This paper discusses the periodic job shop scheduling problem, a problem where an identical mixture of items, called a minimal part set (MPS), is repetitively produced. The performance and behavior of schedules are discussed. Two basic performance measures, cycle time and makespan, are shown to be closely related. The minimum cycle time is identified as a circuit measure in a directed graph. We establish that there exists a class of schedules that minimizes cycle time and repeats an identical timing pattern every MPS. An algorithm is developed to construct such schedules. We show that minimizing the makespan as a secondary criterion, minimizes several other performance measures.

For makespan minimization, we examine earliest starting schedules where each operation starts as soon as possible. We characterize the cases where after a finite number of MPSs, the earliest starting schedule repeats an identical timing pattern every fixed number of MPSs. We also develop a modification to an earliest starting schedule that repeats an identical timing pattern every MPS when the beginning operations on the machines are delayed.

We consider a general job shop that processes large quantities of a number of different items. As opposed to producing a batch of each item, a small set of items is repetitively produced, and each set has the same proportion of items as the production requirement. Hitz (1980) defines the smallest such set as a minimal part set (MPS). When identical MPSs are produced repetitively, and the processing order at each machine is the same for each MPS, the scheduling problem is called periodic scheduling (Hitz 1980). As an example, consider the production of 100 units of part A, 500 units of part B, and 300 units of part C. The minimal part set is (1A, 5B, 3C). The minimal part set would be produced 100 times to complete the production requirement.

When periodic scheduling is used in multiechelon production systems, complete part sets are supplied for downstream assembly in a more continuous fashion than in batch production. The result is smoother production of finished goods. In situations where there is a constant proportion between the demand rates of the items, periodic scheduling achieves timely delivery and reduced inventory.

Furthermore, periodic scheduling is used to simplify the scheduling process. When there are large batches of items to produce, the number of operations to be scheduled is large and scheduling and flow control may be complicated. Since periodic scheduling repeats an identical mixture of items, the behavior is more predictable, and scheduling and flow control are simplified. For instance, periodic scheduling can be applied to flexible manufacturing systems to simplify scheduling once the assignment of operations to the machines is determined (see Afentakis 1986, Cohen et al. 1985, Hitz, and Wittrock 1985).

A variety of objectives can be considered in periodic scheduling. A commonly considered objective is to maximize the throughput rate, or to minimize its reciprocal, the cycle time. This measure is closely related to machine utilization. In classical scheduling, an important objective is to minimize the makespan. For our problem, there are two kinds of makespans. The total makespan, a measure used in classical scheduling, is the time that it takes to complete all jobs. The MPS makespan is the maximum time it takes to complete an MPS. By minimizing the makespan, we keep an MPS in process for the minimum possible time. With this objective, the average number of MPSs simultaneously in process tends to be small. If items are

Subject classifications: Sequencing; deterministic; periodic scheduling, job shops, performance criteria, constraints. Networks/graphs: applications.

Area of review: MANUFACTURING, OPERATIONS, AND SCHEDULING.
shipped out of the plant as MPS bundles (as might be the case when we are producing a variety of items in appropriate proportions for a downstream product), then reducing the MPS makespan also reduces the work-in-process (WIP) inventory.

For most conventional scheduling problems, the operations start as soon as possible. Determining the starting times is not an important issue. This is not true for periodic scheduling. After the operations are sequenced on the machines (the processing order), the starting times of the jobs must be specified. Since the production process repeats the same operations many times, it is desirable to have an uncomplicated, predictable production pattern. A schedule that repeats an identical timing pattern every MPS is called a stable schedule. When the timing of each operation can be controlled as in a computer-integrated shop, a stable schedule has many advantages. A stable schedule leads to steady production, where WIP inventory, material supply, and part flow are relatively constant. Further, for implementation, a stable schedule requires storage of the timing decision of only a single MPS.

Another important property of a schedule relates to the timing control of the operations. If each operation starts as soon as all its preceding operations have completed, we call the schedule an earliest starting schedule. Although it is not known whether the minimum total makespan can be computed in polynomial time, an earliest starting schedule has several desirable characteristics. The schedule minimizes the total makespan. Also, since the operations on each machine are started as soon as possible (in the given machine processing order), it is not necessary to determine the starting times of the operations prior to actual production or to deliberately control the timing of operations. Given a processing order, an earliest starting schedule is unique because the earliest starting time of each operation is unique.

There are numerous studies on periodic scheduling. Many of these studies assume specific shop structures. These include flow shops (Hitzi), flow shops with no buffer (Matsuo 1988), flow lines (Wittrock), and simple assembly lines with finite buffers (Pinedo et al. 1986 and McCormick et al. 1989). Cleaver and Jackson (1987) and Roundy (1992) discuss a special periodic job shop with identical jobs. For a general job shop, Afentakis minimizes the MPS makespan in a manner similar to a conventional job shop scheduling problem. For a periodic job shop where each operation starts as soon as possible, Cohen et al. discuss a method to determine the cycle time.

Periodic scheduling has been also studied in contexts other than shop scheduling. Carlier and Chretienne (1990) and Munier (1990) consider task scheduling in a pipelined computer architecture for parallel processing. The scheduling problem has a set of infinitely repeating tasks that satisfy precedence and resource constraints. The periodic job shop problem is a special case of this resource constrained periodic scheduling problem. Hanen (1990) discusses minimization of cycle time among stable schedules for resource constrained periodic scheduling. Periodic scheduling has been also studied as event scheduling in a timed Petri net. Such works include those of Ramchandani (1973), Sifakis (1980), Chretienne (1983) and (1988), Hanen (1987 and 1989), and Hillion and Proth (1989). Serafini and Ukovich (1989) discuss a model of periodic scheduling where the goal is to schedule a set of activities that occur periodically so that they satisfy particular time and resource constraints. Erschler et al. (1985) determine the characteristics of part release strategies given resource constraints and specified release intervals.

While various authors have used different performance measures, there has been no attempt to determine which of these measures might be most appropriate for a given application. Further, the relationships between the criteria have not been discussed. As a result, the relative cost of choosing a particular measure is unknown. As an example, it would be useful to know what effect minimizing the cycle time has on WIP inventory. There is also limited information about the compatibility of performance criteria and timing pattern constraints. For instance, it is not known whether there exists a schedule that minimizes cycle time and is also stable. The goal of this research is to provide answers to some of these issues. These issues require resolution because heuristic or optimal methods cannot be used until an appropriate model is formulated. Using the results of our research, we can provide the shop floor supervisor with procedures to improve his current schedules.

Unlike conventional scheduling, periodic scheduling involves the timing problem of constructing desirable schedules for a given machine processing order as well as the sequencing problem of determining optimal processing order of operations at each machine. In this paper, we discuss the timing problem when the machine processing order is appropriately determined. We demonstrate how the timing theory is essential for the sequencing problem. The sequencing problem, as a study subsequent to this paper, is partially investigated in Lee (1991).

We address the problems of finding procedures to make schedules stable and of determining the costs associated with using these schedules. Also, when earliest starting schedules are used, we can recommend against using certain machine processing orders because they lead to unstable schedules and large WIP. We are able to determine which of several MPSs or which of several processing orders provide the schedule with the “best” properties.

We first examine the class of schedules that minimize cycle time. A linear program that finds the minimum cycle time is presented. Then, we establish that there exists a set of stable schedules that minimizes the cycle time. A procedure is presented to find a new schedule that has the same cycle time, is stable, and minimizes a variety of other objectives such as WIP.

Next, the relationship between cycle time and total makespan is examined. We show that as the number of MPSs goes to infinity, (cycle time) × (number of MPSs)
becomes a good approximation for the minimum total makespan.

The behavior of earliest starting schedules is characterized. While we establish that the earliest starting schedule has minimum cycle time, the WIP may grow without bound as the number of MPSs increase. We develop conditions under which the earliest starting schedule becomes periodic after a finite number of MPSs. This implies that the increase in WIP is bounded. Then, we show how to delay a minimal number of operations to make the schedule stable without increasing cycle time.

Effects of the sequencing decision on the cycle time are discussed next. We conclude with some comments about other objectives and restrictions.

1. NOTATION AND ASSUMPTIONS

The items to be processed can undergo one or multiple operations. We let \( N \) denote the set of all operations of an MPS. Operation \( i \in N \) has processing time \( p_i \). Each operation is assigned to a machine. We denote the set of machines by \( M \). There may be precedence relations between the various operations of a given item. For example, in circuit board manufacturing, the chips must be inserted onto the board before they can be soldered.

Operation \( i \in N \) in the \( r \)th MPS is said to be the \( r \)th repetition of operation \( i \) and is denoted by \( i^r \). The starting time of \( i^r \) is \( \chi_i^r \). A schedule is periodic at \( r_0 \) if \( \chi_i^{r_d} = \chi_i^{r_0} \) for some integer \( d \) and real constant \( \mu \), and for all \( i \in N \) and all integers \( r \geq r_0 \). This definition implies that the timing pattern repeats every \( d \) MPSs from the \( r_0 \)th MPS. In particular, when \( d = 1 \), such a schedule is said to be stable at \( r_0 \). The modifying phrase “at \( r_0 \)” is dropped when \( r_0 = 1 \).

There are several important decisions that must be made in a periodic scheduling problem. First, the makeup and number of MPSs have to be determined. Then, the order in which the operations are processed on each machine must be specified. Finally, the starting time of each job must be determined. None of these decisions have been adequately addressed in the literature. Since it is impossible to address all of these issues in one paper, we concentrate on the last issue, the problem of determining the starting times of the jobs.

Consider the following problem.

Example 1.

There are seven operations, \( e_1, e_2, \ldots, e_7 \), with processing times 2, 2, 3, 1, 3, 4, 2, respectively. There are three machines, \( M_1, M_2, \) and \( M_3 \). Operations \( e_1 \) and \( e_5 \) are performed on \( M_1 \); operations \( e_2 \) and \( e_6 \) are performed on \( M_2 \); and operations \( e_3, e_4, \) and \( e_7 \) are performed on \( M_3 \). The sequencing decision is to process \( e_1 \) then \( e_3 \) then \( e_5 \) on \( M_1 \); \( e_2 \) then \( e_6 \) then \( e_4 \) on \( M_2 \); and \( e_7 \) on \( M_3 \). The reader might imagine that there are three items; item \( f \) requires operations \( e_1 \) and \( e_7 \), item \( g \) requires operations \( e_2 \) and \( e_3 \), and item \( h \) requires operations \( e_4 \) and \( e_5 \). The requirements are: \( e_1 \) precedes \( e_7 \), \( e_2 \) precedes \( e_4 \), \( e_3 \) precedes \( e_6 \), and \( e_6 \) precedes \( e_5 \). These requirements are due to technological or physical requirements, and they represent the routing of the items through the job shop.

Without a proper understanding of timing decisions, we might construct a schedule with bad performance properties. For instance, if a stable schedule is required, we might start each operation of the first MPS soon as possible and repeat the MPS schedule for every MPS. Then as shown in Figure 1(a), a new MPS completes every 10 time units. We might decrease the cycle time by removing idle time between the operations of the first MPS and the second MPS on machines 2 and 3. While this violates the stability requirement, all operations complete as soon as possible, and we have the earliest starting schedule shown in Figure 1(b). This schedule is periodic with period 2 at the second MPS. The cycle time (as defined in the next section) is 10 for the first MPS, 8 for the first two MPSs, 8 for the first three MPSs and 7.5 for the first four MPSs. As the number of MPSs goes to infinity, the cycle time decreases to 7. The best stable sequence is shown in Figure 1(c). With this schedule, an MPS completes every seven units of time.

We assume that the number of MPSs and the operations which comprise an MPS are specified as input. Sometimes this information is generated based on the judgment of the shop floor dispatcher. For instance, suppose there are requirements to produce 125 units of part \( A \) and 301 units of part \( B \). Since 125 and 301 are relatively prime, if all the parts must be produced in the same run, then the only MPS that can be chosen is \((125A,301B)\). However, if one unit of part \( B \) can be delayed, then an MPS of \((5A,12B)\) can be selected. This MPS would be processed 25 times, and the last unit of \( B \) could be produced separately. If 25 units of \( A \) and one unit of \( B \) can be delayed, then 100 MPSs of \((1A,3B)\) can be produced. The issue of selecting the makeup of an MPS has not been addressed in the literature. A solution could depend on the optimization criteria and the timing constraints that are specified. Fortunately, the problem of selecting an appropriate MPS is frequently dictated by the production process. In assembly operations, the quantities of parts that are needed are determined by final product requirements. For example, a computer may require three boards of type \( A \), two of type \( B \), and five of type \( C \). If we need to produce 300 computers, then we would construct an MPS of \((3A,2B,5C)\) and process 300 MPSs.

While one objective of periodic scheduling is to produce efficiently, another objective is to simplify scheduling and flow control. For this reason, only simple scheduling processes are considered in periodic scheduling. Consequently, we assume that there is only one type of MPS to be scheduled. We do not, for instance, consider the possibility of mixing \((1A,3B,2C)\) and \((4A,1B,5C)\) in the same run. Mixing not only increases the complexity of scheduling and flow control, but also creates difficulties in determining which part belong to a particular type of MPS.
We assume that there is no initial or ending WIP. Hillion and Proth show that if we have certain parts already processed and these parts are used in some of the MPSs, then the cycle time can sometimes be reduced. From a practical point of view, this means that there must be an initial schedule to process these parts, the main schedule to process the majority of the MPSs, and then a final schedule to process the compliment of the initial schedule. Also, the problem of determining which parts should be selected for initial processing has not been discussed in the literature. Further, the cost associated with the additional WIP needs to be considered.

We assume that the machine processing order of the operations is the same for every MPS and the machines process the operations in the same sequence. We show how to meet the specified timing constraints for a given machine processing order. Most of our theoretical results, such as the ones relating cycle time and other performance measures, hold for all machine processing orders. As a result, specifying a particular one to be used in the analysis is not a restriction.

Further, there are a variety of circumstances under which the machine processing order might be given as part of the input for a periodic scheduling problem. For instance, it might be the case that the machine processing order may be specified due to operational reasons such as flow control or setup constraints. Another case might be when our objective is to improve an existing schedule without rescheduling the facility. Observe that the existing schedule must specify the sequencing decision. In such cases, we can provide improved timing and release decisions.

We note that while determining the machine processing order is an important component in a periodic scheduling problem, it is a difficult problem. Lee shows that this problem is \( NP \)-hard. In some cases, as we later show, the choice of a machine processing order has no effect on the cycle time.

The issues that we address are the relationships between performance measures and timing pattern constraints. To focus on these issues and make the problem tractable, we assume that there is nonpreemptive processing of operations, buffers with infinite capacity, negligible setup and transportation times, and no disruptive events like machine breakdowns. Some of these assumptions are not essential to our analysis. For instance, finite buffers can be incorporated in a manner similar to McCormick et al.

### 2. MINIMIZATION OF CYCLE TIME

In this section we investigate the objective of minimizing the cycle time. Using a special type of directed graph, we present a polynomial time algorithm which finds the minimum cycle time. We show that within the class of schedules that minimizes cycle time, there always exist stable schedules. A polynomial time procedure is developed to find the stable schedule that also minimizes the total makespan and several other secondary objectives. Finally, when the number of MPSs is large, we establish that the
minimum cycle time closely approximates the minimum average makespan.

We begin by developing a formal definition of cycle time. Some authors have defined the cycle time only for the case where the schedule becomes periodic. We propose a definition of cycle time that can evaluate any feasible schedule. The definition we propose is a generalization of the one for an assembly line found in McCormick et al. Let \( a(m) \) and \( b(m) \) be the first and the last operations, respectively, of an MPS on machine \( m \). For a feasible schedule \( s \), we define the cycle time of \( n \) MPSs to be

\[
\mu_s(n) = \max_{m \in M} \frac{x_{b(m)}^n + p_{b(m)} - x_{a(m)}^n}{n}.
\]

Observe that \( x_{b(m)}^n + p_{b(m)} - x_{a(m)}^n \) is the time it takes to process \( n \) MPSs on machine \( m \). The cycle time can be interpreted as the mean time it takes to produce an MPS. The minimum cycle time of \( n \) MPSs over all feasible schedules is denoted by \( \mu^*(n) \). Notice that for a stable schedule, \( \mu_{s+1} = p = \mu \) for all \( i \in N, r \geq 1 \), and some constant \( \mu \). Consequently, for a stable schedule \( s \), \( \mu_s(n) = \mu \) for all integers \( n \geq 1 \).

To investigate the schedules that minimize cycle time, we construct a directed graph that represents the sequencing requirements of the operations. Suppose a processing order for each machine is specified. Let \( E \) be the set of ordered pairs of operations that correspond to the immediate precedence relations between operations. The set \( E \) consists of two types of precedence relations. The first type, \( T \), is the processing requirements of the operations due to technological or physical restrictions. For instance, deposition precedes etching. These requirements represent the routings of the items through the machines in a job shop environment. The second type is the processing order of the operations specified for each machine. As mentioned in the previous section, we assume that these orders are given. For the specified machine processing order, let \( R \) denote the set of ordered pairs of operations, \( \{(b(m), a(m))|m \in M\} \). Each \( (i, j) \in R \) represents the precedence relation between the last operation \( i \) of an MPS on a given machine and the first operation \( j \) of the next MPS on the same machine. The arcs corresponding to elements of \( R \) are called recycling arcs. We introduce two dummy operations, \( u \) and \( v \), that represent the start and the completion of all the operations, respectively. Let \( N' = N \cup \{u, v\} \) and \( E' = E \cup \{(u, j), (i, v)|\} \in R \). Associated with each arc \( (i, j) \in E' \) are two weights. The first weight, \( p_i \), corresponds to the processing time of operation \( i \). We sometimes refer to this weight as the "length" between \( i \) and \( j \). The processing time of \( (u, j) \) is assumed to be zero for all \( j \). The second weight is \( \tau_{ij} \), where \( \tau_{ij} = 1 \) if \( (i, j) \in R \) and \( 0 \) otherwise.

We call the graph with node set \( N' \), arc set \( E' \cup R \), and weights on the arcs as defined above a precedence constraints graph (PCG). Observe that a PCG is a convenient way to display the precedence relationships between the various operations. If we ignore the recycling arcs, the PCG represents the precedence relations between the jobs in one MPS. While we could make many duplicate copies of the graph (one for each MPS) and construct the appropriate connections, it is easier to represent the duplicate copies with the use of recycling arcs.

As previously mentioned, the processing constraints between operations are determined by technological precedence relations and a sequencing decision. To make a sequencing decision for an MPS, an ordering for the processing on each machine is specified. After the sequencing decision is made, a PCG can be constructed without any ambiguity. We note that the structure of the PCG depends on the machine processing order. For a different processing order, a different PCG results. In this work, we use the PCG to obtain theoretical properties of an optimal solution.

The PCG for Example 1 is presented in Figure 2. The values on the arcs are the first weights (the processing times). The set of recycling arcs are \( R = \{(e_5, e_1), (e_6, e_2), (e_7, e_3)\} \).

For the sake of clarity in Example 1, we have assumed that there are three distinct types of items to be processed, \( f, g, \) and \( h \). When an MPS includes multiple copies of an item, each copy is regarded as a separate item. For instance, if the MPS is \( (f, g, 2h) \), then the scheduling problem has four items.

![Figure 2. The PCG for Example 1.](image-url)
en and ten operations. Operations $e_8$, $e_9$, and $e_{10}$ are duplicates of operations $e_3$, $e_4$, and $e_7$. If the size of the problem is an important issue for implementation, then we could group multiple copies of an item together into one job. While reducing the number of items and the computational work, this might result in an increase in the minimum cycle time.

A PCG is a generalization of a graph given for a simple assembly line by McCormick et al. We note that some studies on resource constrained periodic scheduling use different versions of this graph (see Carlier and Chretienne, Hanen, and Munier).

We say a path is simple if it does not contain any circuits. Unless specified, we do not assume that a path is simple. Observe that a path in a PCG represents a feasible sequence of operations. The length of the path corresponds to the time it would take to perform this sequence if all other operations are ignored. A path in a PCG may include nodes associated with operations in different MPSs. Consequently, we sometimes refer to a specific node in the path by its associated operation in a particular MPS. Thus, given a path $\psi$ in a PCG that starts at node $u$, a node $i$ that is visited by $\psi$ is denoted as $i'$ if the number of recycling arcs traversed prior to $i$ is $r - 1$ ($i$ may have been visited previously).

Consider a directed graph where each arc has two weights. An elementary circuit is a circuit (closed path) that does not contain any smaller circuits. For a given circuit, the circuit ratio is the ratio of the sum of the first arc weights to the sum of the second arc weights. The maximum circuit ratio among all elementary circuits in the graph is called the critical circuit ratio. An elementary circuit with the critical circuit ratio is called a critical circuit. Since every circuit can be decomposed into elementary circuits, no circuit (elementary or otherwise) has a circuit ratio larger than the critical circuit ratio. We denote the critical circuit ratio of the PCG corresponding to the machine processing order by $\lambda$. Note that excluding the recycling arcs, a PCG is acyclic. If this were not true, there would be a sequence of operations with no feasible processing order. Since any circuit in a PCG has one or more recycling arcs, the critical circuit ratio is always finite. A critical circuit for Example 1, $(e_1, e_7, e_3, e_6, e_5, e_1)$, is shown in Figure 3. The length of this circuit is 14. Since the circuit contains two recycling arcs, $\lambda = 14/2 = 7$.

Let $A^*$ be the critical circuit ratio of the PCG for a machine processing order that produces a feasible schedule with minimum cycle time. We now show that the minimum cycle time for $n$ MPSs is also $A^*$.

The minimum cycle time for a single MPS over all feasible schedules with a given machine processing order can be found by solving the linear program:

$$z_P = \min \mu$$

subject to $x_j - x_i \geq p_{ij}, \quad (i, j) \in E'$,

$$\mu + x_j - x_i \geq p_{ij}, \quad (i, j) \in R,$$

where $x_i$ is the decision variable for the starting time of operation $i$. Also, the set of "forward" arcs, $E'$, and the set of recycling arcs, $R$, in the PCG are determined from the machine processing order. The second constraint ensures that $x_{b(m)} + p_{b(m)} - x_{a(m)} \leq \mu$ for all $m \in M$. If we want a schedule that starts at time zero, then the constraint $x_1 = 0$ is added. $P$ can be rewritten as:

$$z_P = \min \mu$$

subject to $\tau_{ij} \mu + x_j - x_i \geq p_{ij}, \quad (i, j) \in E' \cup R$.

To find a solution to $P$, we construct its dual. The dual is:

$$z_{DP} = \max \sum_{(i,j) \in E' \cup R} p_{ij}y_{ij}$$

subject to $\sum_{(j) \in E' \cup R} y_{ij} - \sum_{(i,j) \in E' \cup R} y_{ij} = 0, \quad i \in N'$

$$\sum_{(i,j) \in E' \cup R} \tau_{ij}y_{ij} = 1,$$

$$y_{ij} \geq 0, \quad (i, j) \in E' \cup R.$$

$DP$ can be interpreted as a minimum cost network circulation problem in a PCG with the restriction that the
weighted sum of the flows is 1. A network problem similar to DP has been investigated in the study of dynamic networks by Ford and Fulkerson (1958, 1962, 1970), and Orlin (1983, 1984a, 1984b). Dantzig et al. (1966) discuss the problem of determining the minimum cost-to-time ratio over all circuits in a network with cost and time weights. They construct a linear program that is identical to DP except that the \( \tau_{ij} \) are positive real numbers. Since DP is a special case of the linear program examined by Dantzig et al., we can use their findings.

**Lemma 1.** \( \mu^*(1) = \lambda^* \).

**Proof.** For the machine processing order that produces a feasible schedule which minimizes the cycle time of one MPS, \( \mu^*(1) = z_p \). LP duality establishes that \( z_p = z_{DP} \). From Dantzig et al., \( z_{DP} \) has the same value as the critical circuit ratio in a PCG. Therefore, \( z_{DP} = \lambda^* \), and hence \( \mu^*(1) = \lambda^* \). □

Karp and Orlin (1981) present two efficient algorithms to find the critical circuit ratio in a directed graph where the second weight is 0 or 1. Depending on the algorithm used, once a machine processing order is given, \( \lambda \) can be computed in \( O(|V|^2) \) or \( O(|E \cup R|\log|N|) \) time. We note that dummy nodes \( u \) and \( v \) and dummy arcs \( \{(u, j), (i, v) | (i, j) \in R\} \) can be excluded from the graph because a circuit in a PCG does not include any dummy nodes or dummy arcs.

We now show that the minimum cycle time is independent of \( n \). This result, as well as some of the analysis, is a generalization of the work of McCormick et al.

**Theorem 1.** For all \( n \geq 1 \), \( \mu^*(n) = \lambda^* \).

**Proof.** There exists a schedule \( s \) for a single MPS where \( \mu_s(1) = \mu^*(1) \). By replicating the schedule for \( n \) MPSs, we have a schedule \( \mu_s(n) = \mu^*(n) \). Consequently, \( \mu^*(n) \leq \mu^*(1) \). Since from Lemma 1 \( \mu^*(1) = \lambda^* \), we have \( \mu^*(n) \leq \lambda^* \).

To complete the proof, we show that \( \mu^*(n) \approx \lambda^* \). Let \( \pi \) be a critical circuit in the PCG for a minimum cycle time schedule with \( q \geq 1 \) recycling arcs. As a result, there exists \( (i, j) \in R \) such that \( (i, j) \) is a recycling arc of \( \pi \). Operation \( j \) appears every \( q \) MPSs in a path that starts from \( j \) and follows \( \pi \). Consequently, there exists a path of length \( \lambda q \lambda^* \) between \( j \) and \( j^{q+1} \) for all integers \( l \geq 1 \). Observe that the last arc of the path is \( (i, j^{q+1}) \). This implies that there is a path from \( j^q \) to \( j^{q+1} \) with length \( \lambda q \lambda^* \). Therefore, \( x_{j^q} - x_j \geq \lambda q \lambda^* - p_i \). Since for some \( m' \in M \), \( (i, j) = (b(m'), a(m')) \), \( x_{j^q} + x_{j^q} - x_{a(m')} \geq \lambda q \lambda^* \). Therefore, \( \mu^*(lq) = \min_s \mu_s(lq) \) where the minimum is taken over all feasible schedules. If \( n = lq \), then we are done. Therefore, consider the case where \( n = lq + n_0 \) is integer and \( 1 \leq n_0 < q \). For any feasible schedule of \( n \) MPSs with cycle time \( \mu(n) \), by replicating the schedule \( q \) times, we have a schedule of \( nq \) MPSs that has cycle time \( \mu(nq) = \mu(n) \). This implies that \( \mu^*(n) \geq \mu^*(nq) \). By replacing \( lq \) with \( nq \) in equation (1), we have \( \mu^*(n) \geq \mu^*(nq) \). □

A schedule that has the minimum cycle time is called a \( \lambda^*\)-schedule. We now discuss the existence of stable \( \lambda^*\)-schedules and their properties.

**Corollary 1.** There always exists a stable \( \lambda^*\)-schedule.

**Proof.** Let \( s \) be a schedule that replicates a single MPS \( \lambda^*\)-schedule \( n \) times. Then for \( s \),

\[
 n \mu_s(n) = \max_{m \in M} \left( x_{b(m)} + p_{b(m)} - x_{a(m)} \right) = n \lambda^*,
\]

where \( \mu_s(n) \) is the cycle time of schedule \( s \). Since \( \mu_s(n) = \lambda^* \), by Theorem 1 \( s \) is a stable \( \lambda^*\)-schedule. □

From Corollary 1, to find a stable \( \lambda^*\)-schedule, we only need to determine a \( \lambda^*\)-schedule for a single MPS. This implies that to minimize cycle time, the periodic job shop scheduling problem reduces to a job shop scheduling problem of a single MPS that minimizes over \( s \) the performance measure \( \mu_s(1) \). Note that \( \mu_s(1) \) is the "width" of a single schedule \( s \).

There may be numerous stable \( \lambda^*\)-schedules. This is illustrated by the next example.

**Example 2**

There are two machines and two operations. Operation \( e_1 \) has processing time 2 and is processed on machine \( M_1 \). Operation \( e_2 \) has processing time 1 and is processed on machine \( M_2 \). Examples of two different stable timing patterns are presented in Figure 4. The MPS makespan in the first example in Figure 4 is shorter than that in the second example even though the cycle times are identical.
For a given schedule, let $S$ be the set of stable schedules with the same machine processing order and minimal cycle time. Let $S^*$ be this cycle time. To select from the elements of $S$, a secondary measure is useful. We show that several potential secondary measures result in the same schedule. Given a schedule $s$ of $n$ MPSs, the total makespan is denoted by $C_{\text{max}}(n)$. While $C_{\text{max}}(n)$ does depend on $s$, for notational simplicity, the variable $s$ is not included. Let $C^S(n) = \min_{s \in S} C_{\text{max}}(n)$. Note that $C^S(1)$ corresponds to the minimum MPS makespan over $S$.

**Theorem 2.** A schedule $s \in S$ minimizes the MPS makespan over $S$ if and only if $s$ minimizes the total makespan over $S$. Furthermore, $C^S(n) = n \lambda^S + (C^S(1) - \lambda^S)$, where $n$ is the number of MPSs.

**Proof.** Consider a schedule $s \in S$ that starts at time zero and has MPS makespan $C_{\text{max}}(1)$. Since $s \in S$, there is some machine $m'$ where

$$x_{b(m')} + p_{b(m')} - x_{a(m')} = n(x_{b(m')} + p_{b(m')} - x_{a(m')}) = n \lambda^S.$$

Therefore, the total makespan of $s$ is

$$C_{\text{max}}(n) = \max_{m \in M} \{x_{b(m')} + p_{b(m)}\} = \max_{m \in M} \{x_{b(m')} + p_{b(m)}\} + n \lambda^S - (x_{b(m')} + p_{b(m')} - x_{a(m')}) = n \lambda^S + x_{a(m')}^1 + \max_{m \in M} \{x_{b(m')} + p_{b(m)} - (x_{b(m')} + p_{b(m')})\} = n \lambda^S + x_{a(m')}^1 + \max_{m \in M} \{x_{b(m')} + p_{b(m)} - (x_{b(m')} + p_{b(m')})\} = n \lambda^S - (x_{b(m')} + p_{b(m')} - x_{a(m')}) + \max_{m \in M} \{x_{b(m')} + p_{b(m)}\} = n \lambda^S - \lambda^S + C_{\text{max}}(1).$$

Hence,

$$C^S(n) = \min_{s \in S} C_{\text{max}}(n) = \min_{s \in S} \{n \lambda^S + (C_{\text{max}}(1) - \lambda^S)\} = n \lambda^S + (\min_{s \in S} C_{\text{max}}(1) - \lambda^S) = n \lambda^S + (C^S(1) - \lambda^S).$$

Consequently, $s$ minimizes the MPS makespan over $S$ if and only if $s$ minimizes the total makespan over $S$. □

Since $\lambda^*$-schedules are minimum cycle time schedules, we can use Theorem 2 to develop a bound on the difference between the minimum cycle time and the minimum average makespan. Let $C_{\text{max}}(n)$ denote the minimum total makespan of $n$ MPSs over all feasible schedules and $S^*$ denote the set of stable $\lambda^*$-schedules.

**Corollary 2.**

$$0 \leq C^*_{\text{max}}(n) - \lambda^* \leq \frac{C^S(1) - \lambda^*}{n}.$$

**Proof.** Let $s \in S^*$ be a schedule with MPS makespan $C^S(1)$ and total makespan $C^S(n)$. The existence of such a schedule follows from Theorem 2. Also by Theorem 2,

$$\frac{C^S(1) - \lambda^*}{n} = \frac{C^S(n) - \lambda^*}{n}.$$

Since $C_{\text{max}}(n) \leq C^S(n)$, it follows that

$$\frac{C_{\text{max}}(n)}{n} - \lambda^* \leq \frac{C^S(1) - \lambda^*}{n}.$$

Because the total makespan of any schedule is greater than or equal to $n \lambda^*$, $C_{\text{max}}(n)/n - \lambda^* \geq 0$. □

**Remark 1.** Since $C^S(1) - \lambda^*$ is finite, as $n$ goes to infinity, the difference between $\lambda^*$ and the minimum average makespan goes to zero.

It is desirable to have as few MPSs as possible simultaneously in production. Flow control is simplified especially in the event of a system malfunction. Also, when complete MPSs are packaged for shipping or prepared for downstream assembly, completed items of an MPS are stored until the remaining items of the MPS finish production. In these situations, the WIP inventory is proportional to the average number of MPSs simultaneously in production. Theorem 3 shows that minimizing the total makespan over $S$ also minimizes the average number of MPSs simultaneously in production.

The reader should observe that Theorem 3 and all subsequent discussion in this section also applies to the set of stable $\lambda^*$-schedules.

**Theorem 3.** A schedule $s$ that minimizes the total makespan over $S$ minimizes the average number of MPSs simultaneously in production over $S$. The minimum average number of MPSs simultaneously in production is $L = (1/\lambda^S) C^S(1)$.

**Proof.** Consider a stable schedule $s$ with MPS makespan $C^S(1)$. The time each MPS spends in production is $C^S(1)$. The reciprocal of the cycle time is the arrival rate. Then, the average number of MPSs simultaneously in production is given as $L = (1/\lambda^S) C^S(1)$ by Little’s formula (see Stidham 1974). The proof now follows from Theorem 2. □

We now discuss a method to find a stable schedule in $S$ that minimizes the total makespan as a secondary objective. To determine such a schedule, it suffices to find a schedule in $S$ for a single MPS that minimizes the MPS makespan as the secondary objective. Once such a schedule is found, Theorem 2 establishes that a stable schedule with minimum total makespan can be constructed by replicating the single MPS schedule. To find the single MPS schedule, we replace the weights associated with each arc $(i, j)$ in the PCG by $c_{ij}$ where
This new graph is called PCG'. Figure 5 depicts the PCG' for Example 1. The graph PCG' is a modification of the PCG given in Figure 2.

We now investigate some properties of a PCG'.

**Lemma 2.** Every circuit \( \pi' \) in a PCG' has \( \sum_{(i,j) \in \pi'} c_{ij} \leq 0 \). Further, \( \sum_{(i,j) \in \pi} c_{ij} = 0 \) if and only if the circuit in the PCG that corresponds to \( \pi' \) is critical.

**Proof.** Follows from the definition of \( c_{ij} \). \( \Box \)

The following linear program determines a schedule of a single MPS that minimizes the MPS makespan as a secondary objective.

\[
\begin{align*}
z_{P'} &= \min \quad y_v + \sum_{i \in N} y_i \\
\text{subject to} \quad y_j - y_i &\geq p_i, \quad (i, j) \in E', \\
-x_i + x_j &\leq \lambda^S - p_i, \quad (i, j) \in R, \\
x_i &\geq 0, \quad i \in N'.
\end{align*}
\] (P')

Observe that (2) and (4) determine the MPS makespan. The set of constraints (3) ensures that the schedule has cycle time \( \leq \lambda^S \).

Constraints (2) and (3) specify the relationships between the two nodes of the arcs of a PCG'. We show that \( P' \) can be solved efficiently by a longest path algorithm. Denote the length of the longest path from \( i \) to \( j \) in a PCG' by \( \gamma_{ij} \).

**Theorem 4.** A schedule \( s \in S \) that minimizes the total makespan and MPS makespan as secondary criteria has \( x_i^s = \gamma_{ui} \) for all \( i \in N \). Further, \( C^S(1) = \gamma_{uv} \).

**Proof.** Let \( \gamma_{ui} \) be the length of the longest path from \( u \) to \( i \) in a PCG'. Since there exists a path from \( u \) to \( i \) that has no recycling arcs, \( \gamma_{ui} \geq 0 \). Consider the linear program

\[
\begin{align*}
z_{P'} &= \min \quad y_v + \sum_{i \in N} y_i \\
\text{subject to} \quad y_j - y_i &\geq p_i, \quad (i, j) \in E', \\
-x_i + x_j &\leq \lambda^S - p_i, \quad (i, j) \in R, \\
y_i &\geq 0, \quad i \in N'.
\end{align*}
\] (P')

The optimal solution of the linear program is given by \( y^*_v = \gamma_{uv} \) (see Lawler 1976, p. 78).

Observe that the constraint sets of \( P'' \) and \( P' \) are identical. While \( P'' \) determines the lengths of the longest paths from \( u \) to every other node in a PCG', \( P' \) just finds the length of the longest path from \( u \) to \( v \). As a result, the optimal solution to \( P'' \) is also an optimal solution to \( P' \). Consequently, the lengths of the longest paths in a PCG', \( \{\gamma_{ui}\} \), determine the schedule for the first MPS of \( s \in S \) that minimizes the MPS makespan as a secondary objective. By Theorem 2, the schedule also minimizes the total makespan over \( S \).

Finally, we show that \( \{\gamma_{ui}\} \) are finite. By Lemma 2, a PCG' does not have any circuits of positive length. Therefore, the lengths of the longest paths in a PCG' are finite. \( \Box \)

Although schedules that minimize the cycle time are generally different than those that minimize the MPS makespan, the next result identifies a case when a \( \lambda^* \)-schedule has the minimum MPS makespan over all feasible schedules.

**Corollary 3.** If a longest path from \( u \) to \( v \) in the PCG' corresponding to a \( \lambda^* \)-schedule does not contain any recycling arcs, then \( C^S(1) = C^*_{\max}(1) \).

**Proof.** Suppose there exists a longest path from \( u \) to \( v \) in the PCG' corresponding to a \( \lambda^* \)-schedule that does not contain any recycling arcs. This implies that \( C^*_{\max}(1) = \gamma_{uv} \). By Theorem 4, \( C^S(1) = \gamma_{uv} \). Therefore, \( C^S(1) = C^*_{\max}(1) \). \( \Box \)

For a given schedule, we present a procedure to find a stable schedule with the same machine processing order.
that minimizes the cycle time and also minimizes the total makespan as a secondary objective. The procedure gives the timing decision for the first MPS. Each succeeding MPS has an identical structure.

**Algorithm SS:** Finds a schedule of $S$ for a single MPS that minimizes the total makespan as a secondary objective. The procedure gives the timing decision for the first MPS. Each succeeding MPS has an identical structure.

1. Determine the critical circuit ratio $\lambda$.
2. Construct the PCG' from the PCG.
3. For each node $i$ in the PCG', find the length of the longest path $\gamma_{ui}$ from $u$ to $i$. The starting time of operation $i$ is $\gamma_{ui}$.

The optimality of algorithm SS follows from Theorem 4. By using one of the algorithms of Karp and Orlin, Step 1 can be completed in $O(|N|^3)$ or $O(|E \cup R||N|\log|N|)$ time. (Recall that $E$ is the set of immediate precedence relations between operations and $R$ is the set of precedence relations between the last operation of an MPS and the first operation of the next MPS on a given machine.) Ahuja et al. (1993) develop a procedure to determine the longest paths from a node to every other node for a directed graph that does not have any circuit with positive length. By applying this procedure, Step 3 can be performed in $O(|E \cup R||N|)$ time. Since in a PCG $|E \cup R| = |N|^2$, the complexity of SS requires $O(|N|^3)$ or $O(|E \cup R||N|\log|N|)$ time.

### 3. Makespan Minimization

While total makespan is not widely used as a performance measure for periodic scheduling, if the goal is to complete all MPSs as quickly as possible, then this is an appropriate criterion. Given a machine processing order, the earliest starting time of an operation $i$ equals the length of the longest path from $u$ to $i'$ in the corresponding PCG. Consequently, the minimum total makespan of $n$ MPSs is the length in the PCG of the longest path from $u$ to $v$ that contains $n - 1$ recycling arcs. It is not known whether the number of steps to solve the constrained longest path problem and find the minimum total makespan is a polynomial function of $|N|$. (Garey and Johnson 1979 state that this problem where only simple paths are considered is NP-complete.) Nonetheless, there exists a simple scheduling rule, the earliest starting time rule, that minimizes the total makespan. As is mentioned in the introduction, there are important reasons to consider an earliest starting schedule. However, there are also some disadvantages. The schedule may not be periodic. Example 2 provides an instance where the earliest starting schedule is not periodic. The timing pattern of this schedule is given in Figure 6. A shortcoming of this schedule is that the MPS makespan grows without bound as the number of MPSs increases. Hence, when $n$ is big, a large number of MPSs are simultaneously in process.

The major results of this section are necessary and sufficient conditions for the MPS makespan of an arbitrary earliest starting schedule to be bounded. Consequently, when there is the constraint that the timing pattern is an earliest starting schedule, we are able to determine whether the number of MPSs simultaneously in process goes to infinity as $n$ goes to infinity. We establish these results by finding conditions for an earliest starting schedule to become periodic after a finite number of MPSs.

Since an arbitrary earliest starting schedule $s$ is considered, we implicitly assume that the PCG relates to the machine processing order of $s$. Now, we introduce some graph related concepts and some additional notation. A directed graph is called strongly connected if there is a directed path between every ordered pair of nodes. A component of a PCG is a maximal strongly connected subgraph. Since $u$ and $v$ are dummy nodes, $u$ and $v$ are not considered to be components. We define a critical component to be a component that contains a critical circuit. If there is a path from one component of a PCG to another component, then the former component precedes the latter component. An operation where the corresponding node in a PCG is part of a critical circuit is called a critical operation. We define a graph $G_c$ that represents the precedence relations among the components of a PCG as follows:

1. The node set is the set of components of a PCG.
2. If a PCG has a directed path from a node in component $g$ to a node in component $h$, then $(g, h)$ is an element of the arc set.

Note that $G_c$ is a forest and may have multiple root nodes. Tarjan (1972) shows that the root nodes can be found in $O(|E \cup R|)$ time. We call the component in a PCG that corresponds to a root node in $G_c$ a root component. The PCG for the problem presented in Figure 6 has two root components, one associated with each operation. Only the component associated with $e_1$ is critical.

In a PCG, let $\psi(i, j)$ denote a path from node $i$ to node $j$. Unless needed for clarity, we suppress the superscript that denotes the particular MPS of the node. Given a path (or circuit) $\psi$, the length of the path is given by $o(\psi)$, and the number of recycling arcs in the path is denoted by $q_\psi$. For example, the path $\psi(u, i')$ starts from operation $u$, ends at operation $i'$, and includes $q_{\psi} = r - 1$ recycling arcs. When considering a subpath of $\psi(u, i')$ that starts at node $j_1$ and ends at node $j_2$, we denote the subpath by $\psi(j_1, j_2)$.

The symbol $\oplus$ is a path composition operator. We use $\oplus$ in two situations. First, if $\psi_1$ and $\psi_2$ are two paths and the last node of $\psi_1$ is also the first node of $\psi_2$, then $\psi_1 \oplus \psi_2$ is the path $\psi_1$ followed by $\psi_2$. Second, if $\psi_2$ is a cycle that has
at least one node \( k \) in common with \( \psi_1(i, j) \), then \( \psi_1 \oplus \psi_2 \) is the path \( \psi_2(i, k) \) followed by \( \psi_2 \) followed by \( \psi_1(k, j) \). Notice that \( o(\psi_1 \oplus \psi_2) = o(\psi_1) + o(\psi_2) \).

Any path can be decomposed into a simple path and a number of circuits. We represent \( l \) instances of circuit \( \pi \) by \( \pi^l \). If \( \psi(u, i') \) can be decomposed into \( l \) instances of circuit \( \pi \) and \( \psi_0(u, i' - l \pi) \), then we write \( \psi(u, i') = \psi_0(u, i' - l \pi) \oplus \pi^l \). For instance, consider a path \( \psi(u, e_1^0) = (u, e_1^1, e_2^0, e_2^1, e_3^0, e_3^1, e_4^0, e_4^1, e_5^0, e_5^1) \) in the PCG of Example 1. The path includes a circuit \( \pi = (e_5^0, e_2^0, e_4^0, e_7, e_5^0, e_3^0, e_2^0) \) that has two recycling arcs. Therefore, \( \psi(u, e_1^0) = \psi_0(u, e_1^0) \oplus \pi^1 \), where \( \psi_0(u, e_1^0) = (u, e_1^0, e_3^0, e_2^0, e_3^1) \). Figure 7 illustrates the decomposition of \( \psi \). An example of a subpath of \( \psi \) is \( \psi(e_5^0, e_2^0, e_7) \). Figure 7 illustrates the decomposition of a path.

**Remark 2.** A PCG has \(|M|\) recycling arcs. Therefore, if \( r > |M| \), then every path from \( u \) to \( i' \) contains at least one circuit. Also, every simple path has no more than \(|M|\) recycling arcs.

For the remainder of this work we assume that \( x_0 = 0 \), i.e., processing starts at time zero. To investigate properties of an earliest starting schedule, we examine the relationship between a longest path and a critical circuit.

**Theorem 5.** If some root component of a PCG has no critical circuits, then the MPS makespan of an earliest starting schedule \( \to \infty \) as \( n \to \infty \).

**Proof.** Suppose that a root component \( h \) is not critical and has maximum circuit ratio \( \lambda' < \lambda \). Let \( \bar{C}_{\max}(n) \) and \( \bar{C}^2(n) \) be defined only for the component \( h \). Then by Theorem 2, for an operation \( i \) in component \( h \), \( x_i^0 + p_i \leq \bar{C}_{\max}(n) \leq n\lambda - (\bar{C}^2(n) - \lambda') \).

Consider a critical root component that has a critical circuit with \( q \) recycling arcs. From Remark 2, there exists a path from \( u \) to \( j \) in a PCG that intersects a critical circuit and has at most \(|M|\) recycling arcs. Consequently, for \( j \) in the critical root component, \( x_j^0 \geq q_j(n - |M| - qj) \lambda \geq n\lambda - |M|\lambda - q\lambda \). Since \( \lambda' < \lambda \) and \( n(\lambda - \lambda') - ((|M| + q\lambda - \bar{C}^2(n) + p_i) \leq x_j^0 \lambda' \), \( x_j^0 - x_i^0 \) diverges as \( n \to \infty \).

Let \( \lambda' \) be the maximum circuit ratio among all noncritical circuits in a PCG.

**Lemma 3.** If every root component of a PCG is critical, then for each

\[
r \geq \left\lceil \frac{\sum_{i \in E} p_i + 2|M|\lambda}{\lambda - \lambda'} \right\rceil,
\]

the longest path from \( u \) to \( i' \) contains a critical circuit.

**Proof.** In a PCG, let \( i \in N \) and

\[
r \geq \left\lceil \frac{\sum_{i \in E} p_i + 2|M|\lambda}{\lambda - \lambda'} \right\rceil.
\]

We construct a path \( \psi(u, i') \) that contains a critical circuit. Further, the length of \( \psi(u, i') \) is longer than \( \psi(u, i) \), the longest path without a critical circuit.

Since every root component is critical, there exists a critical circuit \( \pi_0 \) such that there is a path from some node of \( \pi_0 \) to \( i \). There is a path from \( u \) to a node of \( \pi_0 \) that does not include any recycling arc (the path that corresponds to the operations of the machine that processes the node of \( \pi_0 \)). Therefore, for some \( r_0 \), there exists a simple path \( \psi_0(u, i') \) that goes through a node of a critical circuit. Also, there exists a circuit \( \hat{\pi} \) that contains \( i \) where \( q_{\hat{\pi}} = 1 \). This circuit is defined by the operations that are performed on the machine that processes \( i \). Therefore, for \( r = r_0 + l_0q_{\hat{\pi}} \), there exists a path \( \psi(u, i') = \psi_0(u, i') \oplus \pi_0 \oplus \hat{\pi} \), where \( l_0 = [(r - r_0)/q_{\hat{\pi}}] \) and \( l = (r - r_0) \mod q_{\hat{\pi}} \). Since \( r_0 \leq |M| \) and \( l \leq |M| \) (see Remark 2),

\[
o(\psi(u, i')) \geq l_0q_{\hat{\pi}} \lambda = (r - r_0 - l_0q_{\hat{\pi}}) \lambda > (r - 2|M|) \lambda.
\]

Let \( \bar{\psi}(u, i') \) be the longest path from \( u \) to \( i' \) that does not contain any critical circuit. Then, \( \bar{\psi}(u, i') = \bar{\psi}_0(u, i') \oplus \pi_1 \oplus \cdots \oplus \pi_{k'} \), where \( \bar{\psi}_0 \) is a simple path, \( \pi_1, \pi_2, \ldots, \pi_{k'} \) are noncritical circuits, and \( \Sigma_{i=1}^{k'} q_{\pi_i} = r - r_1 \). By the definition of \( \lambda' \), \( o(\pi_i) \leq q_{\pi_i} \lambda' \) for \( i = 1, 2, \ldots, k' \). Since \( \psi_0(u, i') \) is a simple path, \( o(\psi_0(u, i')) < \Sigma_{i \in E} p_i \). Consequently, \( o(\psi(u, i')) < \Sigma_{i \in E} p_i + (r - r_1)\lambda' \). Therefore,

\[
o(\psi(u, i')) - o(\psi(u, i')) > (r - 2|M|) \lambda - \Sigma_{i \in E} p_i - (r - r_1)\lambda' \]

\[
\geq r(\lambda - \lambda') - 2|M| \lambda - \Sigma_{i \in E} p_i \geq 0.
\]

Hence, the longest path from \( u \) to \( i' \) contains a critical circuit.

The next theorem provides necessary and sufficient conditions for an earliest starting schedule to become periodic. Note that if the schedule becomes periodic, then the MPS makespan is bounded as \( n \to \infty \).

**Theorem 6.** Every root component of a PCG is critical if and only if the earliest starting schedule is periodic no later than

\[
\max\left\lceil \frac{\sum_{i \in E} p_i + 2|M|\lambda}{\lambda - \lambda'} \right\rceil, \quad |M|/2\right\rceil.
\]
An upper bound on the periodicity is the least common multiple of \(\{q_n|n|\) is a critical circuit\).

**Proof.** If a PCG has a root component that does not have any critical circuit, then by Theorem 5, the earliest starting schedule cannot be periodic. Therefore, suppose that every root component is critical. Let
\[
M = \{q_{\min}, \ldots, q_{\max}\}
\]
By Lemma 3, every longest path from \(u\) to \(v\) in a PCG contains a critical circuit. Let \(\psi(u, i')\) be a longest path in a PCG where \(\psi(u, i') = \psi(u, i')_0 = q_{\min} \oplus q_{\min} \oplus \cdots \oplus q_{\min}\) and \(q_{\min}\) is a critical circuit.

Let \(d = \text{least common multiple of } \{q_{n}|n|\) is a critical circuit\). Suppose \(\psi_i(u, i' + d) = \psi_i(u, i' + d) = q_{\min} \oplus q_{\min} \oplus \cdots \oplus q_{\min}\) is a longest path from \(u\) to \(i' + d\). Instead, suppose that another path \(\psi_i(u, i' + d) = \psi_i(u, i' + d) = q_{\min} \oplus q_{\min} \oplus \cdots \oplus q_{\min}\), \(\psi_i(u, i')\) is a simple path, \(\pi_0\) is a critical circuit, \(\pi_1, \pi_2, \ldots, \pi_t\) are circuits, and \(\pi_0\) is maximal among all longest paths from \(u\) to \(i' + d\) that contain \(\psi_i(u, i')\).

We now show that \(d = \text{least common multiple of } \{q_{n}|n|\) is a critical circuit\). Suppose \(\psi_i(u, i' + d) = \psi_i(u, i' + d) = q_{\min} \oplus q_{\min} \oplus \cdots \oplus q_{\min}\) is not a longest path from \(u\) to \(i' + d\). Instead, suppose that another path \(\psi_i(u, i' + d) = \psi_i(u, i' + d) = q_{\min} \oplus q_{\min} \oplus \cdots \oplus q_{\min}\), \(\psi_i(u, i')\) is a simple path, \(\pi_0\) is a critical circuit, \(\pi_1, \pi_2, \ldots, \pi_t\) are circuits, and \(\pi_0\) is maximal among all longest paths from \(u\) to \(i' + d\) that contain \(\psi_i(u, i')\).

Observe that the PCG for Example 1 has one component, and it is a critical root component. Theorem 6 establishes that the earliest starting schedule becomes periodic. As shown in Figure 1(b), this occurs at the second MPS.

Cohen et al. develop a restricted version of Theorem 6 using a linear system model based on minimax algebra. Also, Chretienne presents a result similar to Theorem 6 for timed Petri net models. In these models, there is no restriction on the number of operations that can be processed simultaneously on a single machine.

For our problem, the schedule may take many MPSs to become periodic (see Figure 8).

**Example 3**

There are four operations and two machines. Operation \(e_1\) has processing time 1, \(e_2\) has processing time 3, \(e_3\) has...
processing time 2, and $e_4$ has processing time $2 - \varepsilon$ where $1 > \varepsilon > 0$. Machine $M_1$ performs $e_1$ and $e_2$ in the order $(e_1, e_2)$, and $M_2$ performs $e_3$ and $e_4$ in the order $(e_3, e_4)$. There is a processing constraint that $e_1$ precedes $e_4$, i.e., $T = \{(e_1, e_4)\}$. The PCG for Example 3 is presented in Figure 8(a). This graph has two components $f$ and $g$, where the node sets of $f$ and $g$ are $\{e_1, e_2\}$ and $\{e_3, e_4\}$, respectively. Component $f$ is a critical root component and precedes $g$. Component $g$ is not critical and is not a root component.

The earliest starting schedule for Example 3 is presented in Figure 8(b). The critical circuit is $(e_1, e_2, e_1)$ and the critical circuit ratio is 4. The circuit $(e_3, e_4, e_3)$ is a noncritical circuit with ratio $4 - \varepsilon$. For $r = 1, 4 - \varepsilon$, this value decreases by $\varepsilon$ each time $r$ increases by one. The schedule becomes stable after $x_{e_4} = 1$. This requires $\lceil \frac{1}{\varepsilon} \rceil$ MPSs. Consequently, the number of MPSs needed for the schedule to become stable can be made arbitrarily large by choosing a sufficiently small $\varepsilon$.

Corollary 4 shows that if all processing times are integer, then $\left( \sum_{i \in N} p_i + 2|M| \lambda \right) (\lambda - \lambda')$ is bounded.

**Corollary 4.** Suppose every root component is critical. If all processing times are integer, then the earliest starting schedule is periodic no later than $\left( \sum_{i \in N} p_i + 2|M| \lambda \right) (\lambda - \lambda')$.

**Proof.** Let $\pi$ and $\pi_1$ be a critical circuit and a noncritical circuit, respectively. Since $q_{\pi} = q_{\pi_1} = |M|$ and $\omega(\pi)$ and $\omega(\pi_1)$ are integers,

$$\lambda - \lambda' = \frac{\omega(\pi)}{q_{\pi}} - \frac{\omega(\pi_1)}{q_{\pi_1}} = \frac{\omega(\pi)q_{\pi_1} - \omega(\pi_1)q_{\pi}}{q_{\pi}q_{\pi_1}} \geq \frac{\omega(\pi)q_{\pi_1} - \omega(\pi_1)q_{\pi}}{|M|^2} = \frac{1}{|M|^2}.$$

Consequently,

$$\left( \sum_{i \in N} p_i + 2|M| \lambda \right) (\lambda - \lambda') \leq \left( \sum_{i \in N} p_i + 2|M| \lambda \right) (\lambda - \lambda') = \left( \sum_{i \in N} p_i + 2|M| \lambda \right) (\lambda - \lambda').$$

The result now follows from Theorem 6. □

### 4. EARLIEST STABLE SCHEDULES

In the previous section, we found that the MPS makespan for an earliest starting schedule may be bounded under some conditions. In this section, we show that by delaying a minimal set of operations, the schedule can be modified not only to have a bounded MPS makespan but also to be stable. Further, this schedule has the minimum cycle time among all schedules with the same machine processing order. Consequently, if we select the earliest starting schedule with the machine processing order that gives a cycle time of $\lambda^*$, the modified schedule is a stable $\lambda^*$-schedule.

We first examine stable schedules where all operations except the initial operations $\{i| i \in A\}$ start as soon as possible, where $A = \{a(m)| m \in M\}$. We call such a schedule an earliest stable schedule. To implement an earliest stable schedule, timing control is unnecessary once the initial operations are appropriately delayed. Let $\tilde{N}$ be the set of all critical operations.

To find an earliest stable schedule, we construct two new graphs. Suppose that a stable schedule is given as $\{y_i| i \in N, r \geq 1\}$ where $y_i$ is the starting time of $e_i$. Let graph PCG’ be equivalent to PCG’ with the exception that the length of arc $(u, i)$ is $y_i^*$ if $i \in \tilde{N}$ and $-\infty$ otherwise. We denote the length of the longest path from $u$ to $i$ in PCG’ by $\gamma_u^*$. A second graph, PCG1, is equivalent to the PCG except that for each $i \in A$, the length of $(u, i)$ is $\gamma_u^*$.  

**Example 4**

Consider Example 3 where $\varepsilon = 1$, i.e., the processing time of $e_4$ is 1. Then, $\tilde{N} = \{e_1, e_2\}$ and $A = \{e_1, e_3\}$. We select a stable schedule where $y_1^* = 0, y_2^* = 1, y_3^* = 0$, and $y_4^* = 2$. In a PCG1’, arc $(u, e_1)$ has weight $y_1^* = 0$. Also, arc $(u, e_3)$ has weight $-\infty$ since $e_3 \notin \tilde{N}$. Consequently, $\gamma_{ue_1}^* = 0$ and $\gamma_{ue_3}^* = -2$. This provides arc weights in a PCG1 of 0 and $-2$ for arcs $(u, e_1)$ and $(u, e_3)$, respectively. The PCG1’ and PCG1 for Example 4 are illustrated in Figure 9.

![Figures 9. The PCG1’ and PCG1 for Example 4.](image-url)
We show that an earliest starting schedule constructed from a PCG1 is a stable schedule.

**Lemma 4.** In a PCG1, there exists a path from u to \( t \) that has length \( \gamma_{u,t} + (r - 1)\lambda \) for each \( i \in N \cap A \) and \( r \geq 1 \).

**Proof.** Let \( \pi \) be a critical circuit in a PCG1. We can write \( \pi \) as \( \pi(j_1, j_2) \oplus \pi(j_2, j_3) \oplus \cdots \oplus \pi(j_{q-1}, j_1) \), where \( \{j_1, j_2, \ldots, j_{q-1}\} \subseteq A \) is a subset of nodes in \( \pi \) and \( \pi(j_{i+1}, j_i) \) contains exactly one recycling arc for \( i = 1, 2, \ldots, q-1 \) (for expositional ease we assume that \( j_{q+1} = j_1 \)).

We first show that \( \gamma_{u,j_l} = y_j^1 \) for all \( l = 1, 2, \ldots, q \). Since in a PCG1 arc (u, j_l) has length \( y_j^1 \), suppose that \( \gamma_{u,j_l} > y_j^1 \) for some \( l \). This implies that there exists a path \( \psi(u, j_l) \) in a PCG1 such that \( \omega(\psi(u, j_l)) = q_j \lambda > y_j^1 \). Consequently, \( y_{j_l}^1 \geq \omega(\psi(u, j_l)) > y_j^1 + q_j \lambda \). This contradicts the assumption that \( \{\gamma_{u,i} \mid i \in N, r \geq 1\} \) are the starting times of a stable schedule. Therefore, \( \gamma_{u,j_l} = y_j^1 \) for all \( l = 1, 2, \ldots, q \).

Without loss of generality, we establish the hypothesis for \( j_1 \in N \cap A \). Let \( h = q_0 + 1 - (r - 1) \mod q_\pi \). Then in a PCG1, there exists a path \( \phi(u, j_1') \) = \((u, j_1) \oplus \pi(j_1, j_2) \oplus \cdots \oplus \pi(j_{q-1}, j_1)\). This path is depicted in Figure 10. We show that the length of this path is \( \gamma_{u,j_1'} + (r - 1)\lambda \).

By the definition of a PCG1, \( \omega(\phi(u, j_1')) = \gamma_{u,j_1'} \). From the analysis above, \( \omega(\phi(u, j_1')) = y_j^1 \). Since \( \{y_j^1\} \) are the starting times of a stable schedule and \( \pi \) is a critical circuit,

\[
y_{j_1}^1 = y_j^1 - \lambda = y_j^1 + \omega(\pi(j_h, j_{h+1}))) - \lambda
\]

\[
y_{j_1}^2 = y_j^2 - \lambda = y_{j_h}^1 + \omega(\pi(j_{h+1}, j_{h+2})) - \lambda
\]

\[
\vdots
\]

\[
y_{j_1}^l = y_j^1 - \lambda = y_j^1 + \omega(\pi(j_{q-1}, j_1))) - \lambda
\]

Since \( \gamma_{u,j_1} = y_j^1 \) for all \( l = 1, 2, \ldots, q_\pi \), \( \omega(\phi(u, j_1')) = \omega((u, j_h)) + \omega(\pi(j_{h+1}, j_1)) + \omega(\pi(1 \to q_{\phi} \mod q_\pi))\) + \((q_{\phi} + 1 - h)\lambda\).

\[
= y_{j_h}^1 + (y_{j_h}^1 - y_{j_1}^1 + (q_{\phi} + 1 - h)\lambda)
\]

\[
= y_{j_h}^1 + [(r - 1) \mod q_\pi] \lambda + \left[\frac{r - 1}{q_\pi}\right] q_\pi \lambda
\]

\[
= \gamma_{u,j_1'} + (r - 1)\lambda. \quad \square
\]

**Theorem 7.** Let \( \{y_{u,i} \in N, r \geq 1\} \) be the starting times of a stable schedule. Suppose every root component of a PCG is critical. Then in a PCG1, the length of the longest path from \( u \) to \( t \) is \( \gamma_{u,t} + (r - 1)\lambda \).

**Proof.** The proof has two parts. We first show that for each \( i \in N \) and \( r \geq 1 \) there exists a path from \( u \) to \( i \) in a PCG1 with length \( \gamma_{u,i} + (r - 1)\lambda \). Then, we show that the path is a longest path from \( u \) to \( i \).

We now establish the first part. Since every root component is critical, for each \( i \in N \) there exists a path from \( u \) to \( i \) in a PCG1. Let \( \psi(u, i) \) be a longest such path. For each \( i \in N \cup A \), there exists a path \( \phi(u, j_i) \) = \((u, j_0) \oplus \pi(j_0, j_1) \oplus \cdots \oplus \pi(j_{q-1}, j_1)\). This path is depicted in Figure 10. We show that the length of this path is \( \gamma_{u,i} + (r - 1)\lambda \).

By the definition of a PCG1, \( \omega(\phi(u, j_i)) = \gamma_{u,i} \). From the analysis above, \( \omega(\phi(u, j_i)) = y_j^1 \). Since \( \{y_j^1\} \) are the starting times of a stable schedule and \( \pi \) is a critical circuit,

\[
y_{j_1}^1 = y_j^1 - \lambda = y_j^1 + \omega(\pi(j_h, j_{h+1}) - \lambda
\]

\[
y_{j_1}^2 = y_j^2 - \lambda = y_{j_h}^1 + \omega(\pi(j_{h+1}, j_{h+2})) - \lambda
\]

\[
\vdots
\]

\[
y_{j_1}^l = y_j^1 - \lambda = y_j^1 + \omega(\pi(j_{q-1}, j_1))) - \lambda
\]

Since \( \gamma_{u,j_1} = y_j^1 \) for all \( l = 1, 2, \ldots, q_\pi \), \( \omega(\phi(u, j_i)) = \omega((u, j_h)) + \omega(\pi(j_{h+1}, j_1)) + \omega(\pi(1 \to q_{\phi} \mod q_\pi))\) + \((q_{\phi} + 1 - h)\lambda\).

\[
= y_{j_h}^1 + (y_{j_h}^1 - y_{j_1}^1 + (q_{\phi} + 1 - h)\lambda)
\]

\[
= y_{j_h}^1 + [(r - 1) \mod q_\pi] \lambda + \left[\frac{r - 1}{q_\pi}\right] q_\pi \lambda
\]

\[
= \gamma_{u,i} + (r - 1)\lambda. \quad \square
\]
Figure 11. Paths for Theorem 7.

\[ \omega((u, k_t) \oplus \psi(k_t, i)) = \gamma'_{uk_t} + \omega(\psi(k_t, i)) \]

\[ = \omega(\psi(u, k_t)) - t\lambda + \omega(\psi(k_t, i)) \]

\[ = \omega(\psi(u, i)) - t\lambda \]

\[ = \omega(\psi(u, i)) - q_{\phi}\lambda + (r - 1)\lambda \]

\[ = \gamma'_{ui} + (r - 1)\lambda. \]

Suppose that \( r > q_{\psi} + 1 \). Then, \(-t = r - 1 - q_{\phi} > 0\). As suggested by Lemma 4, because \( k_0 \in N \cap A \), we let \( \phi(u, k_0^{t+1}) \) be a path with length \( \gamma'_{uk_0} + (-t)\lambda \) in a PCG1. Then, a path from \( u \) to \( i' \) in a PCG1 is \( \phi(u, k_0^{t+1}) \oplus \psi(k_0, i) \). This path is depicted in Figure 11(b). The length of the path is

\[ \omega(\phi(u, k_0^{t+1}) \oplus \psi(k_0, i)) \]

\[ = (\gamma'_{uk_0} + (-t)\lambda) + (\gamma'_{ui} + q_{\phi}\lambda - \gamma'_{uk_0}) \]

\[ = \gamma'_{ui} + (q_{\phi} - r)\lambda \]

\[ = \gamma'_{ui} + (r - 1)\lambda. \]

This completes the first part of the proof.

We now prove that a path from \( u \) to \( i' \) in a PCG1 with length \( \gamma'_{ui} + (r - 1)\lambda \) is the longest path. Consider another path \( \phi(u, i') \) in a PCG1, where \( (u, k) \) is the first arc of \( \phi \). There exists a path \( \phi_1(u, k) \) such that \( \omega(\phi_1(u, k)) - q_{\phi}\lambda = \gamma'_{uk} \). Hence,

\[ \omega(\phi(u, i')) = \gamma'_{uk} + \omega(\phi(k, i)) \]

\[ = \omega(\phi_1(u, k)) - q_{\phi}\lambda + \omega(\phi(k, i)) \]

\[ = \omega(\phi_1(u, k) \oplus \phi(k, i)) - q_{\phi}\lambda. \]

By the definition of \( \gamma'_{ui} \),

\[ \gamma'_{ui} = \omega(\phi_1(u, k) \oplus \phi(k, i)) - (q_{\phi_{1\lambda}} + q_{\phi}) \lambda \]

Consequently, \( \gamma'_{ui} + q_{\phi}\lambda = \gamma'_{ui} + (r - 1)\lambda \geq \omega(\phi(u, i')) \).

Hence, the length of the longest path from \( u \) to \( i' \) in a PCG1 is \( \gamma'_{ui} + (r - 1)\lambda \).

Theorem 8 summarizes the prior results and shows how to compute the starting times of the initial operations to construct a stable schedule.

**Theorem 8.** Given a machine processing order, there exists an earliest stable schedule if and only if every root component of the PCG is critical. Further, an earliest starting schedule where each initial operation \( i \) starts at \( x_1^i = \gamma_{ui} - \min_{k \in A} \gamma_{uk} \) is an earliest stable schedule that starts at time 0 and has cycle time \( \lambda \).

**Proof.** If there exists a root component that does not have any critical circuit, then by Theorems 5 and 6, the MPS makespan is not bounded for any finite delay of the initial operations. Hence, there does not exist an earliest stable schedule.

Suppose that every root component is critical. Subtracting \( \min_{k \in A} \gamma_{uk} \) from each starting time of the schedule found by Theorem 7 shifts the starting times of the schedule so that the schedule begins at time zero. Since the longest path from \( u \) to \( i' \) in PCG1 is \( \gamma_{ui} + (r - 1)\lambda \), it follows that the earliest starting schedule that begins each initial operation \( i \) at \( x_1^i = \gamma_{ui} - \min_{k \in A} \gamma_{uk} \) has cycle time \( \lambda \).

We present an algorithm based on Theorem 8 to determine the starting times of the initial operations.

**Algorithm S1:** Finds the starting times of the first job on each machine to generate an earliest stable schedule \( \{x_1^i | i \in N, r = 1\} \) when every root component is critical.

1. Construct a stable schedule \( \{y_1^i | i \in N, r = 1\} \) for the first MPS.
2. Identify \( G_C \) from the PCG.
3. Find all critical circuits of the PCG.
   a) Construct the PCG'.
   b) Find all circuits with length zero in the PCG'. Let \( N \) = the set of nodes in these circuits.
4. Construct the PCG1' using \( \{y_i^j \mid i \in \hat{N} \cap A \} \). Compute \( y_i^j \) for all \( i \in A \).
5. Compute \( x_i^j = y_i^j - \min_{k \in \pi} y_{i,k} \) for all \( i \in A \).

An earliest stable schedule is generated by starting the initial operations as specified by Step 5 and all other operations as soon as possible.

For Step 1, any stable schedule can be selected. We suggest that algorithm SS is used to find the stable schedule. Step 2 can be performed by using the algorithm of Tarjan. The correctness of Step 3 follows directly from Lemma 2 because a longest circuit in the PCG' has length zero. In Step 3b, all the circuits with zero length in the PCG' can be found by finding the longest path between every pair of nodes using the Floyd-Warshall algorithm in \( O(|N|^3) \) time (see Floyd 1962, Warshall 1962, and Lawler). Step 4 can be performed in \( O(|E \cup R||N|) \) time by using the algorithm of Ahuja et al. If we use algorithm SS for Step 1, it can be verified that algorithm S1 requires \( O(|N|^3) \) time.

**Example 4** (continued). The earliest starting schedule that is generated by the PCG for Example 4 (see Figure 8(a) where \( e = 1 \)) is presented in Figure 12(a). Since \( y_{e,1} = 0 \) and \( y_{e,2} = -2 \), an earliest stable schedule is generated by a PCG' (see Figure 9) and is shown in Figure 12(b). Since \( \min_{k \in \pi} y_{i,k} = -2 \), this schedule is shifted right by two time units to get the schedule depicted in Figure 12(c), an earliest stable schedule that starts at time zero.

When a PCG has a noncritical root component, neither does the earliest starting schedule become periodic nor does an earliest stable schedule exist (see, for instance, Example 2 or Figure 6). However, we can construct an earliest stable schedule by algorithm S1 if for each MPS some operations are delayed. To delay an operation, the PCG is modified by increasing the processing time of an appropriate arc. We present a procedure to modify the PCG so that every root component is critical and an earliest stable schedule exists.

**Algorithm MS1**: Modifies the PCG so that every noncritical root component becomes critical.

1. Find the set of all noncritical root components in the PCG. Denote the set by \( Q \).
2. Select a component of \( Q \). Find the circuit \( \pi \) in the component with the maximum circuit ratio. Let \( \delta = q_+ \lambda - \sum_{(i,j) \in \pi} p_i \).
3. Select a machine \( m \) such that \((b(m), a(m)) \in \pi\). Increase the first weight of arc \((b(m), a(m))\) by \( \delta \).
4. Repeat Steps 2 and 3 for each component in \( Q \).

In Step 1, all root components can be found in \( O(|E \cup R|) \) time by using the algorithm of Tarjan. The identification of all noncritical root components and the circuit \( \pi \) with the maximum circuit ratio in Steps 1 and 2 can be accomplished by using the Floyd-Warshall algorithm and Step 3 of algorithm S1. Therefore, algorithm MS1 runs in \( O(|N|^3) \) time.

Notice that for Example 2, MS1 adds one unit of processing time to operation \( e_2 \).

**Theorem 9.** Every root component of an MPCG is critical.

**Proof.** Let \( \pi \) be the circuit in a noncritical root component that is chosen by Step 2 of algorithm MS1. Since the first weight of recycling arc \((b(m), a(m)) \in \pi\) increases by \( q_+ \lambda - \sum_{(i,j) \in \pi} p_i \), \( \pi \) becomes a critical circuit. No other circuit ratio increases beyond \( \lambda \) because \( \pi \) has the maximum circuit ratio within the noncritical root component.

**5. INTERRELATIONSHIPS WITH THE SEQUENCING DECISION**

We have shown that the timing decision and the sequencing decision can be considered separately. Once the machine processing order has been determined (whether optimal or not), the best timing decision can be determined. We observe that the set of root components does
Figure 13. Schedules for two machine processing orders.

not change with the choice of the machine processing order. However, the critical circuit can change. Consequently, the machine processing order can have a profound effect on the cycle time. A poor sequencing decision can result in an arbitrarily large cycle time. Consider a periodic job shop problem with $l = |M|$ items and $l$ machines. Each item $j$ has $l$ operations, $e_{1j}, e_{2j}, \ldots, e_{lj}$ for $j = 1, 2, \ldots, l$. Further, operation $e_{ij}$ is processed on machine $1 + (i - j) \mod l$. Technology considerations require that the operations are processed in the order $e_{1j}, e_{2j}, \ldots, e_{lj}$ for each item $j = 1, 2, \ldots, l$. Also, each operation takes one unit of processing time. The cycle time is minimized by processing the items on machine $i$ in the order $e_{1, (i-1) \mod l}, e_{2, (2-i) \mod l}, \ldots, e_{l, (l-i) \mod l}$ for $i = 1, 2, \ldots, l$. As shown in Figure 13(a), the minimum cycle time for this machine processing order is $l$ units of time. Consequently, the ratio of the cycle time associated with a suboptimal machine processing order to an optimal machine processing order can be arbitrarily large.

However, there are classes of problems where the sequencing decision has no effect on the cycle time. Consider a digraph $G_T$ where there are $|M|$ vertices, one corresponding to each machine. A directed arc $(m, m')$ exists in $G_T$ if there is some operation $i$ performed on machine $m$ and operation $j$ performed on machine $m'$ where $m \neq m'$ and $(i, j) \in T$. Recall that $T$ is the set of routings of items through the machines.

**Theorem 10.** If $G_T$ has no directed cycles, then the minimum cycle time is identical for any machine processing order.

**Proof.** Observe that for a specified $T$, there is a set of associated PCGs, one for each machine processing order. If $G_T$ has no directed cycles, then there is no cycle that contains operations from more than one machine in any associated PCG. Consequently, each elementary cycle of
the PCG contains exactly one recycling arc. Further, the vertices of the cycle correspond to the operations performed by the machine with the recycling arc. For a specified machine, the value of the circuit ratio is just the sum of the processing times of the operations performed on that machine. Thus, the value of the critical circuit ratio, \( \lambda^* \), is independent of the order in which the operations are chosen. By Theorem 1, the minimum cycle time is \( \lambda^* \). \( \square \)

Note that general flow shops satisfy the conditions of Theorem 10.

There can be various factors that restrict the selection of the machine processing order. In some instances, the problem is to find a machine processing order that is compatible with specified timing pattern constraints. Consider an extended version of Example 1 where there is a fourth machine. On this machine we have one new operation, \( e_8 \), with a processing time of 7.5. Further, there are no precedence relations between this new operation and the other operations. Suppose there is a requirement that the timing pattern be a periodic earliest starting schedule. The PCG for this problem has two root components (one with \( e_1, e_2, \ldots, e_7 \), and one with \( e_8 \)). Since the second component has a cycle time of 7.5, any machine processing order that has a cycle time no larger than 7.5 produces a schedule with minimum cycle time. Theorem 6 suggests that we might search for a machine processing order on the first three machines with a cycle time of 7.5. One possible solution is to change the processing order on machine 3 from \((e_3, e_4, e_7)\) to \((e_3, e_5, e_4)\). Since all root components are critical, we have found an earliest starting schedule that is periodic.

Consider another instance (the details of the problem are omitted) where the objective is to select a machine processing order that gives minimum cycle time subject to the constraint that the schedule is an earliest stable schedule. In this instance, the PCG has two root components. Various machine processing orders for the first component give cycle times of 6, 7, 9, and 10, and machine processing orders for the second component give cycle times of 5, 7, and 8. Since both components must have the same cycle time (see Theorem 8), an optimal solution is to select the machine processing order that gives a cycle time of 7 for each component. Observe that if there were no timing constraints, then a schedule that has a cycle time of 6 would be chosen.

6. SUMMARY AND FINAL REMARKS

An appropriate schedule should be implemented depending upon the availability of timing control and the priority among the objectives. To generate schedules with minimal cycle time, good timing decisions are essential. The theory that we develop represents an important step toward solving the general periodic sequencing problem. In this paper, we characterize the performance measures associated with periodic scheduling. To find the cycle time, we represent the processing of a general periodic job shop by a directed graph. Then a linear program is constructed. By using linear programming duality, the minimum cycle time is characterized as a circuit measure of the graph and can be computed in polynomial time. We show that there are a variety of schedules with minimum cycle time.

Based on our discussion, cycle time is a reasonable performance measure. Among the schedules with minimum cycle time, we show that there always exists a schedule that is stable. This implies that the problem of determining the machine processing order for periodic scheduling reduces to a conventional sequencing problem for a single MPS with a new performance measure, the cycle time. It can be seen that the cycle time is the "maximum schedule width" of a single MPS. Also, the difference in completion time between a schedule with minimum cycle time and one with minimum total makespan is less than one cycle, and is a constant independent of \( n \). We develop an efficient procedure to construct a schedule with minimum cycle time that is stable, and that simultaneously minimizes the total makespan, the MPS makespan, and the average number of MPSs simultaneously in production as secondary measures. A by-product of this research is a polynomial time procedure that makes a schedule stable without increasing the cycle time.

When it is desirable to expend minimal effort in the control of the timing of the operations, an earliest starting schedule is a good choice. This schedule minimizes the total makespan and also is easy to implement. Unfortunately, the number of MPSs simultaneously in production may grow without bound and the schedule may not be stable.

We can check whether an earliest starting schedule becomes periodic and we have a bound on the number of MPSs that are needed for this to occur. Also, given a schedule, we can construct in polynomial time a new schedule that delays a minimal number of jobs, that is stable and that has the same cycle time as the original schedule. Hence, in this new schedule, the WIP is bounded as the number of MPSs go to infinity. If the machine processing order gives a cycle time of \( \lambda^* \), then the new schedule is a \( \lambda^* \)-schedule that is an earliest stable schedule.

Our analysis can help in the selection of a machine processing order that is compatible with specified timing constraints. Also, we have described a class of problems where the cycle time does not depend on the sequencing decision.

Measures of shop performance that have not been discussed include maximum workload and machine utilization. The workload of a machine is the sum of the processing times of the operations of an MPS performed on the machine. A PCG has a circuit corresponding to the operations performed on a machine. Such a circuit has a single recycling arc and the sum of the first arc weights is the same as the workload of the machine. Consequently, the minimum cycle time is at least as large as the maximum workload.
Given a schedule with cycle time $\mu$, the total time spent by the machines to produce an MPS is $|M|/\mu$. Machine utilization equals $\sum_{i=1}^{m} p_i / |M|$.$\mu$. Since the machine utilization is inversely related to the cycle time, shorter cycle times lead to higher machine utilization.

We note that there are situations where a stable schedule with real-valued cycle time is not implementable. Hanen states that in pipelined computer scheduling, the computer clock time is the minimum time unit. In a computer controlled manufacturing shop as well as most conventional shops, operations may be allowed to start only at a multiple of a basic time unit. In such situations, it may not be possible to implement a stable schedule with real-valued cycle time even though processing times of operations are integer. Hanen shows that for any stable schedule with starting times $\{\gamma_i^r\} \in N, r \geq 1$ and cycle time $\mu = a/b$, there exists a periodic schedule with cycle time $\mu$ and starting times $\{\gamma_i^r\} \in N, r \geq 1$ such that $\gamma_i^r = \gamma_i^1$ and $\gamma_i^{r+b} - \gamma_i^r = a = b\mu$ for all $i \in N$ and $r \geq 1$, where $a$ and $b$ are the minimal integers such that $\mu = a/b$. As seen from problem $P$, the cycle time $\lambda$ is always rational if the processing times of operations are rational. Therefore, from a stable schedule with noninteger starting times, we can derive a periodic schedule with integer starting times.

**ACKNOWLEDGMENT**

This research has been supported in part by IBM System Integration Division, Owego. The authors thank IBM for their generous support; and Nicholas G. Hall, Charles H. Reilly, and Rakesh Vohra for suggesting numerous improvements to this work.

**REFERENCES**


