

# Cyclic Production Systems: Scheduling and Performance Tradeoffs

Jeffrey W. Herrmann  
Department of Mechanical Engineering  
University of Maryland  
College Park, Maryland 20742 USA

## Abstract

A cyclic production system manufactures a set of products at a constant frequency. Each resource repeatedly performs a certain set of tasks, and the system produces a certain set of units each cycle. The production schedule defines the timing of the tasks within each cycle, which determines system performance on measures such as throughput and work-in-process inventory. This paper describes the tradeoffs that occur when scheduling such systems. This paper presents a problem space search algorithm that explores the tradeoffs between throughput and inventory.

## Keywords

Cyclic schedules, multi-objective optimization, search algorithms.

## 1. Introduction

A cyclic production system manufactures a set of products at a constant frequency. That is, each resource repeatedly performs a certain set of tasks, and the system produces a certain set of units each cycle. Cyclic scheduling is a dynamic rescheduling environment that has no arrival variability [8].

This paper addresses the following type of cyclic production system. The set of tasks does not change, the tasks are non-preemptive, and the task processing times are deterministic. Each task requires exactly one resource, and there are sufficient buffers to hold parts that are not being processed. All resources are perfectly reliable, and each can process at most one task at a time. The processing requirements of different products impose task precedence constraints, and the corresponding sequences of resources may be different for each product. (That is, the system is a general job shop.) In each cycle the system completes one unit of each product. Note that a single unit may require multiple cycles to complete all tasks. A unit may be one part or a batch of multiple parts that must be processed together.

A schedule defines the length of the cycle and the start time (within a cycle) of each task. The schedule is stable: the (relative) start times within each and every cycle are the same. That is, the interval between consecutive start times of each task remains the same. Careful scheduling is required to coordinate activities so that the system completes units at the highest possible throughput (frequency) and with the minimal amount of work-in-process inventory (WIP). Unfortunately, these two objectives conflict. Note that maximizing throughput is the same as minimizing the cycle length; many authors call this length the cycle time. To avoid confusion, this paper uses the term *cycle length*. In addition, we use the term *flow time* (not cycle time) to describe the time that a unit spends in the production system between the start of its first task and the completion of its last task.

Although optimizing either one of the objectives is straightforward, combining the objectives yields a much more difficult problem [4, 5, 6]. Based on these results, it is clear that achieving both minimal WIP (where no unit waits) and maximal throughput is strongly NP-hard. Chauvet, Herrmann, and Proth [1] aim at finding a cyclic schedule that maximizes throughput subject to a constraint on work in process. Their heuristic approach based on Petri nets finds a near-optimal solution, and they give a sufficient condition for optimality.

Section 2 formulates the cyclic production scheduling problem and discusses the tradeoffs that exist between the objective functions. Section 3 presents the schedule generation algorithm and a special case where maximizing throughput and minimizing WIP is easy. Section 4 presents a problem space search algorithm that generates a sequence of solutions and gives a decision-maker the ability to make the tradeoff between WIP and throughput.

Section 5 discusses the results of experiments conducted to evaluate the performance of this search algorithm. Section 6 concludes the paper. Due to space constraints, the paper includes neither illustrative examples nor proofs. More details can be found in Herrmann, Chauvet, and Proth [2].

## 2. Problem Formulation

The cyclic production system has  $R$  resources and produces  $n$  different products. To produce these units, the system must perform  $e$  different tasks each cycle. Note  $R \leq e$  and  $n \leq e$ . Let  $r = 1, \dots, R$  index the resources. Let  $b = 1, \dots, n$  index the products. Let  $i = 1, \dots, e$  index the tasks. Let  $r(i)$  be the resource that performs task  $i$ , which requires  $d(i)$  time units. Let  $S(b)$  be the sequence of tasks required to produce one unit of product  $b$ . Let  $q(b)$  be the number of tasks in  $S(b)$ . Note  $q(1) + \dots + q(n) = e$ . Let  $E(b)$  be the set of immediate precedence constraints. If  $(i, j)$  is in  $E(b)$ , then task  $j$  follows task  $i$  in  $S(b)$ . Then,  $j = \text{succ}(i)$ , and  $i = \text{pred}(j)$ . Let  $D(b)$  be the sum of the  $d(i)$  for all tasks  $i$  in  $S(b)$ , and let  $C(r)$  be the total processing requirements on resource  $r$ :

$$D(b) = \sum_{i \in S(b)} d(i) \quad (1)$$

$$C(r) = \sum_{r(i)=r} d(i). \quad (2)$$

Let  $C^*$  be the maximum processing requirements:  $C^* = \max \{C(1), \dots, C(R)\}$ . Note that  $D(1) + \dots + D(n) = C(1) + \dots + C(R)$ . Given the set of tasks and resources, a feasible cyclic schedule specifies the start time  $x(i)$  and end time  $y(i)$  of each required task  $i$ ,  $i = 1, \dots, e$ , and a length  $C$  that satisfies the following constraints for all  $i$ :  $0 \leq x(i) < C$ .  $y(i) = x(i) + d(i)$ . If  $y(i) \leq C$ , then resource  $r(i)$  is executing task  $i$  in the interval  $[x(i), y(i)]$ . Otherwise, resource  $r(i)$  is executing task  $i$  in the interval  $[x(i), C]$  and in the interval  $[0, y(i)-C]$ . In either case, resource  $r(i)$  is executing no other task when it is executing task  $i$ . (Note that we use the interval  $[0, C]$  for convenience, since the cycle is continuously repeated and any interval of length  $C$  would be sufficient for defining the cyclic schedule.)

The relevant system performance measures are system throughput, average product flow time, and average work-in-process inventory (WIP). The system throughput equals the number of units produced per time unit. Since a cycle of length  $C$  produces  $n$  units, the throughput  $\text{TH} = n/C$ . For a feasible cyclic schedule with length  $C$ , we measure WIP by first calculating the average product flow time and then employing Little's Law, as described in the following algorithm:

Step 1. For each product  $b = 1, \dots, n$ , perform Step 1a for the first task in  $S(b)$  and perform Step 1b for each remaining task. Then go to Step 2.

Step 1a. Let  $i$  be the first task in  $S(b)$ . Assign the label  $w(i) = 0$  to this task.

Step 1b. Let  $i$  be the next unlabeled task in  $S(b)$  and let  $j = \text{pred}(i)$ . Label task  $i$  as follows: If  $y(j) > x(i)$ , then let  $w(i) = w(j) + 1$ . Otherwise, let  $w(i) = w(j)$ .

Step 2. For each product  $b = 1, \dots, n$ , let  $g$  be the first task in  $S(b)$  and let  $h$  be the last task in  $S(b)$ . Then, the product flow time  $Q(b) = w(h)C + y(h) - (w(g)C + x(g))$ . Since  $w(g) = 0$  by definition,  $Q(b) = w(h)C + y(h) - x(g)$ .

Step 3. The average product flow time is  $(Q(1) + \dots + Q(n))/n$ . The average WIP is  $(Q(1) + \dots + Q(n))/C$  units.

**Theorem 1.** *In any feasible cyclic schedule, the cycle length must be at least  $C^*$ , the throughput cannot be greater than  $n/C^*$ , and the average product flow time must be greater than or equal to  $F^* = (D(1) + \dots + D(n))/n$ .*

**Corollary 1.** *Let  $\text{TH}^* = n/C^*$ . Then, a feasible cyclic schedule that achieves the maximal throughput of  $n/C^*$  must yield a WIP greater than or equal to  $W^* = F^* \text{TH}^* = (D(1) + \dots + D(n))/C^*$ .*

In general, a feasible cyclic schedule that achieves a throughput of  $\text{TH}$  must have a WIP greater than or equal to  $F^* \text{TH}$ . Thus, the point  $(\text{WIP}', \text{TH}')$  for any feasible cyclic schedule will lie on or to the right of the line  $\text{WIP} = F^* \text{TH}$  and on or below the line  $\text{TH} = n/C^*$ , the dotted lines shown in Figure 1. In the set of all feasible schedules the non-dominated solutions will form a tradeoff curve between WIP and throughput like those points connected by the solid line in Figure 1.

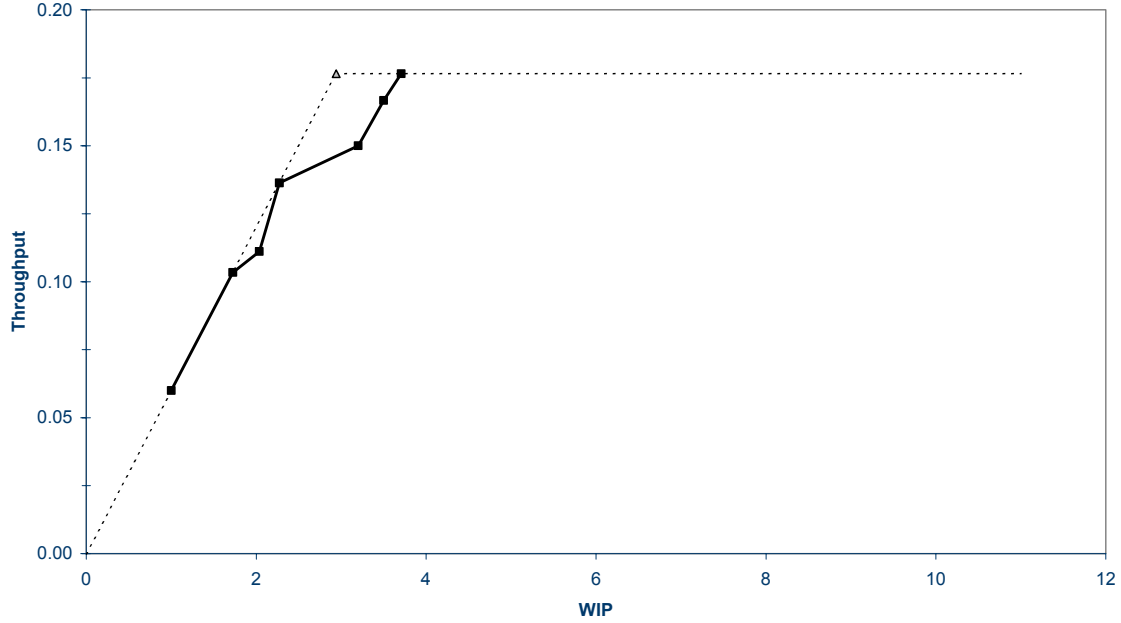


Figure 1. Typical throughput versus WIP tradeoff

### 3. The Shop Scheduling Algorithm

This section presents a heuristic to find a feasible cyclic schedule that completes all tasks for each unit within one cycle. Note that no task sequences are given for any resource. The *shop scheduling algorithm* proceeds as follows:

Step 1. Let  $V$  be the set of completed tasks.  $V = \{\}$ . Let  $W$  be the set of remaining tasks.  $W = \{1, \dots, e\}$ . Let  $U$  be the set of ready tasks. Set  $U = \{\}$ . For each product  $b = 1, \dots, n$ , let  $j$  be the first task in  $S(b)$ , set  $z(j) = 0$ , and add task  $j$  to  $U$ .

Step 2. Let  $A(r)$  denote the available time of resource  $r$ . Set  $A(r) = 0$  for all  $r = 1, \dots, R$ .

Step 3. For each  $j$  in  $U$ , determine the earliest start time  $x(j)$  as follows:  $x(j) = \max\{z(j), A(r(j))\}$ .

Step 4. Let  $x^* = \min\{x(j) : j \text{ in } U\}$ . If there exist  $j$  and  $k$  in  $U$  and a resource  $r$  such that  $x(j) = x(k) = x^*$  and  $r(j) = r(k) = r$ , then go to Step 5. Otherwise, pick any  $j$  in  $U$  such that  $x(j) = x^*$  and go to Step 6.

Step 5. Let  $U^*(r)$  be the tasks  $j$  in  $U$  such that  $x(j) = x^*$  and  $r(j) = r$ . Using some rule, pick a task  $j$  from  $U^*(r)$ .

Step 6. Let  $y(j) = x(j) + d(j)$ . Increase  $A(r(j))$  to  $y(j)$ . Remove  $j$  from  $W$ , remove  $j$  from  $U$ , and add  $j$  to  $V$ . If  $\text{succ}(j)$  exists, let  $z(\text{succ}(j)) = y(j)$  and add  $\text{succ}(j)$  to  $U$ . If  $U$  is empty, stop. Otherwise, go to Step 3.

The cycle length  $C = \max\{y(1), \dots, y(e)\}$ . Let us define for each task  $i$  the quantity  $wr(i)$  (the work remaining) as follows:  $wr(i) = d(i) + wr(\text{succ}(i))$  if  $\text{succ}(i)$  exists;  $wr(i) = d(i)$  if  $\text{succ}(i)$  does not exist. The rule used in Step 5 is the most work remaining (MWR) rule: select one of the tasks that has the maximal value of  $wr(j)$ . The shop scheduling algorithm has to schedule  $e$  tasks, and the set  $U$  has at most  $n$  tasks in it, so the shop scheduling algorithm requires  $O(ne)$  effort.

**Theorem 2.** The WIP of the cyclic schedule created by the shop scheduling algorithm is no greater than  $n$ .

**Theorem 3.** If  $q(b) = 1$  for all  $b = 1, \dots, n$ , then the shop scheduling algorithm creates a feasible cyclic schedule with length  $= C^*$ , throughput  $= TH^*$ , and WIP  $= W^*$ .

#### 4. A Tradeoff Exploration Search in Problem Space

This section presents a search algorithm that explores the tradeoffs between WIP and throughput. The search explores a problem space similar to those considered by Storer *et al.* [7] and Herrmann and Lee [3]. Searching the problem space exploits a heuristic's ability to find good solutions based on the structure of the problem but allows exploration to find even better solutions. The search algorithm finds a sequence of feasible cyclic schedules. The goal is to provide a set of schedules that have different values of WIP and throughput so that a decision-maker can select the most appropriate one.

The initial solution is the solution that the shop scheduling algorithm generates from the problem instance. At each iteration, the search algorithm creates a number of new instances, uses the shop scheduling algorithm to find a solution for each, and selects the instance that yielded the shortest cycle length. To create a new instance, it removes the precedence constraint on a task along a critical path. That is, the algorithm creates a new trial instance by splitting a product sequence and adding a new product for the second part of the sequence. Eventually, the final solution will be a maximal throughput solution, which must happen when all of the precedence constraints are removed, according to Theorem 3.

The *tradeoff exploration search* proceeds as follows:

Step 1. Let  $P(0)$  be the original problem instance. Let  $n(0) = n$  products. Use the shop scheduling algorithm (with the MWR rule) to find a feasible cyclic schedule. Let  $C(0)$  be the resulting schedule length. Output the schedule created for  $P(0)$ . If  $C(0) = C^*$ , then stop. Otherwise, let  $k = 0$ .

Step 2. Determine the earliest and latest start times of each task for the current set of  $n(k)$  products (assuming that all tasks are available at time 0 and must be done by time  $C(k)$ ). For each task  $i = 1, \dots, e$ , perform Step 2a.

Step 2a. If the earliest start time of task  $i$  equals the latest start time of task  $i$ , then task  $i$  is on a critical path for the current solution, and go to Step 2b. Otherwise, let  $L(i) = M$  (some large number greater than  $C(0)$ ) and skip Step 2b.

Step 2b. Create a new product, numbered  $n(k)+1$ . Let  $b$  be the product that requires task  $i$ . Remove from  $S(b)$  the subsequence that starts with task  $i$  and ends with the last task in  $S(b)$ . Let this removed subsequence be  $S(n(k)+1)$ , the sequence of tasks for the new product. Call the modified instance  $I(i)$ . Use the shop scheduling algorithm to find a feasible cyclic schedule. Let  $L(i)$  be the resulting schedule length.

Step 3. Select instance  $I(i)$  with the minimum  $L(i)$  (breaking ties randomly). Let  $P(k+1) = I(i)$ ,  $n(k+1) = n(k)+1$ ,  $C(k+1) = L(i)$ . Output the schedule created for  $P(k+1)$ . If  $C(k+1) = C^*$ , then stop. Otherwise, let  $k = k+1$  and return to Step 2.

Recall that Theorem 3 shows that if the number of products equals the number of tasks, the shop scheduling algorithm will yield a schedule with cycle length equal to  $C^*$ . Thus, Step 2 is entered at most  $e - n$  times. Each execution of Step 2 requires at most  $e - n$  runs of the shop scheduling algorithm. The computational effort of the shop scheduling algorithm is  $O(ne)$ . Thus, the total effort is  $O((e-n)^2 ne)$ .

#### 5. Algorithm Performance

Experimental testing evaluated the tradeoff exploration search on sixteen sets of problem instances. The primary goal of these experiments was to determine the average quality of the schedules that the search constructed. In this case, quality is determined by the throughput and WIP of the schedule. Since the computational complexity of the algorithm is known, testing did not consider the objective of computational effort.

The systematic generation of problem instances considered the following attributes: the number of products  $n$ , the total number of tasks per product  $e$ , number of resources  $R$ , the variation in processing times between tasks on the same resource, and the variation of means among each resource. There were four cases (which corresponded to the values for  $n$ ,  $e$ , and  $R$ ) and, within each case, four variations (which corresponded to different levels of variability). Altogether, we generated 16 problem sets, each with 10 instances. Represented as triples  $(n, e, R)$ , the four cases

are (5, 25, 5), (25, 125, 5), (5, 125, 25), and (25, 125, 25). Details of the problem instances can be found in Herrmann, Chauvet, and Proth [2].

The tradeoff exploration search was used to find a sequence of solutions to each instance. For each instance,  $C^*$ ,  $TH^*$ , and  $W^*$  were calculated, and the throughput and WIP of each cyclic schedule was normalized by dividing it by the  $TH^*$  and  $W^*$  for that instance. The presentation of the results below uses these normalized measures, which are called *relative WIP* and *relative throughput*.

The first result was that the variability of processing times did not affect the performance greatly. Therefore, all reported results are averaged over the forty instances in all four variations of each case. Figure 2 shows the results of the tradeoff exploration search for each case. (Note that the curves for cases (5, 25, 5) and (25, 125, 25) overlap.) The search's initial solution is the schedule generated by the shop scheduling algorithm (using the MWR rule). Thus, the performance of the search is influenced greatly by that heuristic's performance. In all cases, the search found solutions with higher throughput and higher WIP. In cases (5, 25, 5) and (25, 125, 25), the average relative WIP of solutions with maximal throughput was 1.87 and 1.72. In case (25, 125, 5), the average relative WIP of solutions with maximal throughput was 4.62. In case (5, 125, 25), the average relative WIP of solutions with maximal throughput was 2.24.

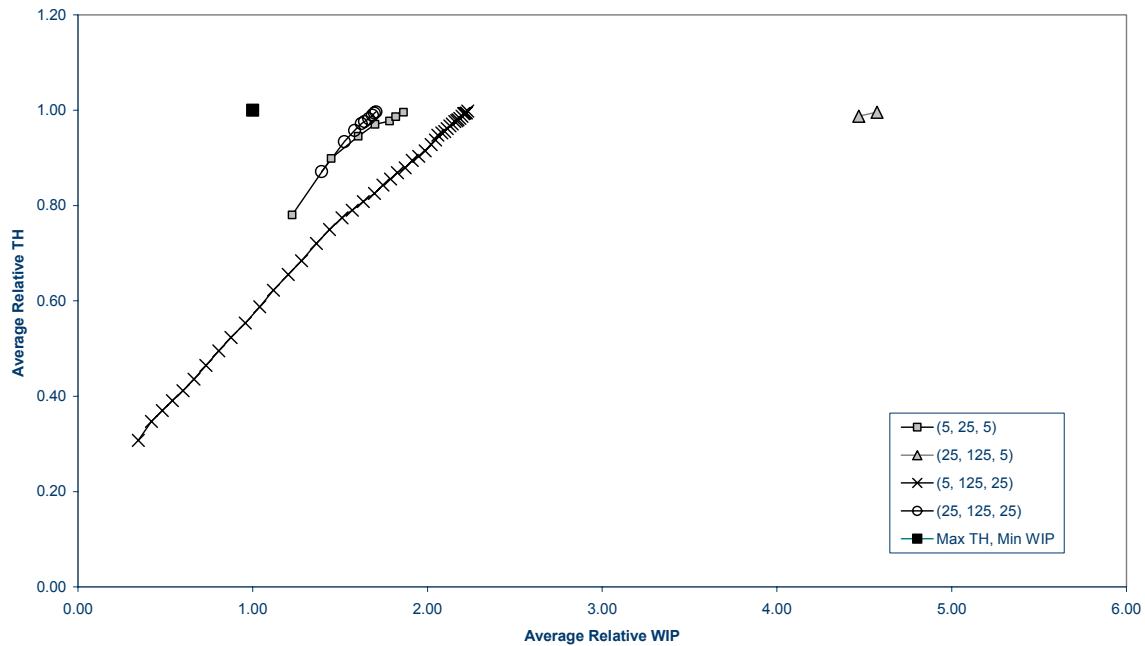


Figure 2. Performance of tradeoff exploration search

The results also show that, in some cases, the search converged to maximal throughput solutions quickly. In case (25, 125, 5), the search required just one iteration to reach the maximal throughput (averaged over 40 instances). In case (5, 25, 5), the search required six iterations; in case (25, 125, 25), the search required eight iterations. In case (5, 125, 25), however, the search started with very low relative throughput solutions, and the search required 43 iterations to reach the maximal throughput (averaged over 40 instances). In cases (5, 25, 5), (5, 125, 25), and (25, 125, 25), the search provides a series of solutions that allow a decision-maker to tradeoff the WIP and throughput objectives.

The structure of the cases helps explain the performance of the scheduling algorithms. In cases (5, 25, 5) and (25, 125, 25), the number of products equals the number of resources, so products generally don't have to wait much, and the resources stay busy. In case (25, 125, 5), the number of products was much larger than the number of resources, so products have to wait a great deal (increasing WIP). In case (5, 125, 25), the number of products is small, but each product has many tasks. Thus, resources are often idle, and it is hard to achieve the maximal

throughput, though the WIP is low. The tradeoff exploration search performed well in case (5, 125, 25) because it created new instances by dividing the long sequences (of 25 tasks per product) into shorter sequences. But in case (25, 125, 5), the sequences were already short so the search could not improve the solution quality greatly.

## 6. Summary and Conclusions

This paper presented the deterministic cyclic scheduling problem and discussed the tradeoffs between two important performance measures: WIP and throughput. The tradeoff exploration search visits a series of points in the problem space to generate a series of schedules that converge to a solution with maximal throughput. The tradeoff exploration search provides a series of solutions that allow a decision-maker to tradeoff the WIP and throughput objectives. An experimental study provided results for evaluating the quality of the solutions generated. The study used 160 randomly generated problem instances that ranged in problem size and in the amount of processing time variability. Processing time variability had less influence on solution quality than problem size and the relative number of products, tasks, and resources.

The cyclic scheduling problem is an interesting scheduling problem because the solution (which is continually repeated) significantly affects the performance of a dynamic production system. However, it is not clear which schedule is “best,” since there exist multiple objectives. This paper seeks to explain the behavior of this problem and provide decision-makers insight and approaches that will help manage such systems effectively. However, more work remains to extend the analysis to other domains.

## Acknowledgements

Dr. Herrmann has a joint appointment with the Institute for Systems Research. The research described in this paper was conducted using the facilities of the Computer Integrated Manufacturing Laboratory at the University of Maryland. The author appreciates the help of Jean-Marie Proth and Fabrice Chauvet on work related to that described here.

## References

1. Chauvet, Fabrice, Jeffrey W. Herrmann, and Jean-Marie Proth, 2003, “Optimization of cyclic production systems: a heuristic approach,” *IEEE Transactions on Robotics and Automation*, 19(1).
2. Herrmann, Jeffrey W., Fabrice Chauvet, and Jean-Marie Proth, 2003, “Tradeoffs During Scheduling of Cyclic Production Systems,” <<http://www.isr.umd.edu/~jwh2/papers/cyclic/cover.html>> (22 January 2003)
3. Herrmann, Jeffrey W., and Chung-Yee Lee, 1995, “Solving a class scheduling problem with a genetic algorithm,” *ORSA Journal on Computing*, 7(4), 443-452.
4. Kamoun, H., and C. Siskandarajah, 1993, “The complexity of scheduling jobs in repetitive manufacturing systems,” *European Journal of Operational Research*, 70(3), 350-364.
5. Matsuo, Hirofumi, 1990, “Cyclic scheduling problems in the two-machine permutation flow shop: complexity, worst-case, and average-case analysis,” *Naval Research Logistics*, 37, 679-694.
6. Roundy, Robin, 1992, “Cyclic schedules for job shops with identical jobs,” *Mathematics of Operations Research*, 17(4), 842-865.
7. Storer, R.H., S.Y.D. Wu, and R. Vaccari, 1992, “New search spaces for sequencing problems with application to job shop scheduling,” *Management Science*, 38(10), 1495-1509.
8. Vieira, Guilherme E., Jeffrey W. Herrmann, and Edward Lin, 2003, “Rescheduling manufacturing systems: a framework of strategies, policies, and methods,” *Journal of Scheduling*, 6(1), 35-58.