ARRIVE AT MCN BEFORE TLAST(MCN). SINCE ONLY ONE BUFFER
C POSITION IS AVAILABLE, LOADING OF J1 MUST BE DELAYED.
C
            SHIFT=TLAST(MCN)-STRT1(MCN)
            STRT1(1)=STRT1(1)+SHIFT
            GAP(1)=GAP(1)+SHIFT
            IF(GAP(1)-SLACK(1)) 198,198,290
C
      302   IF(STRT2(MCN).GE.TLAST(MCN))GO TO 312
C
C J2 ARRIVES BEFORE, AND J1 AT OR AFTER TLAST(MCN)
C
            DIFF=TLAST(MCN)-STRT2(MCN)-WLAST(MCN)
            IF(DIFF.LT.0)GO TO 304
            IF(BUFF(NMC,MCN).EQ.0)GO TO 210
            IF(DIFF.GT.0)GO TO 212
      304   STRT2(MCN)=TLAST(MCN)
            FIN2(MCN)=TLAST(MCN)+TJOB2
C
C BOTH J2 AND J1 ARRIVE AFTER TLAST
C
      312   DIFF=STRT1(MCN)-FIN2(MCN)
            IF(DIFF.GE.0)GO TO 314
            STRT1(MCN)=FIN2(MCN)
            FIN1(MCN)=STRT1(MCN)+TJOB1
      314   IF(TLAST(MCN).GT.0)GO TO 318
```

```
                GAP(MCN)=DIFF
                GO TO 319
        318     GAP(MCN)=STRT1(MCN)-TLAST(MCN)-TJOB2
      C
        319     IF(GAP(MCN).GT.SLACK(MCN))GO TO 290
                M1L=MCN
                M2L=MCN
                GO TO 999
      C
        400     IF(STRT2(MCN).GT.STRT1(MCN))GO TO 401
                STRT2(1)=STRT2(1)+1
                GAP(1)=GAP(1)+1
                IF(GAP(1)-SLACK(1))199,199,290
      C
      C CHAIN J1 REACHES MACHINE MCN BEFORE J2
      C
        401     IF(STRT2(MCN).GE.TLAST(MCN))GO TO 402
      C
      C BOTH J1 AND J2 ARRIVE AT MACHINE MCN BEFORE TLAST(MCN). SINCE ONLY ONE
      C BUFFER POSITION IS AVAILABLE, LOADING OF J2 MUST BE DELAYED.
      C
                SHIFT=TLAST(MCN)-STRT2(MCN)
                STRT2(1)=STRT2(1)+SHIFT
                GAP(1)=GAP(1)+SHIFT
                IF(GAP(1)-SLACK(1))199,199,290
      C
        402     IF(STRT1(MCN).GE.TLAST(MCN))GO TO 412
      C
      C J1 ARRIVES BEFORE, AND J2 AT OR AFTER TLAST(MCN)
      C
                DIFF=TLAST(MCN)-STRT1(MCN)-WLAST(MCN)
                IF(DIFF.LT.0)GO TO 404
                IF(BUFF(NMC,MCN).EQ.0)GO TO 230
                IF(DIFF.GT.0)GO TO 232
        404     STRT1(MCN)=TLAST(MCN)
                FIN1(MCN)=TLAST(MCN)+TJOB1
      C
      C BOTH J1 AND J2 ARRIVE AFTER TLAST(MCN)
      C
        412     DIFF=STRT2(MCN)-FIN1(MCN)
                IF(DIFF.GE.0)GO TO 414
                STRT2(MCN)=FIN1(MCN)
                FIN2(MCN)=STRT2(MCN)+TJOB2
        414     IF(TLAST(MCN).GT.0)GO TO 418
                GAP(MCN)=DIFF
                GO TO 419
        418     GAP(MCN)=STRT2(MCN)-TLAST(MCN)-TJOB1
      C
        419     IF(GAP(MCN).GT.SLACK(MCN))GO TO 290
                M1L=MCN
                M2L=MCN
        999 CONTINUE
      C
```

```
G1  RELEASE 2.0              NEXT              DATE = 79017           15/06/17
          TEST=.TRUE.
          RETURN
      290 IF(J2.EQ.0)GO TO 292
          NODE(LEVEL+1,J2)=(-1)
          GO TO 100
      292 TEST=.FALSE.
          RETURN
          END
      C
            SUBROUTINE DSCND(J1,J2,CROSS,LEVEL)
      C
      C THIS ROUTINE IS REACHED WHEN 'NEXT' HAS DETERMINED THAT J1 (IF CROSS=.F.)
      C OR THE CROSSED PAIR J1,J2 (IF CROSS=.T.) ARE FEASIBLE CONTINUATIONS OF
      C THE CPP. THE ROUTINE USES STRT1,STRT2,FIN1,FIN2 TO UPDATE ARRAYS BUFF,
      C MCPART,REMJOB,MSCHED,START,FINISH,TLAST,WLAST,SLACK; AND SETS
      C NODE(LEVEL,I), AND NODE(LEVEL+1,I) IF CROSS=.T.
      C
      C
            IMPLICIT INTEGER(A-Z)
            LOGICAL CROSS
            COMMON//K,M,BIGLEV,REMJOB(9),MCPART(9),MAXPRT(9),SLACK(9),
           1 MACH(9),CRSPP(9),OTAKF(9,9),REACH(9,9),JOBTIM(9,9),TRTIME(9,9),
           2JOBRTE(9,9),FIN1(9),FIN2(9),STRT1(9),STRT2(9),START(99,9),FINISH(
           399,9),BUFF(99,9),PPERM(99),XPERM(99),NODE(99,9),TLAST(9),WLAST(9),
           4 MSCHED(99,9),SBUFF(99,9),SSTART(99,9),SFIN(99,9),SCHED(99,9),
           5 SPART(9),SPPTR(99)
      C
            REMJOB(J1)=REMJOB(J1)-1
            IF(CROSS) REMJOB(J2)=REMJOB(J2)-1
            DO 2 J=1,M
               IF(REMJOB(J).GT.0)GO TO 4
               NODE(LEVEL,J)=-1
               NODE(LEVEL+1,J)=(-1)
               GO TO 2
          4    NODE(LEVEL,J)=0
               NODE(LEVEL+1,J)=0
          2 CONTINUE
      C
            IF(CROSS) GO TO 100
      C
      C ONLY CHAIN J1 IS ADDED TO PARTIAL SCHEDULE
      C
            NT=MACH(J1)
            MC=1
            DO 6 I=2,NT
               M1L=MC
               MC=JOBRTE(J1,I)
               DIFF=TLAST(MC)-FIN1(M1L)-TRTIME(M1L,MC)
               NMC=MCPART(MC)+1
               IF(DIFF.GT.0)GO TO 8
               BUFF(NMC,MC)=0
               GO TO 6
          8    BUFF(NMC,MC)=DIFF
          6 CONTINUE
            DO 10 I=1,NT
               MC=JOBRTE(J1,I)
               TJOB=JOBTIM(J1,MC)
               NMC=MCPART(MC)+1
               MCPART(MC)=NMC
               MSCHED(NMC,MC)=J1
               FINISH(NMC,MC)=FIN1(MC)
               START(NMC,MC)=FIN1(MC)-TJOB
               GAP=START(NMC,MC)-TLAST(MC)
               TLAST(MC)=FIN1(MC)
               ZLAST(MC)=TJOB
               IF(MCPART(MC).GT.1)SLACK(MC)=SLACK(MC)-GAP
   10      CONTINUE
           RETURN
C
C BOTH J1 AND J2 ARE ADDED TO PARTIAL SCHEDULE. LOOP ENDING AT STMNT 110
C UPDATES BUFFER OCCUPANCY
C
  100 M1L=1
           M2L=1
           DO 110 MC=2,K
               NMC=MCPART(MC)+1
               IF(JOBTIM(J1,MC).GT.0)GO TO 102
               IF(JOBTIM(J2,MC).EQ.0)GO TO 110
               DIFF=TLAST(MC)-FIN2(M2L)-TRTIME(M2L,MC)
               M2L=MC
               GO TO 108
  102          IF(JOBTIM(J2,MC).GT.0)GO TO 104
               DIFF=TLAST(MC)-FIN1(M1L)-TRTIME(M1L,MC)
               M1L=MC
               GO TO 108
  104          IF(STRT2(MC).LT.STRT1(MC))GO TO 106
               DIFF=TLAST(MC)-FIN1(M1L)-TRTIME(M1L,MC)
               DIFF1=FIN1(MC)-FIN2(M2L)-TRTIME(M2L,MC)
               GO TO 107
  106          DIFF=TLAST(MC)-FIN2(M2L)-TRTIME(M2L,MC)
               DIFF1=FIN2(MC)-FIN1(M1L)-TRTIME(M1L,MC)
  107          M1L=MC
               M2L=MC
               IF(DIFF1.GT.0)GO TO 109
               BUFF(NMC+1,MC)=0
               GO TO 108
  109          BUFF(NMC+1,MC)=DIFF1
  108          IF(DIFF.GT.0)GO TO 111
               BUFF(NMC,MC)=0
               GO TO 110
  111          BUFF(NMC,MC)=DIFF
  110      CONTINUE
C
           DO 200 MC=1,K
               NMC=MCPART(MC)+1
               IF(JOBTIM(J1,MC).GT.0)GO TO 210
               IF(JOBTIM(J2,MC).EQ.0)GO TO 200
C
               TJOB=JOBTIM(J2,MC)
               MCPART(MC)=NMC
               MSCHED(NMC,MC)=J2
               START(NMC,MC)=STRT2(MC)
               FINISH(NMC,MC)=FIN2(MC)
               GAP=STRT2(MC)-TLAST(MC)
```

```
                  TLAST(MC)=FIN2(MC)
                  WLAST(MC)=TJOB
                  GO TO 208
      C
        210       IF(JOBTIM(J2,MC).GT.0)GO TO 204
      C
                  TJOB=JOBTIM(J1,MC)
                  MCPART(MC)=NMC
                  MSCHED(NMC,MC)=J1
                  FINISH(NMC,MC)=FIN1(MC)
                  START(NMC,MC)=FIN1(MC)-TJOB
                  GAP=START(NMC,MC)-TLAST(MC)
                  TLAST(MC)=FIN1(MC)
                  WLAST(MC)=TJOB
        202       IF(MCPART(MC).GT.1)SLACK(MC)=SLACK(MC)-GAP
                  GO TO 200
      C
        204       MCPART(MC)=NMC+1
                  IF(STRT2(MC).LT.STRT1(MC))GO TO 206
      C
                  TJOB=JOBTIM(J2,MC)
                  MSCHED(NMC,MC)=J1
                  MSCHED(NMC+1,MC)=J2
                  START(NMC,MC)=FIN1(MC)-JOBTIM(J1,MC)
                  FINISH(NMC,MC)=FIN1(MC)
                  START(NMC+1,MC)=STRT2(MC)
                  FINISH(NMC+1,MC)=FIN2(MC)
                  GAP=FIN2(MC)-TLAST(MC)-TJOB-JOBTIM(J1,MC)
                  TLAST(MC)=FIN2(MC)
                  WLAST(MC)=TJOB
                  IF(MCPART(MC).EQ.2)GAP=STRT2(MC)-FIN1(MC)
                  GO TO 208
      C
        206       TJOB=JOBTIM(J1,MC)
                  MSCHED(NMC,MC)=J2
                  MSCHED(NMC+1,MC)=J1
                  START(NMC,MC)=STRT2(MC)
                  FINISH(NMC,MC)=FIN2(MC)
                  FINISH(NMC+1,MC)=FIN1(MC)
                  START(NMC+1,MC)=FIN1(MC)-TJOB
                  GAP=FIN1(MC)-TLAST(MC)-TJOB-JOBTIM(J2,MC)
                  TLAST(MC)=FIN1(MC)
                  WLAST(MC)=TJOB
                  IF(MCPART(MC).EQ.2)GAP=STRT1(MC)-FIN2(MC)
      C
        208       SLACK(MC)=SLACK(MC)-GAP
      C
        200  CONTINUE
             RETURN
             END
      C
            SUBROUTINE BACKTR(CROSS,LEVEL)
      C
      C THIS ROUTINE IS REACHED WHEN SUBROUTINE 'NEXT' RETURNS TEST=.F.,IE WHEN
      C CPP <PPERM(1),...PPERM(LEVEL)> HAS NO FEASIBLE SATURATED CONTINUATION.
      C IF LEVEL>BIGLEV, CPP IS STORED IN ARRAYS SPART, SPPER,SBUFF,SSTART,SFIN,
      C SCHED AND LEVEL REPLACES THE PREVIOUS VALUE OF BIGLEV.
      C IF CROSS=.F., ROUTINE DELETES PPERM(LEVEL) FROM THE CPP AND UPDATES
      C ARRAYS MCPART,REMJOB,TLAST,WLAST,SLACK. IF CROSS=.T., ROUTINE DELETES
      C BOTH PPERM(LEVEL) AND PPERM(LEVEL-1) AND UPDATES THE SAME ARRAYS.
      C
      C
            IMPLICIT INTEGER (A-Z)
            LOGICAL CROSS
            COMMON//K,M,BIGLEV,REMJOB(9),MCPART(9),MAXPRT(9),SLACK(9),
           1 MACH(9),CRSPR(9),OTAKE(9,9),REACH(9,9),JOBTIM(9,9),TRTIME(9,9),
           2JOBRTE(9,9),FIN1(9),FIN2(9),STRT1(9),STRT2(9),START(99,9),FINISH(
           399,9),BUFF(99,9),PPERM(99),XPERM(99),NODE(99,9),TLAST(9),WLAST(9),
           4 MSCHED(99,9),SBUFF(99,9),SSTART(99,9),SFIN(99,9),SCHED(99,9),
           5 SPART(9),SPPER(99)
      C
            IF(LEVEL.LE.BIGLEV)GO TO 10
            BIGLEV=LEVEL
            DO 1 MC=1,K
          1 SPART(MC)=MCPART(MC)
            DO 2 I=1,LEVEL
               SPPER(I)=PPERM(I)
               DO 2 J =1,K
                  SBUFF(I,J)=BUFF(I,J)
                  SSTART(I,J) =START(I,J)
                  SFIN(I,J)=FINISH(I,J)
                  SCHED(I,J) =MSCHED(I,J)
          2 CONTINUE
      C
         10 J2=PPERM(LEVEL)
            REMJOB(J2) =REMJOB(J2)+1
            IF(CROSS)GO TO 20
      C
      C ONLY CHAIN J2 IS DELETED FROM SCHEDULE
      C
            NT=MACH(J2)
            DO 12 J=1,NT
               MC=JOBRTE(J2,J)
               NMC=MCPART(MC)
               NMCL=NMC-1
               MCPART(MC)=NMCL
               IF(NMCL.GT.0)GO TO 14
               TLAST(MC)=0
               WLAST(MC)=0
               GO TO 12
         14    SLACK(MC)=SLACK(MC)+START(NMC,MC)-FINISH(NMCL,MC)
               TLAST(MC)=FINISH(NMCL,MC)
               WLAST(MC)=TLAST(MC)+START(NMCL,MC)
      12 CONTINUE
         RETURN
C
C PAIR OF CROSSED CHAINS ARE DELETED FROM SCHEDULE
C
      20 J1=PPERM(LEVEL-1)
         REMJOB(J1)=REMJOB(J1)+1
C
         DO 22 MC=1,K
            IF(JOBTIM(J1,MC).GT.0)GO TO 24
            IF(JOBTIM(J2,MC))22,22,26
      24    IF(JOBTIM(J2,MC))26,26,28
      26    NMC=MCPART(MC)
            NMCL=NMC-1
            MCPART(MC)=NMCL
            IF(NMCL.GT.0)GO TO 30
            TLAST(MC)=0
            WLAST(MC)=0
            GO TO 22
      30    SLACK(MC)=SLACK(MC)+START(NMC,MC)-FINISH(NMCL,MC)
            TLAST(MC) =FINISH(NMCL,MC)
            WLAST(MC)=TLAST(MC)-START(NMCL,MC)
            GO TO 22
C
      28    NMC =MCPART(MC)
            NMCL =NMC-2
            MCPART(MC)=NMCL
            IF(NMCL.EQ.0)GO TO 32
            SLACK(MC)=SLACK(MC)+START(NMC,MC)-FINISH(NMC-1,MC)
        1                  +START(NMC-1,MC)-FINISH(NMCL,MC)
            TLAST(MC)=FINISH(NMCL,MC)
            WLAST(MC)=TLAST(MC)-START(NMCL,MC)
            GO TO 22
      32    SLACK(MC)=SLACK(MC)+START(NMC,MC)-FINISH(NMC-1,MC)
            TLAST(MC)=0
            WLAST(MC) =0
      22 CONTINUE
         RETURN
         END
```

A3: <u>Sample Problems and Printouts of Solutions</u>

```
PROBLEM DATA:  EXAMPLE 1

     NO OF MACHINES: 6
     TRAVEL TIMES BETWEEN MACHINES:                   FROM
                                            TO    1    2    3    4    5

                                             2    5
                                             3   10    6
                                             4   16   12    7
                                             5   21   17   12    6
                                             6   25   21   16   10    5
     NO OF ITEM TYPES: 5

                        ITEM TYPE     NO REQD    PROC. TIMES ON MC
                                                  1    2    3    4    5    6

                            1            2        4    6    0    0   10    2
                            2            2        4    7    5    4    0    2
                            3            1        2    0   10    9    0    1
                            4            1        2    4    7    0    4    1
                            5            3        2    0    5    7    6    1

                    PERIOD =   42       SLACK(J)     16   12    0    4    0   29

ITEM TYPE   CRSPR   OTAKE
     1        0       0
     2        2       3 5
     3        0       0
     4        3       1 3 5
     5        1       1
REACH(MC,MCN):
   7   12   24   32   45
  10   25   23   38
  12   19   26
  13   14
   9

 FIRST OPTIMAL SCHEDULE FOUND AFTER GENERATING  20 NODES:

  1   5   3   1   4   5   2   5   2

             TIME        MC1        MC2        MC3        MC4        MC5        MC6


               1       0  101     0    0     0    0     0    0     0    0     0    0
               2       0  101     0    0     0    0     0    0     0    0     0    0
               3       0  101     0    0     0    0     0    0     0    0     0    0
               4       0  101     0    0     0    0     0    0     0    0     0    0
               5       0  501     0    0     0    0     0    0     0    0     0    0
               6       0  501     0    0     0    0     0    0     0    0     0    0
               7       0  301     0    0     0    0     0    0     0    0     0    0
               8       0  301     0    0     0    0     0    0     0    0     0    0
               9       0    0     0    0     0    0     0    0     0    0     0    0
              10       0    0     0  101     0    0     0    0     0    0     0    0
              11       0  102     0  101     0    0     0    0     0    0     0    0
              12       0  102     0  101     0    0     0    0     0    0     0    0
              13       0  102     0  101     0    0     0    0     0    0     0    0
              14       0  102     0  101     0    0     0    0     0    0     0    0
              15       0  401     0  101     0    0     0    0     0    0     0    0
              16       0  401     0    0     0    0     0    0     0    0     0    0
              17       0  502     0    0     0  501     0    0     0    0     0    0
              18       0  502     0    0     0  501     0    0     0    0     0    0
              19       0    0     0    0   301  501     0    0     0    0     0    0
```

PROBLEM DATA: <u>*EXAMPLE 2*</u>

```
     NO OF MACHINES: 6
     TRAVEL TIMES BETWEEN MACHINES:                   FROM
                                              TO    1   2   3   4   5

                                               2    5
                                               3   10   6
                                               4   16  12   7
                                               5   21  17  12   6
                                               6   25  21  16  10   5
     NO OF ITEM TYPES: 5

                       ITEM TYPE    NO REQD    PROC. TIMES ON MC
                                                1   2   3   4   5   6

                            1          5        4   6   0   0  10   2
                            2          5        4   7   5   4   0   2
                            3          3        2   0  10   9   0   1
                            4          2        2   4   7   0   4   1
                            5         10        2   0   5   7   6   1

                  PERIOD = 119      SLACK(J)   49  46   0   2   1  84

ITEM TYPE  CRSPR  OTAKE
     1       0     0
     2       2     3 5
     3       0     0
     4       3     1 3 5
     5       1     1
REACH(MC,MCN):
   7   12   24   32   45
  10   25   23   38
  12   19   26
  13   14
   9

 FIRST OPTIMAL SCHEDULE FOUND AFTER GENERATING 370 NODES:

  1  3  1  4  5  2  5  2  1  5  5  4  5  5  2  5  2  1  5  5  3  1  5  3  2

               TIME         MC1        MC2        MC3        MC4        MC5        MC6


                 1        0 101      0   0      0   0      0   0      0   0      0   0
                 2        0 101      0   0      0   0      0   0      0   0      0   0
                 3        0 101      0   0      0   0      0   0      0   0      0   0
                 4        0 101      0   0      0   0      0   0      0   0      0   0
                 5        0 301      0   0      0   0      0   0      0   0      0   0
                 6        0 301      0   0      0   0      0   0      0   0      0   0
                 7        0 102      0   0      0   0      0   0      0   0      0   0
                 8        0 102      0   0      0   0      0   0      0   0      0   0
                 9        0 102      0   0      0   0      0   0      0   0      0   0
                10        0 102      0 101      0   0      0   0      0   0      0   0
                11        0 401      0 101      0   0      0   0      0   0      0   0
                12        0 401      0 101      0   0      0   0      0   0      0   0
                13        0 501      0 101      0   0      0   0      0   0      0   0
                14        0 501      0 101      0   0      0   0      0   0      0   0
                15        0 201      0 101      0   0      0   0      0   0      0   0
                16        0 201      0 102      0   0      0   0      0   0      0   0
                17        0 201      0 102      0 301      0   0      0   0      0   0
                18        0 201    401 102      0 301      0   0      0   0      0   0
                19        0   0    401 102      0 301      0   0      0   0      0   0
```

20	0	0	0	102	301	501	0	0	0	0	0	0
21	0	0	0	102	301	501	0	0	0	0	0	0
22	0	201	401	102	0	301	0	0	0	0	0	0
23	0	201	401	102	0	301	0	0	0	0	0	0
24	0	201	401	102	0	301	0	0	0	0	0	0
25	0	201	401	102	0	301	0	0	0	0	0	0
26	0	503	0	401	0	301	0	0	0	0	0	0
27	0	503	0	401	0	301	0	0	0	0	0	0
28	0	202	0	401	0	301	0	0	0	0	0	0
29	0	202	0	401	502	301	0	501	0	0	0	0
30	0	202	0	0	502	301	0	501	0	0	0	0
31	0	202	0	201	502	301	0	501	0	0	0	0
32	0	0	0	201	0	502	0	501	0	0	0	0
33	0	0	0	201	0	502	0	501	0	101	0	0
34	0	0	0	201	0	502	0	501	0	101	0	0
35	0	0	0	201	0	502	0	501	0	101	0	0
36	0	0	0	201	401	502	0	0	0	101	0	0
37	0	0	202	201	0	401	0	0	0	101	0	0
38	0	0	0	202	503	401	0	0	0	101	0	0
39	0	0	0	202	503	401	0	301	0	101	0	0
40	0	0	0	202	503	401	0	301	0	101	0	0
41	0	0	0	202	503	401	0	301	0	101	0	0
42	0	0	0	202	503	401	0	301	501	101	0	0
43	0	0	0	202	503	401	0	301	102	501	0	0
44	0	0	0	202	201	503	502	301	102	501	0	0
45	0	0	0	0	201	503	502	301	102	501	0	0
46	0	0	0	0	201	503	502	301	102	501	0	0
47	0	0	0	0	201	503	502	301	102	501	0	0
48	0	0	0	0	201	503	0	502	102	501	0	101
49	0	0	0	0	0	201	0	502	0	102	0	101
50	0	0	0	0	0	201	0	502	0	102	0	0
51	0	0	0	0	202	201	0	502	0	102	0	0
52	0	0	0	0	202	201	0	502	0	102	0	0
53	0	0	0	0	202	201	0	502	0	102	0	0
54	0	0	0	0	0	202	0	502	0	102	0	501
55	0	0	0	0	0	202	0	0	0	102	0	0
56	0	0	0	0	0	202	0	503	401	102	0	0
57	0	0	0	0	0	202	0	503	401	102	0	0
58	0	0	0	0	0	202	0	503	401	102	0	301
59	0	0	0	0	0	0	0	503	0	401	0	0
60	0	0	0	0	0	0	0	503	0	401	0	0
61	0	0	0	0	0	0	201	503	502	401	0	0
62	0	0	0	0	0	0	201	503	502	401	0	0
63	0	0	0	0	0	0	0	201	0	502	0	0
64	0	0	0	0	0	0	0	201	0	502	0	102
65	0	0	0	0	0	0	0	201	0	502	0	102
66	0	0	0	0	0	0	202	201	0	502	0	0
67	0	0	0	0	0	0	0	202	0	502	0	0
68	0	0	0	0	0	0	0	202	0	502	0	401
69	0	0	0	0	0	0	0	202	0	503	0	0
70	0	0	0	0	0	0	0	202	0	503	0	0
71	0	0	0	0	0	0	0	0	0	503	0	0
72	0	0	0	0	0	0	0	0	0	503	0	0
73	0	0	0	0	0	0	0	0	0	503	0	0
74	0	0	0	0	0	0	0	0	0	503	0	502
75	0	0	0	0	0	0	0	0	0	0	0	0
76	0	0	0	0	0	0	0	0	0	0	0	0
77	0	0	0	0	0	0	0	0	0	0	0	201
78	0	0	0	0	0	0	0	0	0	0	0	201
79	0	0	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0	0	0	503
81	0	0	0	0	0	0	0	0	0	0	0	202
82	0	0	0	0	0	0	0	0	0	0	0	202

20	0	0	401	102	0	301	0	0	0	0	0	0
21	0	502	401	102	0	301	0	0	0	0	0	0
22	0	502	0	401	0	301	0	0	0	0	0	0
23	0	202	0	401	0	301	0	0	0	0	0	0
24	0	202	201	401	0	301	0	0	0	0	0	0
25	0	202	201	401	501	301	0	0	0	0	0	0
26	0	202	0	201	501	301	0	0	0	0	0	0
27	0	0	0	201	0	501	0	0	0	0	0	0
28	0	0	0	201	0	501	0	0	0	0	0	0
29	0	0	0	201	0	501	0	0	0	0	0	0
30	0	0	0	201	0	501	0	0	0	0	0	0
31	0	0	0	201	0	501	0	0	0	0	0	0
32	0	0	202	201	0	401	0	0	0	0	0	0
33	0	103	0	202	502	401	0	0	0	101	0	0
34	0	103	0	202	502	401	0	301	0	101	0	0
35	0	103	0	202	502	401	0	301	0	101	0	0
36	0	103	0	202	502	401	0	301	0	101	0	0
37	0	503	0	202	502	401	0	301	0	101	0	0
38	0	503	0	202	502	401	0	301	0	101	0	0
39	0	0	0	202	201	502	501	301	102	101	0	0
40	0	0	0	0	201	502	501	301	102	101	0	0
41	0	0	0	0	201	502	501	301	102	101	0	0
42	0	504	0	103	201	502	501	301	102	101	0	0
43	0	504	0	103	201	502	0	501	0	102	0	0
44	0	0	0	103	0	201	0	501	0	102	0	0
45	0	0	0	103	0	201	0	501	0	102	0	0
46	0	0	0	103	202	201	0	501	0	102	0	0
47	0	402	0	103	202	201	0	501	0	102	0	0
48	0	402	0	0	202	201	0	501	0	102	0	101
49	0	505	0	0	503	202	0	501	0	102	0	101
50	0	505	0	0	503	202	0	0	0	102	0	0
51	0	0	0	0	503	202	0	502	401	102	0	0
52	0	0	0	0	503	202	0	502	401	102	0	0
53	0	0	0	0	503	202	0	502	0	401	0	301
54	0	0	0	402	504	503	0	502	0	401	0	0
55	0	0	0	402	504	503	0	502	0	401	0	0
56	0	0	0	402	504	503	201	502	501	401	0	0
57	0	506	0	402	504	503	201	502	0	501	0	0
58	0	506	0	0	504	503	0	201	0	501	0	102
59	0	203	0	0	0	504	0	201	0	501	0	102
60	0	203	0	0	0	504	0	201	0	501	0	0
61	0	203	0	0	505	504	202	201	0	501	0	0
62	0	203	0	0	505	504	0	202	0	501	0	401
63	0	0	0	0	505	504	0	202	0	0	0	0
64	0	507	0	0	402	505	0	202	0	502	0	0
65	0	507	0	0	402	505	0	202	103	502	0	0
66	0	204	0	0	402	505	0	503	103	502	0	0
67	0	204	0	0	402	505	0	503	103	502	0	0
68	0	204	0	203	402	505	0	503	103	502	0	501
69	0	204	0	203	506	402	0	503	103	502	0	0
70	0	0	0	203	506	402	0	503	0	103	0	0
71	0	0	0	203	506	402	504	503	0	103	0	0
72	0	0	0	203	506	402	504	503	0	103	0	201
73	0	0	0	203	506	402	0	504	0	103	0	201
74	0	0	0	203	506	402	0	504	0	103	0	0
75	0	0	0	204	506	402	0	504	0	103	0	502
76	0	0	0	204	507	506	505	504	0	103	0	202
77	0	104	0	204	507	506	505	504	0	103	0	202
78	0	104	0	204	507	506	505	504	0	103	0	0
79	0	104	0	204	507	506	505	504	503	103	0	0
80	0	104	0	204	507	506	0	505	0	503	0	0
81	0	508	0	204	203	507	0	505	0	503	0	0
82	0	508	0	0	203	507	0	505	0	503	0	0
83	0	0	0	0	203	507	0	505	0	503	0	0
84	0	509	0	0	203	507	0	505	0	503	0	0
85	0	509	0	0	203	507	0	505	0	503	0	103

```
 86     0    0     0  104     0  203     0  505     0  504     0  103
 87     0    0     0  104     0  203     0    0     0  504     0    0
 88     0    0     0  104   204  203     0  506   402  504     0    0
 89     0  302     0  104   204  203     0  506   402  504     0    0
 90     0  302     0  104   204  203     0  506   402  504     0    0
 91     0    0     0  104     0  204     0  506   402  504     0  503
 92     0    0     0    0     0  204     0  506     0  402     0    0
 93     0    0     0    0   508  204   507  506   505  402     0    0
 94     0    0     0    0   508  204   507  506   505  402     0    0
 95     0    0     0    0   508  204     0  507   505  402     0    0
 96     0    0     0    0   509  508     0  507     0  505     0    0
 97     0    0     0    0   509  508     0  507     0  505     0  504
 98     0    0     0    0   509  508   203  507     0  505     0    0
 99     0  105     0    0   509  508   203  507     0  505     0    0
100     0  105     0    0   509  508   203  507     0  505     0    0
101     0  105     0    0   302  509   203  507   506  505     0  402
102     0  105     0    0   302  509     0  203     0  506     0    0
103     0  510     0    0   302  509   204  203     0  506     0    0
104     0  510     0    0   302  509   204  203     0  506     0    0
105     0  303     0    0   302  509   204  203     0  506     0    0
106     0  303     0    0     0  302     0  204     0  506     0    0
107     0  205     0    0     0  302     0  204     0  506     0  505
108     0  205     0  105     0  302   508  204     0  507     0    0
109     0  205     0  105     0  302   508  204   104  507     0    0
110     0  205     0  105     0  302     0  508   104  507     0    0
111     0    0     0  105     0  302     0  508   104  507     0    0
112     0    0     0  105     0  302     0  508   104  507     0    0
113     0    0     0  105     0  302   509  508   104  507     0  506
114     0    0     0    0     0  302   509  508     0  104     0    0
115     0    0     0    0   510  302   509  508     0  104     0    0
116     0    0     0  205     0  510   509  508     0  104     0  203
117     0    0     0  205   303  510     0  509     0  104     0  203
118     0    0     0  205   303  510     0  509     0  104     0    0
119     0    0     0  205   303  510     0  509     0  104     0  507
120     0    0     0  205   303  510     0  509     0  104     0  204
121     0    0     0  205     0  303     0  509     0  104     0  204
122     0    0     0  205     0  303     0  509     0  104     0    0
123     0    0     0    0     0  303   302  509   508  104     0    0
124     0    0     0    0     0  303     0  302     0  508     0    0
125     0    0     0    0     0  303     0  302     0  508     0    0
126     0    0     0    0     0  303     0  302     0  508     0    0
127     0    0     0    0     0  303     0  302     0  508     0    0
128     0    0     0    0     0  303   510  302     0  508     0    0
129     0    0     0    0   205  303   510  302     0  508     0  104
130     0    0     0    0   205  303   510  302     0  509     0  104
131     0    0     0    0     0  205   510  302   105  509     0    0
132     0    0     0    0     0  205   510  302   105  509     0    0
133     0    0     0    0     0  205     0  510   105  509     0    0
134     0    0     0    0     0  205     0  510   105  509     0    0
135     0    0     0    0     0  205     0  510   105  509     0  508
136     0    0     0    0     0    0     0  510     0  105     0    0
137     0    0     0    0     0    0     0  510     0  105     0    0
138     0    0     0    0     0    0   303  510     0  105     0    0
139     0    0     0    0     0    0   303  510     0  105     0    0
140     0    0     0    0     0    0     0  303     0  105     0    0
141     0    0     0    0     0    0     0  303     0  105     0  509
142     0    0     0    0     0    0     0  303     0  105     0    0
143     0    0     0    0     0    0   205  303     0  105     0  302
144     0    0     0    0     0    0   205  303     0  105     0    0
145     0    0     0    0     0    0   205  303     0  105     0    0
146     0    0     0    0     0    0   205  303     0  510     0    0
147     0    0     0    0     0    0   205  303     0  510     0    0
148     0    0     0    0     0    0   205  303     0  510     0    0
149     0    0     0    0     0    0     0  205     0  510     0    0
150     0    0     0    0     0    0     0  205     0  510     0    0
151     0    0     0    0     0    0     0  205     0  510     0  105
```

152	0	0	0	0	0	0	0	205	0	0	0	105
153	0	0	0	0	0	0	0	0	0	0	0	0
154	0	0	0	0	0	0	0	0	0	0	0	0
155	0	0	0	0	0	0	0	0	0	0	0	0
156	0	0	0	0	0	0	0	0	0	0	0	0
157	0	0	0	0	0	0	0	0	0	0	0	510
158	0	0	0	0	0	0	0	0	0	0	0	0
159	0	0	0	0	0	0	0	0	0	0	0	303
160	0	0	0	0	0	0	0	0	0	0	0	0
161	0	0	0	0	0	0	0	0	0	0	0	0
162	0	0	0	0	0	0	0	0	0	0	0	0
163	0	0	0	0	0	0	0	0	0	0	0	205
164	0	0	0	0	0	0	0	0	0	0	0	205

```
PROBLEM DATA:  EXAMPLE 3

     NO OF MACHINES: 6
     TRAVEL TIMES BETWEEN MACHINES:                           FROM
                                                  TO     1    2    3    4    5

                                                   2     5
                                                   3    10    6
                                                   4    16   12    7
                                                   5    21   17   12    6
                                                   6    25   21   16   10    5
     NO OF ITEM TYPES: 5

                                  ITEM TYPE     NO REQD    PROC. TIMES ON MC
                                                             1    2    3    4    5    6

                                      1            5         4    6    0    0   10    2
                                      2            5         4    7    5    4    0    2
                                      3            3         2    0   10    9    0    1
                                      4            2         2    4    7    0    1    1
                                      5           10         1    0    0    7    6    1

                             PERIOD =  117       SLACK(J)   57   44   48    0    5   82

ITEM TYPE   CRSPR   OTAKE
     1          0      0
     2          2      3 5
     3          2     4 5
     4          3     1 3 5
     5          0     0
REACH(MC,MCN) :
   7   12   17   30   41
  10   25   23   35
  12   19   25
  13   14
   6

 FIRST OPTIMAL SCHEDULE FOUND AFTER GENERATING26961  NODES:

  1   3   4   1   5   5   2   5   2   1   5   5   2   5   2   1   5   4   5   2   5   1   3   5   3

                  TIME          MC1          MC2          MC3          MC4          MC5          MC6


                    1         0  101      0    0      0    0      0    0      0    0      0    0
                    2         0  101      0    0      0    0      0    0      0    0      0    0
                    3         0  101      0    0      0    0      0    0      0    0      0    0
                    4         0  101      0    0      0    0      0    0      0    0      0    0
                    5         0  301      0    0      0    0      0    0      0    0      0    0
                    6         0  301      0    0      0    0      0    0      0    0      0    0
                    7         0  401      0    0      0    0      0    0      0    0      0    0
                    8         0  401      0    0      0    0      0    0      0    0      0    0
                    9         0    0      0    0      0    0      0    0      0    0      0    0
                   10         0    0      0  101      0    0      0    0      0    0      0    0
                   11         0    0      0  101      0    0      0    0      0    0      0    0
                   12         0    0      0  101      0    0      0    0      0    0      0    0
                   13         0    0      0  101      0    0      0    0      0    0      0    0
                   14         0    0    401  101      0    0      0    0      0    0      0    0
                   15         0  102    401  101      0    0      0    0      0    0      0    0
                   16         0  102      0  401      0    0      0    0      0    0      0    0
                   17         0  102      0  401      0  301      0    0      0    0      0    0
                   18         0  102      0  401      0  301      0    0      0    0      0    0
                   19         0  501      0  401      0  301      0    0      0    0      0    0
```

20	0	0	0	0	0	301	0	0	0	0	0	0
21	0	0	0	0	0	301	0	0	0	0	0	0
22	0	0	0	0	0	301	0	0	0	0	0	0
23	0	0	0	0	0	301	0	0	0	0	0	0
24	0	0	0	102	0	301	0	0	0	0	0	0
25	0	0	0	102	0	301	0	0	0	0	0	0
26	0	502	0	102	401	301	0	0	0	0	0	0
27	0	201	0	102	0	401	0	0	0	0	0	0
28	0	201	0	102	0	401	0	0	0	0	0	0
29	0	201	0	102	0	401	0	0	0	0	0	0
30	0	201	0	0	0	401	0	0	0	0	0	0
31	0	0	0	0	0	401	0	0	0	0	0	0
32	0	0	0	0	0	401	0	0	0	0	0	0
33	0	503	0	0	0	401	0	0	0	101	0	0
34	0	202	0	0	0	0	0	301	0	101	0	0
35	0	202	0	0	0	0	0	301	0	101	0	0
36	0	202	0	201	0	0	501	301	0	101	0	0
37	0	202	0	201	0	0	501	301	0	101	0	0
38	0	103	0	201	0	0	501	301	0	101	0	0
39	0	103	0	201	0	0	501	301	0	101	0	0
40	0	103	0	201	0	0	501	301	0	101	0	0
41	0	103	0	201	0	0	501	301	0	101	0	0
42	0	0	0	201	0	0	501	301	0	101	0	0
43	0	0	0	202	0	0	502	501	0	0	0	0
44	0	0	0	202	0	0	502	501	0	0	0	0
45	0	0	0	202	0	0	502	501	0	0	0	0
46	0	0	0	202	0	0	502	501	0	401	0	0
47	0	0	103	202	0	0	502	501	0	102	0	0
48	0	0	103	202	0	0	502	501	0	102	0	101
49	0	0	103	202	0	201	502	501	0	102	0	101
50	0	0	0	103	0	201	503	502	0	102	0	0
51	0	0	0	103	0	201	503	502	0	102	0	0
52	0	504	0	103	0	201	503	502	0	102	0	401
53	0	0	0	103	0	201	503	502	0	102	0	301
54	0	0	0	103	0	0	503	502	0	102	0	0
55	0	505	0	103	0	0	503	502	0	102	0	0
56	0	203	0	0	0	202	503	502	501	102	0	0
57	0	203	0	0	0	202	0	503	0	501	0	0
58	0	203	0	0	0	202	0	503	0	501	0	0
59	0	203	0	0	0	202	0	503	0	501	0	0
60	0	0	0	0	0	202	0	503	0	501	0	0
61	0	0	0	0	0	0	201	503	0	501	0	0
62	0	506	0	0	0	0	201	503	0	501	0	102
63	0	204	0	0	0	0	201	503	0	502	0	102
64	0	204	0	0	0	0	0	201	0	502	0	0
65	0	204	0	203	0	0	0	201	0	502	0	0
66	0	204	0	203	0	0	0	201	0	502	0	0
67	0	104	0	203	0	0	0	201	0	502	0	0
68	0	104	0	203	0	0	0	202	0	502	0	501
69	0	104	0	203	0	0	504	202	0	0	0	0
70	0	104	0	203	0	0	504	202	0	503	0	0
71	0	0	0	203	0	0	504	202	0	503	0	0
72	0	0	0	204	0	0	505	504	0	503	0	0
73	0	0	0	204	0	0	505	504	103	503	0	0
74	0	0	0	204	0	0	505	504	103	503	0	502
75	0	0	0	204	0	0	505	504	103	503	0	0
76	0	0	104	204	0	0	505	504	0	103	0	0
77	0	0	104	204	0	0	505	504	0	103	0	0
78	0	0	104	204	0	203	505	504	0	103	0	201
79	0	0	0	104	0	203	506	505	0	103	0	201
80	0	0	0	104	0	203	506	505	0	103	0	0
81	0	507	0	104	0	203	506	505	0	103	0	503
82	0	402	0	104	0	203	506	505	0	103	0	202
83	0	402	0	104	0	0	506	505	0	103	0	202
84	0	508	0	104	0	0	506	505	0	103	0	0
85	0	205	0	0	0	204	506	505	504	103	0	0

86	0	205	0	0	0	204	0	506	0	504	0	0
87	0	205	0	0	0	204	0	506	0	504	0	0
88	0	205	0	0	0	204	0	506	0	504	0	0
89	0	0	0	402	0	204	0	506	0	504	0	0
90	0	0	0	402	0	0	203	506	0	504	0	0
91	0	509	0	402	0	0	203	506	0	504	0	103
92	0	0	0	402	0	0	203	506	0	505	0	103
93	0	0	0	0	0	0	0	203	0	505	0	0
94	0	0	0	205	0	0	0	203	0	505	0	0
95	0	0	0	205	0	0	0	203	0	505	0	0
96	0	0	0	205	0	0	0	203	0	505	0	0
97	0	105	0	205	0	0	0	204	0	505	0	504
98	0	105	0	205	0	0	507	204	0	0	0	0
99	0	105	0	205	0	402	507	204	0	506	0	0
100	0	105	0	205	0	402	507	204	0	506	0	0
101	0	302	0	0	0	402	508	507	0	506	0	0
102	0	302	0	0	0	402	508	507	104	506	0	0
103	0	0	0	0	0	402	508	507	104	506	0	505
104	0	0	0	0	0	402	508	507	104	506	0	0
105	0	510	0	0	0	402	508	507	0	104	0	0
106	0	303	0	105	0	0	508	507	0	104	0	0
107	0	303	0	105	0	205	508	507	0	104	0	203
108	0	0	0	105	0	205	509	508	0	104	0	203
109	0	0	0	105	0	205	509	508	0	104	0	0
110	0	0	0	105	0	205	509	508	0	104	0	506
111	0	0	0	105	0	205	509	508	0	104	0	204
112	0	0	0	0	0	0	509	508	0	104	0	204
113	0	0	0	0	0	302	509	508	0	104	0	0
114	0	0	0	0	0	302	509	508	507	104	0	0
115	0	0	0	0	0	302	0	509	0	507	0	0
116	0	0	0	0	0	302	0	509	0	507	0	0
117	0	0	0	0	0	302	0	509	0	507	0	0
118	0	0	0	0	303	302	0	509	402	507	0	0
119	0	0	0	0	303	302	205	509	402	507	0	0
120	0	0	0	0	303	302	205	509	402	507	0	104
121	0	0	0	0	303	302	205	509	508	402	0	104
122	0	0	0	0	303	302	510	205	0	508	0	0
123	0	0	0	0	0	303	510	205	0	508	0	0
124	0	0	0	0	0	303	510	205	0	508	0	0
125	0	0	0	0	0	303	510	205	0	508	0	0
126	0	0	0	0	0	303	0	510	0	508	0	507
127	0	0	0	0	0	303	0	510	0	508	0	402
128	0	0	0	0	0	303	0	510	0	509	0	0
129	0	0	0	0	0	303	0	510	105	509	0	0
130	0	0	0	0	0	303	302	510	105	509	0	0
131	0	0	0	0	0	303	302	510	105	509	0	0
132	0	0	0	0	0	303	302	510	105	509	0	0
133	0	0	0	0	0	0	0	302	105	509	0	508
134	0	0	0	0	0	0	0	302	0	105	0	0
135	0	0	0	0	0	0	0	302	0	105	0	0
136	0	0	0	0	0	0	0	302	0	105	0	205
137	0	0	0	0	0	0	0	302	0	105	0	205
138	0	0	0	0	0	0	0	302	0	105	0	0
139	0	0	0	0	0	0	0	302	510	105	0	509
140	0	0	0	0	0	0	303	302	510	105	0	0
141	0	0	0	0	0	0	303	302	510	105	0	0
142	0	0	0	0	0	0	0	303	510	105	0	0
143	0	0	0	0	0	0	0	303	510	105	0	0
144	0	0	0	0	0	0	0	303	0	510	0	0
145	0	0	0	0	0	0	0	303	0	510	0	0
146	0	0	0	0	0	0	0	303	0	510	0	0
147	0	0	0	0	0	0	0	303	0	510	0	0
148	0	0	0	0	0	0	0	303	0	510	0	0
149	0	0	0	0	0	0	0	303	0	510	0	105
150	0	0	0	0	0	0	0	303	0	0	0	105
151	0	0	0	0	0	0	0	0	0	0	0	0

152	0	0	0	0	0	0	0	0	0	0	0	302
153	0	0	0	0	0	0	0	0	0	0	0	0
154	0	0	0	0	0	0	0	0	0	0	0	0
155	0	0	0	0	0	0	0	0	0	0	0	510
156	0	0	0	0	0	0	0	0	0	0	0	0
157	0	0	0	0	0	0	0	0	0	0	0	0
158	0	0	0	0	0	0	0	0	0	0	0	0
159	0	0	0	0	0	0	0	0	0	0	0	0
160	0	0	0	0	0	0	0	0	0	0	0	0
161	0	0	0	0	0	0	0	0	0	0	0	303

```
PROBLEM DATA:  EXAMPLE 4

     NO OF MACHINES: 6
     TRAVEL TIMES BETWEEN MACHINES:                 FROM
                                          TO    1   2   3   4   5

                                           2    5
                                           3   10   6
                                           4   16  12   7
                                           5   21  17  12   6
                                           6   25  21  16  10   5
     NO OF ITEM TYPES: 5

                    ITEM TYPE    NO REQD    PROC. TIMES ON MC
                                             1   2   3   4   5   6

                        1           5        4   6   0   0  10   2
                        2           5        4  20   5  10   0   2
                        3           3        2   0  10  15   0   2
                        4           2        2   4  10   0   4   1
                        5           6        3   0   5   8  10   1

                 PERIOD = 143      SLACK(J)  75   5  38   0  25 109

ITEM TYPE   CRSPR   OTAKE
     1        0     0
     2        2     3 5
     3        2     1 4
     4        3     1 3 5
     5        1     1
REACH(MC,MCN):
   7   12   25   32   47
  10   38   23   38
  12   22   31
  14   20
   9

 FIRST OPTIMAL SCHEDULE FOUND AFTER GENERATING8812 NODES:

  2  3  1  1  5  4  3  2  5  2  5  2  5  1  5  4  3  2  1  1  5

          TIME        MC1        MC2        MC3        MC4        MC5        MC6

            1       0  201     0    0     0    0     0    0     0    0     0    0
            2       0  201     0    0     0    0     0    0     0    0     0    0
            3       0  201     0    0     0    0     0    0     0    0     0    0
            4       0  201     0    0     0    0     0    0     0    0     0    0
            5       0  301     0    0     0    0     0    0     0    0     0    0
            6       0  301     0    0     0    0     0    0     0    0     0    0
            7       0    0     0    0     0    0     0    0     0    0     0    0
            8       0    0     0    0     0    0     0    0     0    0     0    0
            9       0    0     0    0     0    0     0    0     0    0     0    0
           10       0    0     0  201     0    0     0    0     0    0     0    0
           11       0    0     0  201     0    0     0    0     0    0     0    0
           12       0    0     0  201     0    0     0    0     0    0     0    0
           13       0    0     0  201     0    0     0    0     0    0     0    0
           14       0    0     0  201     0    0     0    0     0    0     0    0
           15       0    0     0  201     0    0     0    0     0    0     0    0
           16       0    0     0  201     0    0     0    0     0    0     0    0
           17       0    0     0  201     0  301     0    0     0    0     0    0
           18       0    0     0  201     0  301     0    0     0    0     0    0
           19       0    0     0  201     0  301     0    0     0    0     0    0
```

20	0	0	0	201	0	301	0	0	0	0	0	0
21	0	0	0	201	0	301	0	0	0	0	0	0
22	0	0	0	201	0	301	0	0	0	0	0	0
23	0	101	0	201	0	301	0	0	0	0	0	0
24	0	101	0	201	0	301	0	0	0	0	0	0
25	0	101	0	201	0	301	0	0	0	0	0	0
26	0	101	0	201	0	301	0	0	0	0	0	0
27	0	102	0	201	0	0	0	0	0	0	0	0
28	0	102	0	201	0	0	0	0	0	0	0	0
29	0	102	0	201	0	0	0	0	0	0	0	0
30	0	102	0	0	0	0	0	0	0	0	0	0
31	0	501	0	0	0	0	0	0	0	0	0	0
32	0	501	0	101	0	0	0	0	0	0	0	0
33	0	501	0	101	0	0	0	0	0	0	0	0
34	0	401	0	101	0	0	0	301	0	0	0	0
35	0	401	0	101	0	0	0	301	0	0	0	0
36	0	302	102	101	0	201	0	301	0	0	0	0
37	0	302	102	101	0	201	0	301	0	0	0	0
38	0	202	0	102	0	201	0	301	0	0	0	0
39	0	202	0	102	0	201	0	301	0	0	0	0
40	0	202	0	102	0	201	0	301	0	0	0	0
41	0	202	401	102	0	0	0	301	0	0	0	0
42	0	0	401	102	0	0	0	301	0	0	0	0
43	0	0	401	102	0	0	0	301	0	0	0	0
44	0	0	0	401	0	501	0	301	0	0	0	0
45	0	0	0	401	0	501	0	301	0	0	0	0
46	0	502	0	401	0	501	0	301	0	0	0	0
47	0	502	202	401	0	501	0	301	0	0	0	0
48	0	502	0	202	302	501	201	301	0	0	0	0
49	0	203	0	202	0	302	0	201	0	0	0	0
50	0	203	0	202	0	302	0	201	0	0	0	0
51	0	203	0	202	0	302	0	201	0	0	0	0
52	0	203	0	202	0	302	0	201	0	0	0	0
53	0	0	0	202	0	302	0	201	0	0	0	0
54	0	0	0	202	401	302	0	201	0	0	0	0
55	0	0	0	202	401	302	0	201	0	101	0	0
56	0	0	0	202	401	302	501	201	0	101	0	0
57	0	0	0	202	401	302	501	201	0	101	0	0
58	0	0	203	202	401	302	501	201	0	101	0	0
59	0	0	203	202	502	401	0	501	0	101	0	301
60	0	0	203	202	502	401	0	501	0	101	0	301
61	0	0	203	202	502	401	0	501	102	101	0	0
62	0	503	203	202	502	401	0	501	102	101	0	0
63	0	503	203	202	502	401	0	501	102	101	0	0
64	0	503	203	202	502	401	0	501	102	101	0	0
65	0	204	203	202	502	401	0	501	0	102	0	0
66	0	204	203	202	502	401	302	501	0	102	0	0
67	0	204	203	202	502	401	0	302	0	102	0	0
68	0	204	0	203	502	401	0	302	0	102	0	0
69	0	0	0	203	0	502	0	302	0	102	0	201
70	0	0	0	203	0	502	0	302	0	102	101	201
71	0	0	0	203	0	502	0	302	0	102	0	101
72	0	0	0	203	0	502	0	302	0	102	0	101
73	0	0	0	203	0	502	0	302	501	102	0	0
74	0	0	204	203	0	202	0	302	501	102	0	0
75	0	0	204	203	503	202	0	302	0	501	0	0
76	0	0	204	203	503	202	0	302	0	501	0	0
77	0	0	204	203	503	202	0	302	0	501	0	0
78	0	0	204	203	503	202	0	302	0	501	0	0
79	0	0	204	203	0	503	0	302	0	501	0	0
80	0	0	204	203	0	503	0	302	0	501	0	102
81	0	0	204	203	0	503	502	302	401	501	0	102
82	0	504	204	203	0	503	0	502	401	501	0	0
83	0	504	204	203	0	503	0	502	401	501	0	0
84	0	504	204	203	0	0	0	502	401	501	0	0
85	0	0	204	203	0	0	0	502	0	401	0	0

```
 86      0    0    204 203      0    0    202 502      0 401      0    0
 87      0    0    204 203      0    0    202 502      0 401      0    0
 88      0    0      0 204      0    0    202 502      0 401      0    0
 89      0    0      0 204      0    0    202 502      0   0      0    0
 90      0    0      0 204      0    0      0 202      0   0      0  501
 91      0    0      0 204      0    0    503 202      0   0      0    0
 92      0    0      0 204      0    0    503 202      0   0      0  302
 93      0    0      0 204      0    0    503 202      0   0      0  302
 94      0    0      0 204      0  203    503 202      0   0      0  401
 95      0    0      0 204    504  203    503 202      0   0      0    0
 96      0    0      0 204    504  203    503 202      0 502      0    0
 97      0    0      0 204    504  203    503 202      0 502      0    0
 98      0    0      0 204    504  203    503 202      0 502      0    0
 99      0    0      0 204      0  504    503 202      0 502      0    0
100      0    0      0 204      0  504      0 503      0 502      0    0
101      0  103      0 204      0  504      0 503      0 502      0    0
102      0  103      0 204      0  504      0 503      0 502      0    0
103      0  103      0 204      0  504      0 503      0 502      0    0
104      0  103      0 204      0    0      0 503      0 502      0    0
105      0  505      0 204      0    0      0 503      0 502      0    0
106      0  505      0 204      0    0    203 503      0   0      0    0
107      0  505      0 204      0    0    203 503      0   0      0    0
108      0  402      0   0      0    0      0 203      0   0      0    0
109      0  402      0   0      0    0      0 203      0   0      0    0
110      0  303      0 103      0    0      0 203      0   0      0  202
111      0  303      0 103      0    0    504 203      0   0    502  202
112      0  205      0 103      0    0    504 203      0   0      0  502
113      0  205      0 103      0    0    504 203      0   0      0    0
114      0  205      0 103      0  204    504 203      0 503      0    0
115      0  205    402 103      0  204    504 203      0 503      0    0
116      0  104      0 402      0  204    504 203      0 503      0    0
117      0  104      0 402      0  204    504 203      0 503      0    0
118      0  104      0 402    505  204      0 504      0 503      0    0
119      0  104      0 402      0  505      0 504      0 503      0    0
120      0    0      0   0      0  505      0 504      0 503      0    0
121      0    0      0 205      0  505      0 504      0 503      0    0
122      0    0      0 205    303  505      0 504      0 503      0    0
123      0    0      0 205    303  505      0 504      0 503      0    0
124      0    0      0 205      0  303      0 504      0   0      0    0
125      0    0    104 205      0  303      0 504      0   0      0    0
126      0    0    104 205    402  303      0 204      0   0      0    0
127      0    0    104 205    402  303      0 204      0   0      0    0
128      0    0    104 205    402  303      0 204      0   0      0  203
129      0    0    104 205    402  303      0 204      0   0    503  203
130      0    0    104 205    402  303      0 204      0   0      0  503
131      0    0    104 205    402  303    505 204      0   0      0    0
132      0  105    104 205    402  303    505 204      0 504      0    0
133      0  105    104 205    402  303    505 204    103 504      0    0
134      0  105    104 205      0  402    505 204    103 504      0    0
135      0  105    104 205      0  402    505 204    103 504      0    0
136      0  506    104 205      0  402      0 505    103 504      0    0
137      0  506    104 205      0  402      0 505    103 504      0    0
138      0  506    104 205      0  402      0 505    103 504      0    0
139      0    0    104 205      0  402      0 505    103 504      0    0
140      0    0    104 205      0  402      0 505    103 504      0    0
141      0    0    105 104      0  402    303 505    103 504      0    0
142      0    0    105 104      0  402    303 505      0 103      0    0
143      0    0    105 104      0  402    303 505      0 103      0    0
144      0    0    105 104      0    0      0 303      0 103      0    0
145      0    0    105 104      0    0      0 303      0 103      0    0
146      0    0    105 104      0    0      0 303      0 103      0  204
147      0    0      0 105      0  205      0 303      0 103    504  204
148      0    0      0 105      0  205      0 303      0 103      0  504
149      0    0      0 105    506  205      0 303      0 103      0    0
150      0    0      0 105    506  205      0 303    505 103      0    0
151      0    0      0 105    506  205      0 303    505 103      0    0
```

152	0	0	0	105	0	506	0	303	0	505	0	0
153	0	0	0	0	0	506	0	303	0	505	0	0
154	0	0	0	0	0	506	0	303	0	505	0	0
155	0	0	0	0	0	506	0	303	0	505	0	0
156	0	0	0	0	0	506	0	303	402	505	0	0
157	0	0	0	0	0	0	0	303	402	505	0	103
158	0	0	0	0	0	0	0	303	402	505	0	103
159	0	0	0	0	0	0	0	205	402	505	0	0
160	0	0	0	0	0	0	0	205	402	505	0	0
161	0	0	0	0	0	0	0	205	402	505	0	0
162	0	0	0	0	0	0	0	205	0	402	0	0
163	0	0	0	0	0	0	0	205	0	402	0	0
164	0	0	0	0	0	0	506	205	104	402	0	0
165	0	0	0	0	0	0	506	205	104	402	0	0
166	0	0	0	0	0	0	506	205	0	104	0	0
167	0	0	0	0	0	0	506	205	0	104	0	505
168	0	0	0	0	0	0	506	205	0	104	0	0
169	0	0	0	0	0	0	0	506	0	104	0	303
170	0	0	0	0	0	0	0	506	105	104	0	303
171	0	0	0	0	0	0	0	506	105	104	0	402
172	0	0	0	0	0	0	0	506	105	104	0	0
173	0	0	0	0	0	0	0	506	105	104	0	0
174	0	0	0	0	0	0	0	506	105	104	0	0
175	0	0	0	0	0	0	0	506	105	104	0	0
176	0	0	0	0	0	0	0	506	0	105	0	0
177	0	0	0	0	0	0	0	0	0	105	0	0
178	0	0	0	0	0	0	0	0	0	105	0	0
179	0	0	0	0	0	0	0	0	0	105	0	205
180	0	0	0	0	0	0	0	0	0	105	0	205
181	0	0	0	0	0	0	0	0	0	105	0	104
182	0	0	0	0	0	0	0	0	0	105	0	104
183	0	0	0	0	0	0	0	0	506	105	0	0
184	0	0	0	0	0	0	0	0	506	105	0	0
185	0	0	0	0	0	0	0	0	506	105	0	0
186	0	0	0	0	0	0	0	0	0	506	0	0
187	0	0	0	0	0	0	0	0	0	506	0	0
188	0	0	0	0	0	0	0	0	0	506	0	0
189	0	0	0	0	0	0	0	0	0	506	0	0
190	0	0	0	0	0	0	0	0	0	506	0	0
191	0	0	0	0	0	0	0	0	0	506	0	105
192	0	0	0	0	0	0	0	0	0	506	0	105
193	0	0	0	0	0	0	0	0	0	506	0	0
194	0	0	0	0	0	0	0	0	0	506	0	0
195	0	0	0	0	0	0	0	0	0	506	0	0
196	0	0	0	0	0	0	0	0	0	0	0	0
197	0	0	0	0	0	0	0	0	0	0	0	0
198	0	0	0	0	0	0	0	0	0	0	0	0
199	0	0	0	0	0	0	0	0	0	0	0	0
200	0	0	0	0	0	0	0	0	0	0	0	0
201	0	0	0	0	0	0	0	0	0	0	0	506

```
PROBLEM DATA:  EXAMPLE 5

     NO OF MACHINES: 6
     TRAVEL TIMES BETWEEN MACHINES:                    FROM
                                              TO    1    2    3    4    5

                                               2    5
                                               3   10    6
                                               4   16   12    7
                                               5   21   17   12    6
                                               6   25   21   16   10    5
     NO OF ITEM TYPES: 5

                             ITEM TYPE    NO REQD    PROC. TIMES ON MC
                                                       1    2    3    4    5    6

                                   1          5        4    6    0    0   10    2
                                   2          5        4   20    5   10    0    2
                                   3          3        2    0   10   15    8    2
                                   4          2        2    4   10    0    4    1
                                   5          6        3    0    5    8   10    1

                        PERIOD = 143       SLACK(J)   75    5   38    0    1  109

ITEM TYPE   CRSPR   OTAKE
     1        0      0
     2        2      3 5
     3        2      1 4
     4        3      1 3 5
     5        1      1
REACH(MC,MCN) :
   7   12   25   32   47
  10   38   23   38
  12   22   31
  14   20
   9

 NO SATURATED SCHEDULE EXISTS
1667 NODES WERE GENERATED

 LONGEST SATURATED PARTIAL SCHEDULE HAS   18 ITEMS:
  2   1   1   5   5   3   4   2   5   2   1   1   3   1   3   2   5   4

                 TIME        MC1         MC2         MC3         MC4         MC5         MC6


                   1        0  201      0     0     0     0     0     0     0     0     0     0
                   2        0  201      0     0     0     0     0     0     0     0     0     0
                   3        0  201      0     0     0     0     0     0     0     0     0     0
                   4        0  201      0     0     0     0     0     0     0     0     0     0
                   5        0    0      0     0     0     0     0     0     0     0     0     0
                   6        0    0      0     0     0     0     0     0     0     0     0     0
                   7        0    0      0     0     0     0     0     0     0     0     0     0
                   8        0    0      0     0     0     0     0     0     0     0     0     0
                   9        0    0      0     0     0     0     0     0     0     0     0     0
                  10        0    0      0   201     0     0     0     0     0     0     0     0
                  11        0    0      0   201     0     0     0     0     0     0     0     0
                  12        0    0      0   201     0     0     0     0     0     0     0     0
                  13        0    0      0   201     0     0     0     0     0     0     0     0
                  14        0    0      0   201     0     0     0     0     0     0     0     0
                  15        0    0      0   201     0     0     0     0     0     0     0     0
                  16        0    0      0   201     0     0     0     0     0     0     0     0
                  17        0    0      0   201     0     0     0     0     0     0     0     0
```

18	0	0	0	201	0	0	0	0	0	0	0	0
19	0	0	0	201	0	0	0	0	0	0	0	0
20	0	0	0	201	0	0	0	0	0	0	0	0
21	0	0	0	201	0	0	0	0	0	0	0	0
22	0	101	0	201	0	0	0	0	0	0	0	0
23	0	101	0	201	0	0	0	0	0	0	0	0
24	0	101	0	201	0	0	0	0	0	0	0	0
25	0	101	0	201	0	0	0	0	0	0	0	0
26	0	102	0	201	0	0	0	0	0	0	0	0
27	0	102	0	201	0	0	0	0	0	0	0	0
28	0	102	0	201	0	0	0	0	0	0	0	0
29	0	102	0	201	0	0	0	0	0	0	0	0
30	0	501	0	0	0	0	0	0	0	0	0	0
31	0	501	0	101	0	0	0	0	0	0	0	0
32	0	501	0	101	0	0	0	0	0	0	0	0
33	0	502	0	101	0	0	0	0	0	0	0	0
34	0	502	0	101	0	0	0	0	0	0	0	0
35	0	502	102	101	0	0	0	0	0	0	0	0
36	0	301	102	101	0	201	0	0	0	0	0	0
37	0	301	0	102	0	201	0	0	0	0	0	0
38	0	401	0	102	0	201	0	0	0	0	0	0
39	0	401	0	102	0	201	0	0	0	0	0	0
40	0	202	0	102	0	201	0	0	0	0	0	0
41	0	202	0	102	0	0	0	0	0	0	0	0
42	0	202	0	102	0	0	0	0	0	0	0	0
43	0	202	0	0	0	501	0	0	0	0	0	0
44	0	0	0	0	0	501	0	0	0	0	0	0
45	0	0	0	401	0	501	0	0	0	0	0	0
46	0	0	0	401	502	501	0	0	0	0	0	0
47	0	0	0	401	502	501	0	0	0	0	0	0
48	0	0	0	401	301	502	0	201	0	0	0	0
49	0	0	0	202	301	502	0	201	0	0	0	0
50	0	503	0	202	301	502	0	201	0	0	0	0
51	0	503	0	202	301	502	0	201	0	0	0	0
52	0	503	0	202	301	502	0	201	0	0	0	0
53	0	203	0	202	0	301	0	201	0	0	0	0
54	0	203	0	202	0	301	0	201	0	101	0	0
55	0	203	0	202	401	301	501	201	0	101	0	0
56	0	203	0	202	401	301	501	201	0	101	0	0
57	0	0	0	202	401	301	501	201	0	101	0	0
58	0	0	0	202	401	301	0	501	0	101	0	0
59	0	0	0	202	401	301	0	501	0	101	0	0
60	0	103	0	202	401	301	502	501	102	101	0	0
61	0	103	0	202	401	301	502	501	102	101	0	0
62	0	103	203	202	401	301	502	501	102	101	0	0
63	0	103	203	202	503	401	502	501	102	101	0	0
64	0	0	203	202	503	401	502	501	0	102	0	0
65	0	0	203	202	503	401	502	501	0	102	0	0
66	0	0	203	202	503	401	0	502	0	102	0	0
67	0	0	203	202	503	401	0	502	0	102	0	0
68	0	0	203	202	503	401	0	502	0	102	0	201
69	0	0	103	203	503	401	0	502	0	102	101	201
70	0	0	103	203	503	401	301	502	0	102	0	101
71	0	0	103	203	503	401	301	502	0	102	0	101
72	0	0	103	203	503	401	301	502	501	102	0	0
73	0	0	103	203	0	503	301	502	501	102	0	0
74	0	0	103	203	0	503	0	301	0	501	0	0
75	0	0	103	203	202	503	0	301	0	501	0	0
76	0	0	103	203	202	503	0	301	0	501	0	0
77	0	0	103	203	202	503	0	301	0	501	0	0
78	0	0	103	203	0	202	0	301	0	501	0	0
79	0	0	103	203	0	202	0	301	0	501	0	102
80	0	104	103	203	0	202	0	301	502	501	0	102
81	0	104	103	203	0	202	0	301	502	501	0	0
82	0	104	103	203	0	202	0	301	502	501	0	0
83	0	104	103	203	0	0	0	301	502	501	0	0

```
 84     0 302   103 203     0   0     0 301     0 502     0   0
 85     0 302   103 203     0   0   503 301   401 502     0   0
 86     0   0   103 203     0   0   503 301   401 502     0   0
 87     0   0   103 203     0   0   503 301   401 502     0   0
 88     0   0   103 203     0   0   503 301   401 502     0   0
 89     0   0   104 103     0   0     0 503   401 502     0 501
 90     0   0   104 103     0   0   202 503   401 502     0   0
 91     0   0   104 103     0   0   202 503   401 502     0   0
 92     0   0   104 103     0   0   202 503   401 502     0   0
 93     0   0   104 103     0   0   202 503   401 502     0   0
 94     0 105   104 103     0   0   202 503     0 401     0   0
 95     0 105     0 104     0 203   202 503   301 401     0   0
 96     0 105     0 104   302 203   202 503   301 401     0   0
 97     0 105     0 104   302 203     0 202   301 401     0   0
 98     0 303     0 104   302 203     0 202     0 301     0   0
 99     0 303     0 104   302 203     0 202     0 301     0 502
100     0 204     0 104     0 302     0 202     0 301     0   0
101     0 204     0   0     0 302     0 202     0 301     0   0
102     0 204     0   0     0 302     0 202     0 301     0   0
103     0 204     0 105     0 302     0 202   503 301     0 401
104     0 504     0 105     0 302     0 202   503 301     0   0
105     0 504     0 105     0 302     0 202   503 301     0   0
106     0 504     0 105     0 302     0 202     0 503     0   0
107     0 402     0 105     0 302     0 203     0 503     0   0
108     0 402     0 105     0 302     0 203     0 503     0   0
109     0   0     0 204     0 302     0 203     0 503     0   0
110     0   0     0 204     0 303     0 203     0 503     0   0
111     0   0     0 204     0 303     0 203     0 503     0 301
112     0   0     0 204     0 303     0 203   103 503     0 301
113     0   0     0 204     0 303     0 203   103 503     0   0
114     0   0   402 204     0 303     0 203   103 503     0   0
115     0   0   402 204     0 303     0 203   103 503     0   0
116     0   0   402 204     0 303     0 203     0 103     0   0
117     0   0   402 204   504 303     0 302     0 103     0 202
118     0   0   402 204   504 303     0 302   104 103     0 202
119     0   0   402 204   504 303     0 302   104 103     0   0
120     0   0   402 204     0 504     0 302   104 103     0   0
121     0   0   402 204     0 504     0 302   104 103     0 503
122     0   0   402 204     0 504     0 302   104 103     0   0
123     0   0   402 204     0 504     0 302   104 103     0   0
124     0   0   402 204     0 504     0 302   104 103     0   0
125     0   0   402 204     0   0     0 302   104 103     0   0
126     0   0   402 204     0   0     0 302   105 104     0   0
127     0   0   402 204     0   0   303 302   105 104     0 203
128     0   0   402 204     0   0   303 302   105 104     0 203
129     0   0     0 402     0   0   303 302   105 104     0   0
130     0   0     0 402     0   0   303 302   105 104     0   0
131     0   0     0 402     0   0   303 302   105 104     0 103
132     0   0     0 402     0   0   504 303   105 104     0 103
133     0   0     0   0     0   0   504 303   105 104     0   0
134     0   0     0   0     0   0   504 303   105 104     0   0
135     0   0     0   0     0 204   504 303   105 104     0   0
136     0   0     0   0     0 204   504 303     0 105     0   0
137     0   0     0   0     0 204   504 303     0 105     0   0
138     0   0     0   0     0 204   504 303   302 105     0   0
139     0   0     0   0   402 204   504 303   302 105     0   0
140     0   0     0   0     0 402   504 303   302 105     0   0
141     0   0     0   0     0 402   504 303   302 105     0 104
142     0   0     0   0     0 402   504 303   302 105     0 104
143     0   0     0   0     0 402   504 303   302 105     0   0
144     0   0     0   0     0 402   504 303   302 105     0   0
145     0   0     0   0     0 402   504 303   302 105     0   0
146     0   0     0   0     0 402   504 303     0 302     0   0
147     0   0     0   0     0 402   204 504     0 302     0   0
148     0   0     0   0     0 402   204 504     0 302     0   0
149     0   0     0   0     0 402   204 504     0 302     0   0
```

150	0	0	0	0	0	0	204	504	0	302	0	0
151	0	0	0	0	0	0	204	504	0	302	0	105
152	0	0	0	0	0	0	204	504	0	302	0	105
153	0	0	0	0	0	0	204	504	303	302	0	0
154	0	0	0	0	0	0	204	504	0	303	0	0
155	0	0	0	0	0	0	0	204	0	303	0	0
156	0	0	0	0	0	0	0	204	0	303	0	0
157	0	0	0	0	0	0	0	204	0	303	0	0
158	0	0	0	0	0	0	0	204	0	303	0	0
159	0	0	0	0	0	0	0	204	0	303	0	302
160	0	0	0	0	0	0	0	204	0	303	0	302
161	0	0	0	0	0	0	0	204	504	303	0	0
162	0	0	0	0	0	0	0	204	402	504	0	0
163	0	0	0	0	0	0	0	204	402	504	0	0
164	0	0	0	0	0	0	0	204	402	504	0	0
165	0	0	0	0	0	0	0	0	402	504	0	0
166	0	0	0	0	0	0	0	0	402	504	0	0
167	0	0	0	0	0	0	0	0	402	504	0	303
168	0	0	0	0	0	0	0	0	402	504	0	303
169	0	0	0	0	0	0	0	0	402	504	0	0
170	0	0	0	0	0	0	0	0	402	504	0	0
171	0	0	0	0	0	0	0	0	402	504	0	0
172	0	0	0	0	0	0	0	0	0	402	0	0
173	0	0	0	0	0	0	0	0	0	402	0	0
174	0	0	0	0	0	0	0	0	0	402	0	0
175	0	0	0	0	0	0	0	0	0	402	0	204
176	0	0	0	0	0	0	0	0	0	0	0	204
177	0	0	0	0	0	0	0	0	0	0	0	504
178	0	0	0	0	0	0	0	0	0	0	0	0
179	0	0	0	0	0	0	0	0	0	0	0	0
180	0	0	0	0	0	0	0	0	0	0	0	0
181	0	0	0	0	0	0	0	0	0	0	0	402

REFERENCES

1. M. Athans, N.H. Cook, S.B. Gershwin, et al., "Complex Materials Handling and Assembly Systems"
Interim Reports: ESL-IR-740, ESL-IR-771 (1977)
Final Report: ESL-FR-834-1 (1979)
Laboratory for Information and Decision Systems, M.I.T.

2. J. Kimemia, S.B. Gershwin, "Multicommodity Networks Flows Optimization in Flexible Manufacturing Systems", Report No. ESL-FR-834-2, Laboratory for Information and Decision Systems, M.I.T. (1978)

3. E.G. Coffman, Jr., (Ed.), "Computer and Job/Shop Scheduling Theory" J. Wiley & Sons, N.Y., 1976.

4. B.J. Lageweb, J.K. Lenstra, A.H.G. Rinnoy Kan, "A General Bounding Scheme for the Permutation Flow Shop Problem", Operations Research, Vol. 26, nc. 1, Jan.-Feb. 1978, pp. 53-67.

5. ——— , "Job-Shop Scheduling by Implicit Enumeration", Management Science, Vol. 24, No. 4, Dec. 1977, pp. 441-450.

6. T. Gonzalez, S. Sahni, "Flowshop and Jobshop Schedules: Complexity and Approximation", Operations Research, Vol. 26, no. 1, Jan.-Feb. 1978, pp. 36-52.

7. P.C. Kanellakis, "Algorithms for a Scheduling Application of the Asymmetric Traveling Salesman Problem", Report No. ESL-FR-834-5, Laboratory for Information and Decision Systems, M.I.T. (1978)

8. R. Hildebrand (C.S. Draper Laboratory, Cambridge, Mass.) Private Communications.

9. J.A. Buzacott, L.E. Hanifin, "Models of Automatic Transfer Lines with Inventory Banks - A Review and Comparison", AIIE Trans., Vol. 10, no. 2, June 1978, p. 197-207.

10. I.C. Schick, S.B. Gershwin, "Modelling and Analysis of Unreliable Transfer Lines with Finite Interstage Buffers", Report No. ESL-FR-834-6, Laboratory for Information and Decision Systems, M.I.T. (1978)