PRODUCTION AND OPERATIONS MANAGEMENT
Vol. 3, No. 4, Fall 1994
Printed in U S A.

AN OPTIMIZATION-BASED METHODOLOGY FOR RELEASE SCHEDULING*

MAGED M. DESSOUKY AND ROBERT C. LEACHMAN
Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA 90089-0193, USA
Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA 94720, USA

Operational-level scheduling decisions on the factory floor include the release policy and the dispatching rule. Recent work by researchers suggests that factory performance depends heavily on the release policy. Several heuristic factory release rules have been developed that generate release decisions based on the status of the bottleneck machine. The heuristic rules perform very well in reducing the work-in-process inventory. However, factory output schedules generated by these rules generally fail to match the demand pattern, creating excessive finished goods inventory and/or excessive backorders. To overcome these difficulties, a computationally efficient integer programming model is proposed for release scheduling.
(SCHEDULING; RELEASE POLICIES; RESTRICTED START MODELS)

1. Introduction

The overall operational level scheduling problem may be decomposed into decisions for the release of new work and operations dispatching decisions. This paper focuses on the former, release scheduling. A computationally practical integer programming model is proposed for release scheduling, with which simple heuristics such as first-in-first-out (FIFO) can be integrated for dispatching decisions. In computational experiments the proposed model is compared to release scheduling based purely on heuristics and to release scheduling based in part on an approximate linear programming model.

Following Holt et al. (1960) we distinguish between three levels of managerial decisions: strategic, tactical, and operational. Strategic decisions are concerned with selecting the product line, factory location, and equipment investment. At the tactical level of planning, the decision maker determines the factory operating hours, demand levels, and the monthly or weekly schedules. Operational decisions, the focus of this paper, include release and dispatching decisions. Release rules determine when to release a new lot of raw material into the production line. Dispatching rules select

* Received December 1993; revision received June 1994, accepted September 1994.

1059-1478/94/0304/276$1.25

which product(s) to manufacture next at a machine when it becomes idle. In support of release decisions, an optimization model is proposed that is to be resolved periodically or whenever a major change in factory status occurs, such as a major machine failure. In this manner the production starts schedule is updated to reflect the current factory status and the current factory demands.

2. Survey of Models for Operational Level Scheduling

Our focus will be on models for classical job shop or flow shop environments characterized by jobs or production lots that must pass through a long series of processing operations involving different kinds of equipment and labor. Changeovers of equipment to perform different operations are assumed to be negligible or lot-specific (not operation-specific), as is the case in many types of electronic and pharmaceutical factories and other high-technology manufacturing. Our particular motivation comes from high-volume semiconductor manufacturing.

Standard integer programming formulations for the job shop scheduling problem (Manne 1960; Pritsker et al. 1969) have been shown to be NP-Complete (Garey, Johnson, and Sethi 1976). The problem size and its computational requirements grow exponentially with the number of machines and the number of units to be processed, rendering these formulations impractical for real-world application. For this reason, numerous heuristic procedures have been developed to determine the release and dispatching rules for operational level decisions.

Dispatching rules for selecting the job to perform next when a machine becomes idle depend on the desired objective. Some of the objectives are to minimize the average flow time, the average completion times, the makespan, the average lateness, and the average tardiness. Graves (1981) surveyed various heuristics for dispatching, which address the different objectives. These rules include shortest processing time, earliest due date, least slack, and number of remaining operations.

Glassey and Resende (1988) and Wein (1988) show experimentally that factory performance in high-volume wafer fabrication processes is more dependent on the release rule than the dispatching rule. These authors argue that the implementation of release rules is relatively simple and practical.

Bechte (1988) argues that changing the priorities at the queues in front of machines does not change their capacity and therefore can neither accelerate nor delay the manufacturing flow as a whole. Bechte proposes a load-oriented approach that attempts to minimize the work-in-process (WIP) inventory and minimize the difference between planned and real lead times. The load limit reflects the capacity of the workstation during the planning period plus a buffer to ensure high capacity utilization. The load limit can be calculated directly from the planned lead times and the planning horizon. Once the load limits for each workstation are established, the workload at each workstation is calculated taking into account work already present at the workstation and future workload. The future workload at a workstation is based on work upstream that will be visiting the workstation within the planning horizon. Upstream workload is reduced by a discounting factor expressing the probability that an arbitrary work-order resident at an upstream workstation will visit the workstation within the planning horizon. Hence, the further away the workload and the more congested the intermediate workstation, the more the workload is discounted for downstream workstations. The release policy is to release a job if the

expected load based on the discounting factor at each workstation the job will visit is less than the workstation's load limit.

A continuous-review release policy applicable to high-volume wafer fabrication where the main source of variability is machine failure is proposed by Glassey and Resende (1988). This policy is termed "starvation avoidance." Under this policy the total load, expressed in machine processing time units, is computed by summing all the work at the bottleneck workstation plus work expected to arrive at the bottleneck workstation within a replenishment lead time. The lead time is the estimated time that a newly released job takes to arrive at the bottleneck workstation for the first time. If the total load in front of the bottleneck workstation is less than a prespecified level, a new job is released. In this manner the bottleneck workstation does not starve because the workload from the current work-in-process inventory reaching it within the lead time is not less than its available capacity.

Another release policy that takes into account the bottleneck workstation in wafer fabrication scheduling is workload regulating (Wein 1988). This policy is that a new job is released whenever the total work (measured in machine time units) in the system that will visit the bottleneck workstation drops below a prespecified level. The total work includes jobs waiting in front of the bottleneck machine and upstream jobs that will be visiting the bottleneck. The objective of the workload-regulating release policy is to minimize work-in-process inventory. The release policy is derived by approximating the scheduling problem with a dynamic control problem involving Brownian motion (Harrison 1988) that includes only the most heavily used workstations. Wein compares the workload-regulating release policy to a schedule that has constant and exponential interarrival times. He also tested a policy referred to as *fixed work-in-process* that keeps the WIP inventory in the system fixed. The simulation results performed by Wein showed that the workload-regulating release policy created a production starts schedule providing the lowest WIP inventory levels.

Other recent work in the area of job shop scheduling include heuristics developed by Barker and McMahon (1985), Adams, Balas, and Zawack (1988), Carlier and Pinson (1989), and Applegate and Cook (1991). The primary objective of these heuristic rules is to minimize WIP inventory levels or the makespan, neglecting customer demand patterns. Notwithstanding the fact that minimizing WIP inventory is a worthy objective, it is imperative to develop a schedule that can meet demand in a timely manner so that corporate planners may properly commit to customer shipment dates.

Roundy, Herer, and Tayur (1989) develop a two-module scheduling system to perform job-shop scheduling with the objective of meeting the due date. The two modules are planning and dispatching. The planning module, an integer programming model that minimizes the total (weighted) tardiness cost, determines the priorities for the dispatching module. The constraints of the optimization model are job precedence and capacity. The dual variables from the capacity constraints, which are the shadow prices of the integer programming model, are then used to set the dispatching priority rules for the job shop. The authors compared this scheduling system to several simple heuristics that do shop floor control. In general the two-module approach outperformed the other heuristics in terms of the tardiness cost. The procedure did very well when the due dates were loose and/or scattered. The disadvantage of this approach is a high computational cost for the integer programming formulation

used in the planning level. Another priced-based shop-floor scheduling system is described by Morton et al. (1988).

3. Optimization Models Used to Determine the Release Policy

Although the use of optimization models for operational scheduling has largely been unsuccessful, we believe that appropriately formulated models may be successfully applied to release scheduling. Two approaches in this regard are investigated here. First, a linear programming model incorporating time lags to approximately account for processing times is formulated. Although such a model is a somewhat crude approximation, it is computationally practical to solve. Next, a new integer programming model of operational scheduling that precisely accounts for processing times, but restricts the domain of allowed schedules, is formulated. As will be shown, this formulation is such that near-optimal integer solutions are computationally practical to obtain.

This section only presents the constraints of the formulations because the objective function can vary depending on the goal of the manufacturing manager. An example objective function is described in Section 4.

The optimization models are based on the following assumptions: (1) Products are processed in lots, with each lot type belonging to a particular product that has its own sequence of operations. (2) Each operation is performed on one lot of a product at a time and is fully processed by one unit of a resource of a certain type. In addition, the resource unit processes only one lot at a time. (3) Each resource type consists of a bank of identical parallel servers (i.e., machines, operators, etc.) that can be allocated among the operations. (4) The lot setup time for any operation is negligible or is sequence-independent so that it can be included in the lot processing time for the operation. (5) The lot processing time of each operation i, p_i, is deterministic. (6) All operations requiring the same resource have the same processing time. Even though the above assumptions may not be applicable to some industries, they are appropriate for the modeling of the manufacture of high-technology products such as semiconductors.

A flow chart of a sample route in the semiconductor wafer fabrication process is shown in Figure 1. The number above the arrow is the operation number. The boxes indicate the machine type performing the operation. There are typically several identical machines in the factory that can perform the operation. The number in the box is the processing time at the machine. A wafer may be processed by a particular machine type more than once in its route. Hence, an appropriately formulated op-

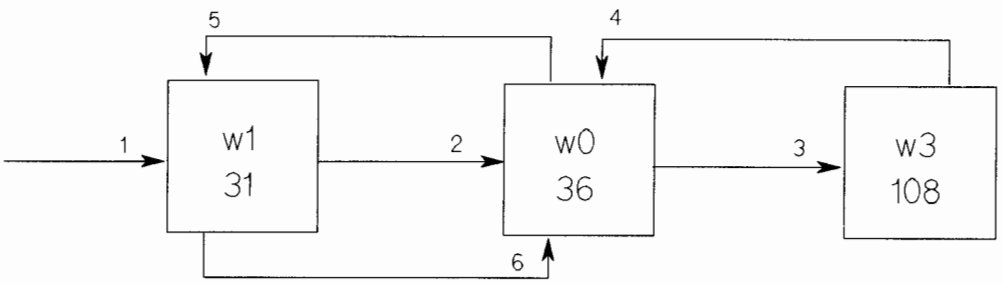

FIGURE 1. Flow Chart of Route.

timization model for the wafer fabrication process needs to account for multiple identical machines and the ability to visit a machine type more than once in its route. Before the models are presented, notation that is used is defined.

3.1. *Notation*

The notation for parameters of the models is as follows:

1. Indices.

t = Continuous time index, $t \leq T$, where T is the time horizon.

i, j = Operation index, $i, j = 1, \ldots, N$, where N is the number of operations. Each finished good requires some sequence of these operations to be completed. We also use i to index intermediate products (i.e., production lots that have completed operation i but have not yet begun follow-on operations).

k = Resource type index, $k = 1, \ldots, K$, where K is the number of resources. Typically, each index k represents a workstation or machine type that can perform several different operations.

2. Decision variables.

$z_i(t)$ = The work performed per unit time at time t at operation i, measured in terms of lot units of completed product quantity per unit time.

$s_i(t)$ = Number of production lot starts at operation i at time t, expressed in units of operation i output.

$f_i(t)$ = Number of units of intermediate product finished by operation i at time t (i.e., the number of production outs at time t).

$I_i(t)$ = Inventory of completed intermediate product i at time t (i.e., the number of lots that have completed operation i but have not yet started any successive operations).

3. Parameters.

$d_i(t)$ = External demand for product i at time t (only certain values of i correspond to finished goods).

$\bar{a}_{ij}$ = Number of units produced by operation i required as input per unit start of operation j.

$c_k(t)$ = Number of available units of resource k at time t.

p_i = Processing time of operation i. We assume lot-operations are uninterruptable.

An upper case of a decision variable or parameter is the notation used herein to refer to the cumulative flow. In addition, let x^+ denote the smallest integer not smaller than x, and let x^- be the greatest integer not greater than x.

3.2. *A Linear Programming Formulation with Processing Time Modeled as a Transfer Lag*

Hackman and Leachman (1989) derive from their framework the traditional linear programming model for production planning. They show that the linear programming model assumes instantaneous processing time and that all production rates are constant during prespecified time intervals. That is, all allocation, production, transfer, capacity, and demand flows are assumed to be constant in each planning period. Each planning period is assumed to be unit-length ($t = 1, 2, \ldots, T$). Let P denote the set of time epochs marking the end points of the planning periods. Thus, the work and capacity rates during time period t are denoted by $z_{i,t}$ and $c_{k,t}$, respectively. The interval $(t - 1, t]$ is referred to as period t. The cumulative flow of the rate-

based variables (i.e., $z_{i,t}$) in a period-based model can be expressed as summations (i.e., $Z_i(t) = \sum_{\tau=1}^{t} z_{i,\tau}$).

The authors also develop an alternative linear programming model that relaxes the instantaneous transfer-time assumption. A transfer lag is associated with each operation. The authors divide the transfer time into a lag after operation and a lag before operation. The lag after operation is any period of time between the time a product finishes processing and the time a product is released to inventory. Examples of this time delay are dry time after painting and cooling time after heating.

The lag before operation is any period of time between the time a product is withdrawn from inventory to the time a product starts processing. Examples of this time delay are travel time between operations and inspection time before an operation. A lag before operation is defined for each successor operation j.

As proposed by Petrakian (1991), an approximate method for incorporating the processing time delay into the linear programming model is to set the lag after operation to be the processing time and the lag before operation to zero. This model is referred to as Model LAGS. Figure 2 shows the cumulative curve of production starts, work, and output when the lag after operation is set to the processing time. The cumulative output curve increases at a linear rate within each planning period because production starts within a time period are applied uniformly. The output curve is shifted to the right by the processing time p_i. For operation i, the work at time t is released to inventory at time $t + p_i$. The production starts curve for operation i is typically shifted to the left by the lag before operation from operation j to operation i. Because the lag before operation is set to zero, cumulative production starts equal the cumulative work at each instant of time.

The resulting linear programming model contains three types of constraints. The first constraint type is referred to as material balance, applicable to each operation i. This constraint restricts the cumulative production starts of each intermediate product plus any external demand to be less than or equal to the cumulative amount produced plus initial inventory. The slack of this constraint is the inventory at time t of product i. The second set of constraints restrict the total work of each resource to be less than or equal to the available capacity. The third set of constraints restrict the domain of the decision variables.

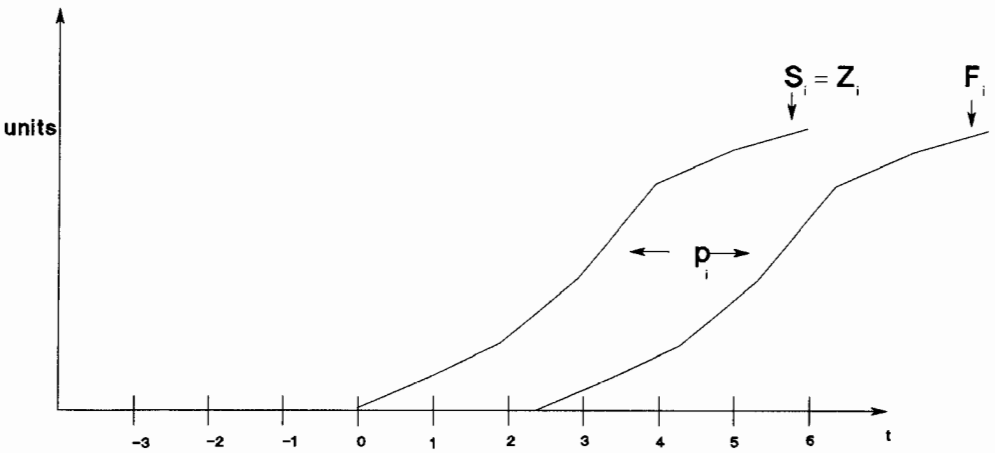

FIGURE 2. Model LAGS.

The material-balance constraint at operation i needs to be enforced only at points in time when the rates in the model change. If the processing times are noninteger, then enforcing the material-balance constraint only at the integer time points does not ensure feasibility. The material-balance constraints need to be enforced at time points when $t - p_i$, or t itself is integer. This set of time points is referred to as GI_i.

$$GI_i = \begin{cases} p_i, 1 + p_i, 2 + p_i, 3 + p_i, \ldots & \text{flow into inventory} \\ 0, 1, 2, 3, \ldots & \text{intermediate product transfer and external demand} \end{cases}$$

3.3. *An Integer Programming Formulation Explicitly Accounting for Processing Time*

When the assumption of instantaneous processing times is relaxed, the scheduling problem can be precisely formulated only as an integer program. In terms of the framework, $s_i(t)$ and $f_i(t)$ are event-based flows, whereas $z_{i,t}$ is still a rate-based flow during the time interval indexed by t. This section formulates a model that explicitly accounts for the processing time but restricts the domain for production starts of an operation to be the grid of integer multiples of the processing time. This model is referred to as Model PSTPT. As before, the planning periods are assumed to be unit-length. The demand events and capacity changes in the workstation can only occur at the integer time points. In Model PSTPT each operation has its own time grid representing the points in time when the operation workload can change (i.e., when operation starts can occur). As general notation, let W_i denote the set of time epochs marking the end of a time period for operation i. More specifically, the workload of operation i is required to be constant during each time interval $(t - p_i, t]$, where $t \in W_i$.

Because the processing time is nontrivial, a count of the number of starts at an operation by time t needs to be made because the cumulative workload, $Z_i(t)$, may not equal the cumulative number of finished products, $F_i(t)$, at the integer time epochs. For example, if $p_i = 2$, there are two servers, and each server started processing at time 0, then $Z_i(1) = 1$ yet $F_i(1) = 0$. In this example, if no more starts are made at time 1, then at time 2 the cumulative workload would equal the cumulative number of finished products.

Figure 3 shows a sample plot of the cumulative number of finished units of a product over time for Model PSTPT. The cumulative number of finished units of a product is a step function with the jumps occurring at the time grid points. By setting the time grid points to be integer multiples of the processing time, the workload in each period is still constant, and the cumulative workload equals the cumulative output at the time grid points ($S_i(t - p_i) = Z_i(t) = F_i(t)$, where $t \in W_i$). The entire model can be expressed in terms of the production start variables.

Define the set GC_k as the set of operations i visiting workstation k (i.e., resource type k). The integer programming formulation of Model PSTPT for the special case that all operations visiting the same workstation have the same processing time is as follows.

3.3.1. MATERIAL BALANCE.

$$I_i(t) = I_i(0) + \sum_{\tau=1}^{(t/p_i)^-} s_i(\tau^* p_i - p_i) - \sum_{j=1}^{N} \sum_{\tau=0}^{(t/p_j)^-} \bar{a}_{ij} s_j(\tau^* p_j) - D_i(t) \geq 0,$$

all $t \in GI_i$, all i.

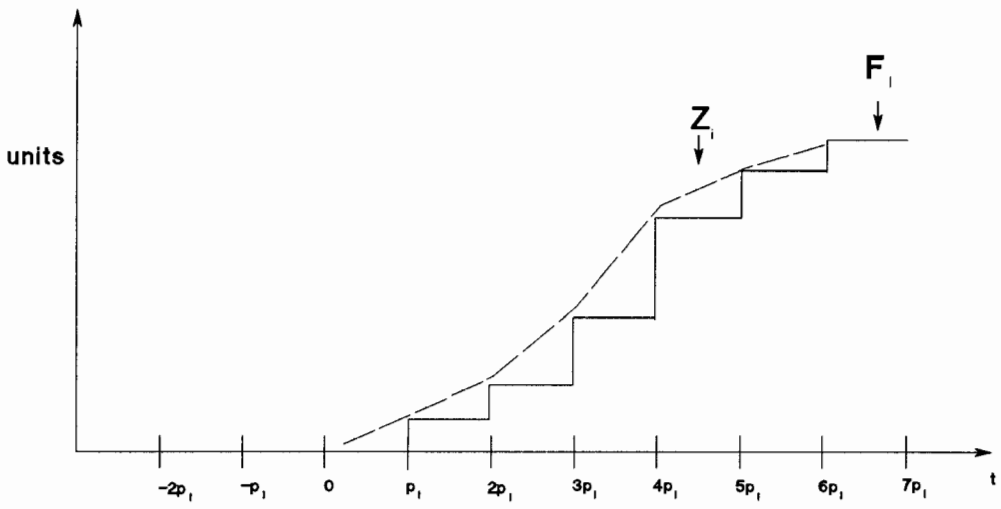

FIGURE 3. Cumulative Outs and Work Curves for Model PSTPT.

3.3.2. CAPACITY.

$$\sum_{i \in GC_k}^{N} s_i(t - p_i) \leq c_{k,t^+}, \qquad \text{all } t \in W_i \text{ where } i \in GC_k, \text{ all } k.$$

3.3.3. DOMAIN.

$$s_i(t) \geq 0 \text{ and integer}, \quad \text{all } t \in W_i, \text{ all } i.$$

To determine the inventory level of product i throughout continuous time, the material-balance constraint for product i only needs to be evaluated at points in time where inventory is withdrawn or added. When noninteger processing times are included in the model, the workload is not constant across unit intervals. However, the points in time when production operations can start are restricted so the points in time where the inventory at operation i can possibly increase or decrease can be identified. This set of time points is referred to as GI_i.

$$GI_i = \begin{cases} \text{all } t \in W_i & \text{flows into inventory of product } i \\ \text{all } t \in \bigcup_{\{j \mid \bar{a}_{ij} \neq 0\}} W_j & \text{intermediate product input by follow-on operations} \\ \text{all } t \in P & \text{external demand for product } i \end{cases}$$

Assuming external demand flows are event-based flows occurring at the integer time points, the external demand term can be expressed as

$$D_i(t) = \sum_{\tau=1}^{t^-} d_{i,\tau}.$$

The available capacity at time t is c_{k,t^+}, assuming that capacities change only at the planning period time grid points. In general, the capacity constraint for resource type k needs to be enforced for each subinterval between the points in time in which the rates $z_{i,\tau}$ for all operations i that require resource type k can change. Because all operations requiring the same resource set have the same processing time, the capacity

TABLE 1
Machine Data

Machine	Capacity	Time Between Failure (min)	Repair Time (min)
w0	4	900	100
w1	3	700	100
w2	1	1500	100
w3	4	1350	150

constraint needs to be enforced for the intervals corresponding to the time grid, W_i, which pertains to any operation i requiring resource type k.

4. A Methodology for Implementing Optimization Models in Operational Level Planning

A production network consisting of 12 operations, four machine types, and two end products (items) is developed to evaluate the models. The planning horizon is five days, and each day is assumed to be 8 hours. Production of each item involves a sequence of six operations with no assembly. Machine types w0 and w2 are very heavily utilized with machine utilizations of over 90% if all demand is met. The machine and route data are described in Tables 1 and 2, respectively. This data set is very similar to one used by Glassey (1990) and Petrakian (1991). We have chosen to model such a configuration because this type of route structure is very common in the semiconductor industry.

The demand for each item is assumed to be stationary. A unit of demand for item type 1 occurs at every planning period time grid point. A unit of demand for item type 2 occurs at every other planning period time grid point because the total demand of item type 2 is half of the total demand for item type 1.

Consistent with the previous studies, it is assumed that the set-up and move times are zero. Also, the buffer in front of each machine is assumed to be infinite so that no blocking of downstream machines occurs. The breakdown of machines is included

TABLE 2
Route Data

Item	Operation	Machine	Processing Time (min)
i1	op1	w1	31.0
i1	op2	w0	36.0
i1	op3	w3	108.0
i1	op4	w0	36.0
i1	op5	w1	31.0
i1	op6	w0	36.0
i2	op7	w2	42.0
i2	op8	w0	36.0
i2	op9	w1	31.0
i2	op10	w0	36.0
i2	op11	w2	42.0
i2	op12	w0	36.0

in the models with the repair and failure distributions assumed to be exponential. Finally, the yield loss is assumed to be zero ($\bar{a}_{ij} = 1$).

The production-starts schedule from Model PSTPT is compared to a production-starts schedule generated by a linear programming model (Model LAGS). Model PSTPT is also compared to a constant release policy and to a workload-regulating release policy. Recent analyses (see Petrakian 1991 and Glassey 1990) have shown that the workload-regulating release policy outperforms other release strategies based on the average work-in-process inventory performance measure.

To keep the focus of the experiments on the release policy, all the models use the simple FIFO dispatching rule. Wolff (1989) shows that for the single-server queue FIFO is optimal (same mean delay but minimizes the delay variance) among all order of service rules in which the processing times of the products in the rearranged sequence are independent and identically distributed and independent of the arrival process. Thus, a simple dispatching rule (FIFO) that in general performs reasonably well is utilized.

The production starts variables of the first operation in each route given by Model PSTPT are used to determine the release schedule. Each production start variable in the optimization model indicates both the time to release the product and the quantity to release. The optimization model is run at periodic review times or whenever a major change in factory status occurs. To test this methodology, Model PSTPT is integrated into a simulation model. The simulation model calls the optimization model at periodic review times or when there is a major machine failure to determine the release policy up to a planning horizon. The optimization model takes into account the current state of the simulation to determine the release policy.

Figure 4 diagrams the interaction between the optimization model and the simulation model. The first step is to read the data. Then, the optimization model is initialized. This involves setting the current WIP inventory, finished-goods inventory, and backorders to zero, and setting the current machine availability to the machine capacities. The optimization model for Model PSTPT is an integer programming model using a branch-and-bound procedure with the solution to the linear programming relaxation being the start node in the search. The branch-and-bound procedure stops when the objective value of a feasible integer solution is within 5% of the optimal solution of the linear programming model relaxation. The branch-and-bound search is stopped at a close-to-optimal feasible solution instead of searching for an optimal solution, because the amount of computation required to find an optimal solution is considerably longer. We had little difficulty finding a feasible integer solution within its range of the lower bound on the optimum; typically the first integer solution found satisfied this criterion (Appendix A).

From the optimization model the release policy is determined and fed into the simulation model. The simulation model is run for the length of the review interval (2,400 minutes) or until a bottleneck machine breaks down. If the current simulation time is greater than the simulation end time, the average and standard deviation of the backorders, WIP inventory, and finished-goods inventory are printed. The simulation end time is 2,448,000 minutes. To get independent sample statistics (Law and Kelton 1991), the simulation run time is divided into ten blocks with a duration of 240,000 minutes. The initial transient warm-up period is 48,000 minutes.

Each time the optimization model is called, the current WIP inventory of the simulation is used as the initial inventory for the optimization program. The current

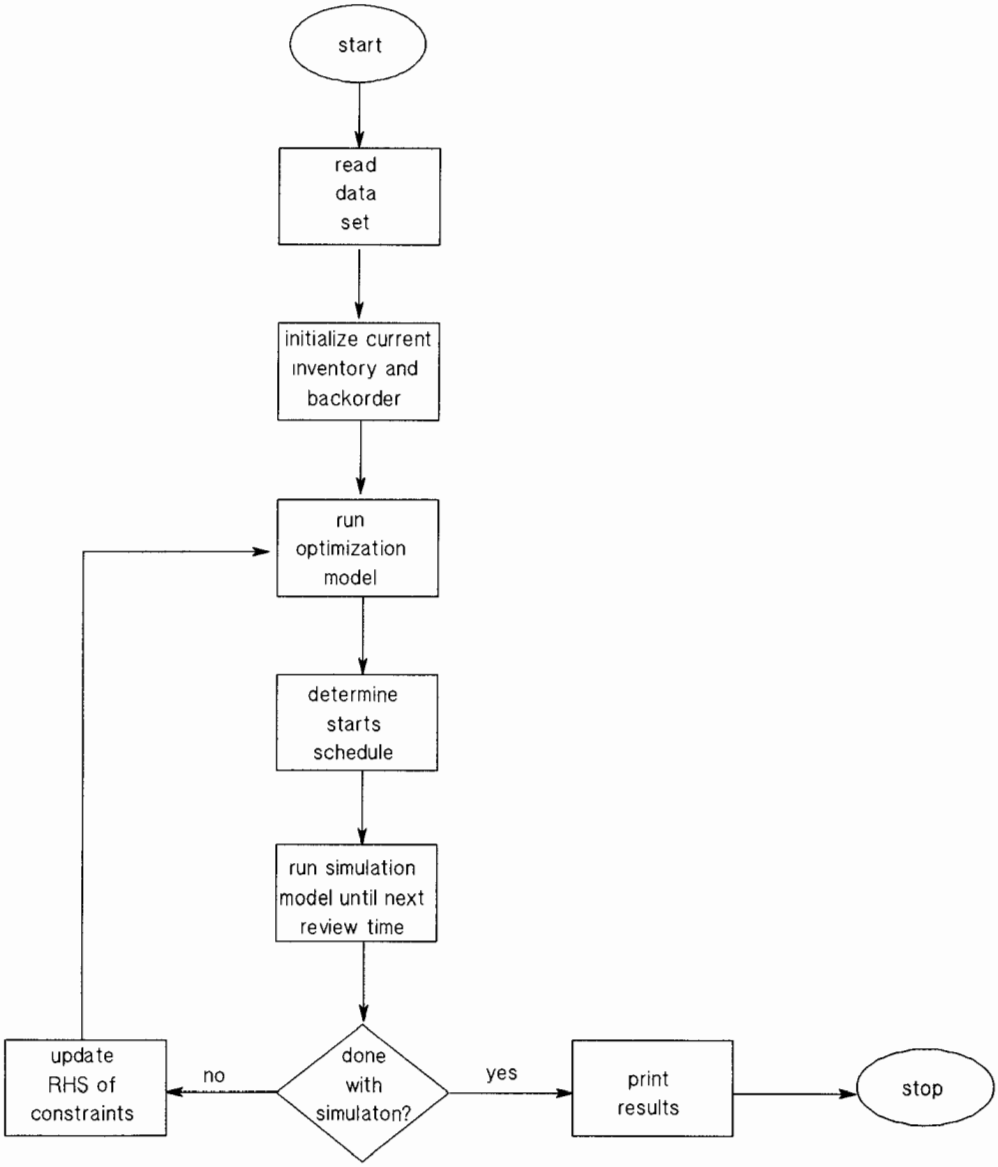

FIGURE 4. Linking Optimization Model with Simulation Model.

backorders and finished goods inventory of the simulation are used to adjust the demand in the optimization model. The optimization program also takes into account the machines that are currently in a state of failure. The current machine availability of the simulation program is used as the machine capacity in the optimization program. Updates to the machine capacity in the capacity constraints are made when the machine is repaired. Note that these changes require only updating the right-hand sides of the constraints in the optimization program.

The objective function of Model PSTPT is to minimize costs. The costs are inventory-holding cost and backorder-penalty cost. Production costs were not included in the objective function, because demand is considered to be fixed and must be met sometime before the end of the planning horizon. A backorder variable is added to

the material-balance equation of the last operation for each item type. The backorder variable is added to drive the start of production without having to force strict feasibility of meeting demand before the end of the planning horizon. The backorder cost is the same for all backorder variables. The inventory-holding cost is the same for all inventory variables. Thus, the WIP inventory cost is equal to the finished-goods-inventory holding cost. The backorder-to-inventory holding cost ratio in these experiments is 50:1, because the average backorder level is considered to be the primary performance measure. Our computational experience has shown that cost ratios greater than 50:1 yield about the same results.

The formulation for experiments is coded using the Optimization Subroutine Library (OSL) developed by IBM (1990) on an IBM Model 550 Workstation. The mixed-integer programming Subroutine EKKMSLV is used to solve Model PSTPT. The linear programming relaxation of the model is solved using Subroutine EKKSSLV, which uses the simplex method to solve the problem. Subroutine EKKMSLV solves the problem using a branch-and-bound method with the start node in the search being the solution to the linear programming relaxation.

The release policy based on Model PSTPT is compared to a constant release policy, a workload-regulating release policy, and a release policy based on a linear programming model. A constant release policy releases an item type 1 product at every planning period time grid point and releases an item type 2 product at every other planning period time grid point. This model is referred to as Model UNIFORM.

The workload-regulating release policy developed by Wein (1988) releases an item whenever the total work (measured in machine minutes) in front of the bottleneck machine is less than a predetermined threshold level. For this data set, machine type w0 is considered to be the bottleneck machine and is the one that the rule is based on. However, machine type w2 is very close to being a bottleneck machine. If more than one item type is being produced, the rule does not indicate which item type to release. Thus, we use a simple rule, furthest-behind-in-the-schedule, to select the item to release. That is, the item for which the difference between the cumulative demand and cumulative production starts at the first operation in the route is the greatest is released. It is seldom possible to select a threshold level to match exactly the desired output rate. Either the cumulative finished-goods inventory or backorders grows without bound if no adjustments to the rule are made. The threshold is set at a level that would generate an output rate greater than the desired rate but would restrict the maximum size of the finished-goods inventory to 10 units for each item type. Hence, if the work in front of the bottleneck machine is less than the threshold level and both items have a finished goods inventory of 10 units, no product will be released. This modification tended to stabilize the estimates for average backorders and finished-goods inventory for this model. This model is referred to as Model WORKREG.

The last model uses a linear programming model (Model LAGS) to determine the release policy. The solutions to this linear programming model are not schedules that specify the start and completion of production operations, because this formulation treats production as a continuous flow. Hence, the production-starts schedule from the optimal solution needs to be altered. The cumulative production starts schedule for initial operations is rounded down to the nearest integer. The scheduling methodology for Model LAGS is similar to the procedure implemented for Model

PSTPT, except the optimization model uses a linear programming model instead of an integer programming model.

The comparison criteria of the four models are the average backorders, average finished-goods inventory, and average WIP inventory. Each performance measure for each model is evaluated at different levels of bottleneck machine utilization. Again, the bottleneck machine is machine type w0. Table 3 shows the different bottleneck-machine utilizations that were run and the values of the parameters to support that utilization. For example, to support a bottleneck machine utilization of 93.8% for the current data set, the planning review interval and the planning period length are set to 2,400 and 48 minutes, respectively. To accommodate a rolling use of a finite horizon model, the optimization model used a planning horizon that included 10 extra planning periods. The optimization model used a planning horizon of 2,880 minutes, whereas the planning review interval of the simulation model is 2,400 minutes. Note that the planning horizon needs to be longer than the review interval by at least the length of the cycle time through the factory. The table also gives the workload-regulating threshold values to support the bottleneck machine utilization values. For a utilization of 93.8%, the threshold value is equal to 750 machine minutes.

A problem size of two end-items and 12 operations is selected so that extensive simulation testing could be performed. Because of the unreliability of the simulated machines (a realistic assumption in semiconductor wafer fabrication processes), the simulation run time has to be extremely long to get reliable estimates of the performance measures. To make any statistical inferences on the results, we had to simulate the model for 2,448,000 minutes (1,020 weeks). This simulation run time translates into an average CPU time of 6.67 minutes on an IBM 550 Workstation for Models UNIFORM and WORKREG and 4,007 minutes for Models LAGS and PSTPT. The optimization models require additional processing because of running OSL at the end of each week or when a bottleneck machine has a major failure. Each optimization-model run takes on average 2 CPU minutes (Appendix A), and an average of 2,000 optimization runs are performed per simulation run.

Our experimental analysis tests a problem set with a total demand of 75 lots and a total number of machines of 12, comparing favorably with tests performed by other researchers. However, the main difference is that our experiments include many jobs with similar routes and many machines of the same type to replicate a high-volume wafer fabrication process. The problem size in Model PSTPT is not a function of the demand size per product type and the number of machines per machine type, but a function of the number of operations per product type.

TABLE 3
Parameters for Operational Level Models

Bottleneck Machine Utilization	Planning Period	Planning-review Interval	Optimization Model Horizon	Workload-regulating Threshold (min)
83.3	54	2700	3240	550
90.0	50	2500	3000	670
93.8	48	2400	2880	750
95.7	47	2350	2820	920
97.8	46	2300	2760	1180

Finally, the assumption of steady-state demand is not made to reduce the computational time requirements but rather to facilitate the comparison of the models. Models WORKREG and UNIFORM do not take into consideration when demand is required, whereas the optimization-based models (PSTPT and LAGS) account for the demand when determining the release schedule. Hence, if the optimization-based models outperform the other models in terms of backorders in a steady-state demand situation, it is likely that the optimization models also will outperform the other models when the demand is variable.

5. Experimental Results and Analysis

Figures 5 and 6 show the average backorders and average finished-goods inventory at different levels of bottleneck-machine utilization for the four models. Model UNIFORM has the highest backorders at all levels of bottleneck machine utilization because the model does not account for when the demand is needed. As Figure 6 shows, Model UNIFORM has an average finished-goods inventory equal to zero. Thus, a constant-release policy for this data set cannot produce an output schedule that can keep up with the demand.

To perform pairwise statistical tests for the mean backorders and mean finished goods on the different models, the variances of the models have to be same. The standard deviation of the average backorders estimate is contained in Figure 7. As the diagram shows, Model WORKREG has a much larger variance of backorders than the other models, especially at high utilization levels. The Bartlett Procedure (Anderson and McLean 1974) to test the homogeneity of the variances concludes that the variances of the four models are different. However, when Model WORKREG is

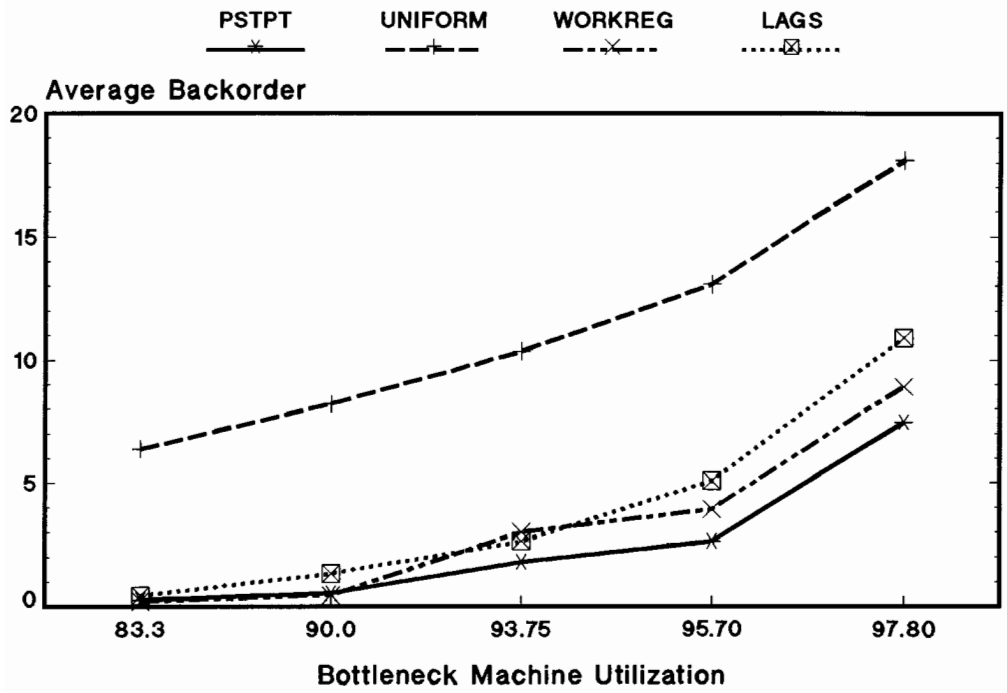

FIGURE 5. Average Backorder Results.

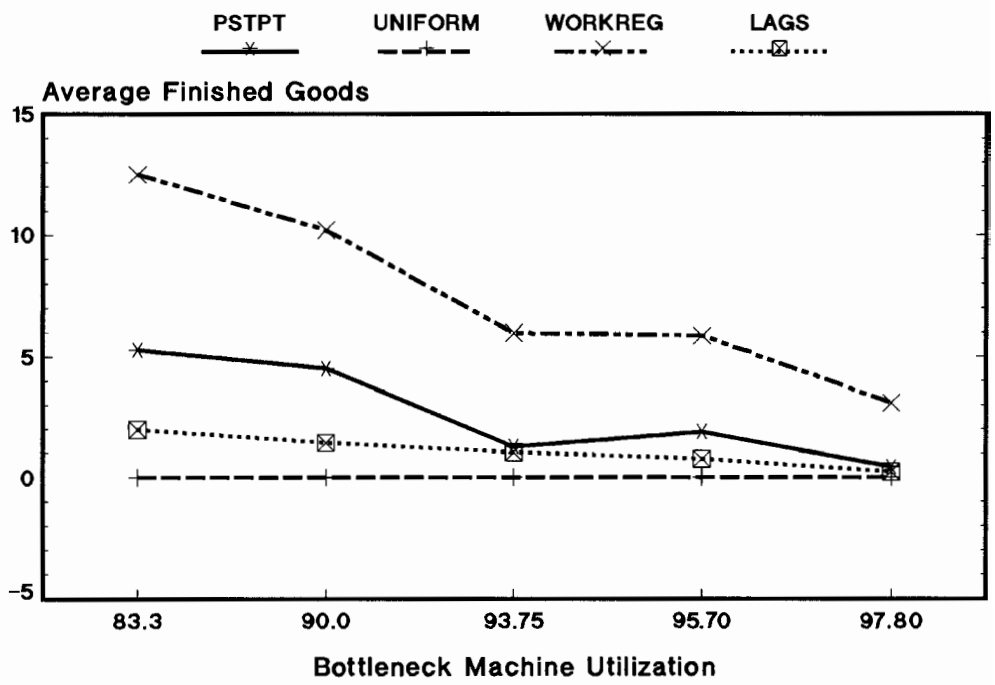

FIGURE 6. Average Finished-Goods Inventory Results.

excluded, the Bartlett test concludes that the variances of Models UNIFORM, PSTPT, and LAGS are not different at a confidence level of 99.9%. Hence, for the backorders performance measure, pairwise statistical tests using the Newman-Keuls test (Anderson and McLean 1974) on the mean backorders for Models UNIFORM, PSTPT, and LAGS is performed. At a confidence level of 95.0%, all pairwise statistical tests on the mean backorders for all levels of bottleneck-machine utilization are statistically significant. The ranking of the models in terms of backorders is Model PSTPT, LAGS, and UNIFORM.

Model PSTPT maintains lower backorder levels than Model LAGS because it is a more realistic model of production. Model PSTPT incorporates the processing time in the model formulation. Model LAGS uses a continuous-variable and period-based model to schedule a discrete-variable and continuous-time situation. Thus, there is a mismatch between Model LAGS and the real-world. Model PSTPT remedies the discrete-variable mismatch by using an integer programming model to schedule production starts. By using a special time grid at each operation, the integer programming model operates similarly to a continuous-time model when the bottleneck machine type is heavily utilized because at each possible production start time point in the model the bottleneck will be scheduled to begin processing. Model LAGS has lower finished-goods inventory levels than Model PSTPT, but it cannot meet the demand as consistently as Model PSTPT.

No statistical conclusions based on the mean backorders between Model PSTPT and Model WORKREG can be made because the variances of the mean backorders differ. Some general observations are that Model PSTPT has lower average backorders than Model WORKREG especially at the higher levels of bottleneck machine utilization. In addition, Model PSTPT has less finished goods inventory than Model WORKREG

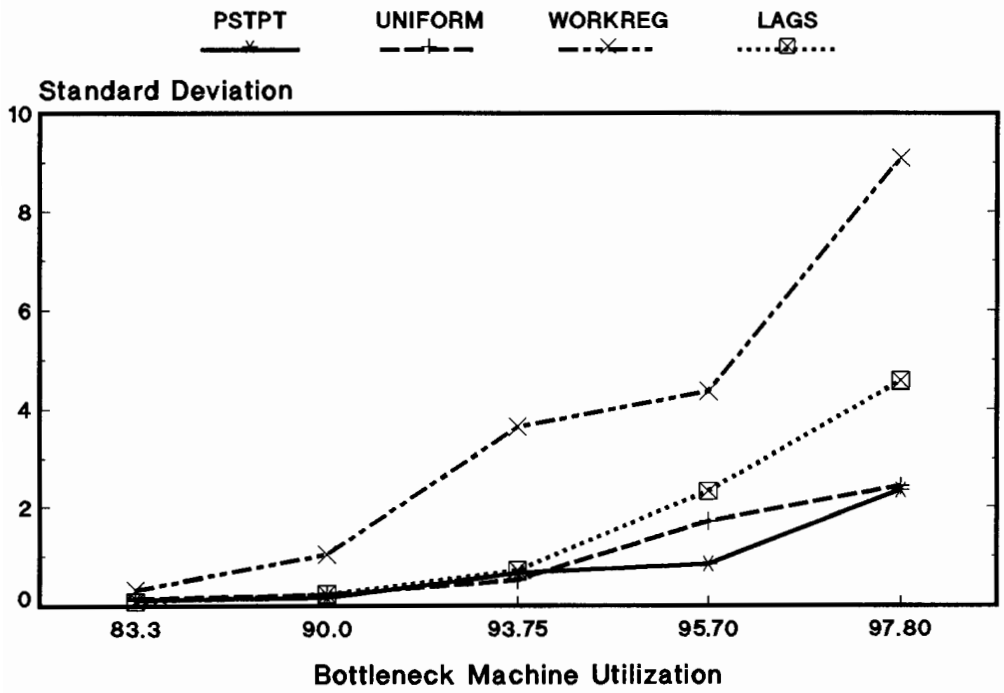

FIGURE 7. Standard Deviation of Backorder Results.

and a much smaller deviation between cumulative demand and cumulative output that causes the high variance of backorders in Model WORKREG. The workload-regulating release policy is designed for situations that clearly have only one bottleneck machine type and all items visiting the bottleneck machine type. Under these conditions a simple heuristic rule, such as furthest-behind-in-the-schedule, is sufficient to select which item to release. However, in cases where there are several close-to-bottleneck machine types and all items do not visit the close-to-bottleneck machine types, simple heuristic rules do not yield an output schedule that can meet the demand. More complicated rules that monitor the factory state are needed under such conditions. It is not the purpose of this paper to develop these rules because the heuristic rules tend not to be robust and are applicable only to a particular data set. The intention is to show that robust models, such as optimization models, can yield an output schedule that meets the demand.

For this data set machine type w2 is very close to being the bottleneck, and only item type 2 visits machine type w2. In some circumstances using the furthest-behind heuristic in conjunction with the workload-regulating release rule can release the wrong item type. If machine type w2 is down for long periods of time and item type 2 is far behind in its schedule, item type 2 products will be released resulting in a large queue in front of machine type w2. When machine type w2 breaks down for a long period of time, it would be better to release item type 1 products even though item type 2 is further behind in the schedule because the capacity of machine type w2 is one.

The average work-in-process inventory of the four models is shown in Figure 8. As the machine utilization increases, the average work-in-process inventory increases exponentially. Figure 8 also shows the results of the Newman-Keuls test for a 95.0%

Utilization	Model			
83.3	WORKREG	UNIFORM	LAGS	PSTPT
	0.944	1.155	1.220	1.915
90.0	WORKREG	UNIFORM	LAGS	PSTPT
	1.751	2.442	2.510	3.718
93.8	WORKREG	LAGS	UNIFORM	PSTPT
	2.634	3.761	4.261	4.560
95.7	WORKREG	LAGS	UNIFORM	PSTPT
	4.302	5.174	6.731	6.938
97.8	WORKREG	LAGS	PSTPT	UNIFORM
	7.661	8.940	10.26	11.77

FIGURE 8. Newman-Keuls Test on Mean WIP at $\alpha = 0.05$.

confidence level on the mean WIP for the four models. (In this case the Bartlett test concluded that the variance of all four models are not different at all levels of bottleneck machine utilization.) An underline between two models indicates that the difference between the mean WIP levels of the two models are statistically insignificant. For example, at a bottleneck utilization of 83.3% the mean WIP of Models UNIFORM and LAGS are not statistically different. Some general conclusions are:

1. Model WORKREG has the lowest average WIP inventory at all levels of machine utilization. However, it is statistically significant only at the lower levels of machine utilization. At the higher levels there is no statistical difference between Models WORKREG and LAGS.
2. Model WORKREG outperforms Model PSTPT in terms of mean WIP at all levels of machine utilization.
3. Model UNIFORM has lower mean WIP than Model PSTPT at lower levels of machine utilization.

These results are consistent with previous findings that state that in general a closed-loop release policy gives better results in terms of WIP inventory than an open-loop release policy such as Model PSTPT. However, Model PSTPT maintains lower total inventory levels (sum of finished goods and WIP inventory) than Model WORKREG at all levels of machine utilization.

6. Summary of Results and Future Direction of Research

Experimental results showed a production-starts schedule based on Model PSTPT yields an output schedule that closely matches the demand. Model PSTPT outperformed models based on workload regulating, constant release, and rate-based linear programming in terms of the average backorders and the total inventory performance measures. The backorders performance measure is especially important in very competitive industries such as the semiconductor industry, because it is imperative to develop a schedule that can provide on-time delivery. However, in terms of the WIP performance measure, Model WORKREG generated a production schedule yielding the smallest WIP. In addition, the computational requirements of workload regulating are significantly less than Model PSTPT.

Using a special time grid at each operation, the solution to the linear programming relaxation of model PSTPT will tend to be integral. The results from this model show that it closely approximates the overall behavior of a production system in which times of operations are unrestricted. It accomplishes this by ensuring that the bottleneck machines do not starve. The model then allocates idle time for the remaining nonbottleneck machines. Hence, the computational advantages of Model PSTPT justifies using a restricted production start model to represent an unrestricted production start system.

Even though our experiments were for a relatively small data set, with the speed of today's computers, much larger problems can be formulated and solved in a realistic amount of computation time. Leachman (1993) solves linear programming models on an IBM Model 550 Workstation consisting of more than 150,000 columns and 100,000 rows to determine weekly production schedules. For model PSTPT, the number of integer variables is approximately $\sum_{i=1}^{N} T/p_i$, the number of material balance constraints for a serial production network is approximately $2 * \sum_{i=1}^{N} T/p_i$, and the number of capacity constraints when all operations requiring the same resource set have the same processing time is $\sum_{k=1}^{K} T/p_i$ where $i \in GC_k$.

In terms of the average WIP inventory performance measure, workload regulating gave better results than Model PSTPT. This may reflect the fact that the optimization models do not capture uncertainty on the factory floor. Safety stock could be added to the optimization models to account for uncertainty but that could increase the WIP and/or finished goods inventories unnecessarily. An alternative method to represent uncertainty in the optimization models needs to be developed.

Workload regulating is a closed-loop release policy. Past results have shown that closed-loop release rules maintain lower WIP inventory levels than open-loop release rules. Using the production-starts schedule based on the start time of the first operation in each route given by the optimization model is an open-loop release rule. To make Model PSTPT operate in a manner similar to a closed-loop release rule, the production starts schedule proposed by Model PSTPT could be modified based on monitoring the total work through the bottleneck machine instead of strictly following the start times.

Finally, this research has focused on release scheduling. Future research can study the integration of Model PSTPT with dispatching rules more sophisticated than FIFO.

Appendix A

To test the difficulty of optimizing Model PSTPT and the efficiency of suboptimal integer solutions quickly obtained from the linear programming relaxation of the model, we developed the formulation

TABLE 4
Route Data

Item	Operation	Machine	Processing Time (min)
i1	op1	w1	35.0
i1	op2	w0	40.0
i1	op3	w3	120.0
i1	op4	w0	40.0
i1	op5	w1	35.0
i1	op6	w0	40.0
i2	op7	w2	45.0
i2	op8	w0	40.0
i2	op9	w1	35.0
i2	op10	w0	40.0
i2	op11	w2	45.0
i2	op12	w0	40.0

for a data set similar to the data contained in Tables 1 and 2. Because these experiments require running only the optimization model, the machine failure times are not included in the analysis. To have the same machine utilizations as in the previous experiments, the processing times in Table 2 are divided by the fraction of time a machine is available for processing. The new processing times are summarized in Table 4.

Table 5 shows the optimal value and the CPU time to obtain the optimal solution for Model PSTPT and for the linear programming relaxation for the sample data set. The ratio of backorder to inventory holding cost ranged from 500:1 to 1:1 in five experiments. The resulting integer program from this data set has 700 integer variables, 1,995 columns, 1,374 rows, and 68,426 nonzero matrix entries. As Table 5 shows, for a backorder-inventory holding cost ratio of 500:1, the linear program yields an all integer variable optimal solution. For the other cost ratios, the integer optimal and linear relaxed optimal solutions are very close. They differ by at most 1%. The CPU time to calculate the integer optimal solution can be considerably greater than that for the linear program. For example, with a cost ratio of 5:1, the integer program takes about 2,000 times as long to optimize as the linear program.

These results suggest that the computer run time to find an optimal solution is considerably greater for the integer program than for the linear program. However, a feasible solution to the integer program that is very close-to-optimal can be found in little more CPU time requirement than the time to solve the corresponding linear programming relaxation. A good feasible solution to the integer program can be found fast in terms of CPU time because the linear program's optimal solution has many basic feasible variables that are integer, and the gap between integer and linear optimal solutions is very small. The linear program solution tends to be integral because without the capacity constraints Model PSTPT can be formulated as a pure network model and if only one lot of each product is waiting and only one machine is free the result of a Simplex Pivot is integer. If multiple machines of the same type are free and multiple lots of each product type are waiting, the result of a simplex pivot might be fractional. On

TABLE 5
Results for Model PSTPT

	LP		IP (optimal)		IP (feasible)	
Cost Ratio	Objective Value	CPU Time (min)	Objective Value	CPU Time (min)	Objective Value	CPU Time (min)
500:1	797565.0	1.54	797565.0	1.54	797565.0	1.54
10:1	32955.0	1.53	32985.0	175.71	33030.0	2.61
5:1	24625.0	1.51	24690.0	2965.83	24825.0	2.93
1:1	15610.0	1.41	15645.0	80.53	15655.0	2.10

balance, the solution is close to integer. It takes considerably more CPU time to find the integer optimal solution because the cost versus production curve is very flat for Model PSTPT. In other words, there are many close-to-optimal feasible solutions.

References

ADAMS, J., E. BALAS, AND D. ZAWACK (1988), "The Shifting Bottleneck Procedure for Job Shop Scheduling," *Management Science*, 34, 391–401.

ANDERSON, V. L. AND R. A. MCLEAN (1974), *Design of Experiments*, Marcel Dekker, New York.

APPLEGATE, D. AND W. COOK (1991), "A Computational Study of the Job-Shop Problem," *ORSA Journal on Computing*, 3, 149–156.

BARKER, J. R. AND G. B. MCMAHON (1985), "Scheduling the General Job-Shop," *Management Science*, 31, 594–598.

BECHTE, W. (1988), "Theory and Practice of Load-oriented Manufacturing Control," *International Journal of Production Research*, 26, 3, 375–395.

CARLIER, J. AND E. PINSON (1989), "An Algorithm for Solving the Job-Shop Problem," *Management Science*, 35, 164–176.

GAREY, M. R., D. S. JOHNSON, AND R. SETHI (1976), "The Complexity of Flowshop and Job Shop Scheduling," *Mathematics of Operations Research*, 1, 2, 117–120.

GLASSEY, P. G. (1990), "A Comparison of Release Rules Using BLOCS/M Simulations," Research Report 90-15, Engineering Systems Research Center, University of California, Berkeley, Berkeley, CA.

——— and M. G. C. RESENDE (1988), "Closed-loop Job Release Control for VLSI Circuit Manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, 1, 1, 36–46.

GRAVES, S. C. (1981), "A Review of Production Scheduling," *Operations Research*, 29, 4, 646–675.

HACKMAN, S. T. AND R. C. LEACHMAN (1989), "A General Framework for Modeling Production," *Management Science*, 35, 4, 478–495.

HARRISON, J. M. (1988), "Brownian Models of Queueing Networks with Heterogeneous Customer Populations," in *Stochastic Control Theory and Applications*, W. H. Fleming and P. Lions (eds.), Springer-Verlag, NY, 147–186.

HOLT, C. C., F. MODIGLIANI, J. F. MUTH, AND H. A. SIMON (1960), *Planning Production, Inventories, and Work Force*, Prentice-Hall, Englewood Cliffs, NJ.

IBM (1990), "Optimization Subroutine Library," Kingston, NY.

LAW, A. M. AND W. D. KELTON (1991), *Simulation Modeling & Analysis*, McGraw-Hill, Princeton, NJ.

LEACHMAN, R. C. (1993), "Modeling Techniques for Automated Production Planning in the Semiconductor Industry," in *Optimization in Industry*, T. A. Ciriani and R. C. Leachman (eds.), John Wiley & Sons, NY, 1–30.

MANNE, A. S. (1960), "On the Job-Shop Scheduling Problem," *Operations Research*, 8, 2, 219–223.

MORTON, T. E., S. KEKRE, S. LAWRENCE AND S. RAJAGOPALAN (1988), "SCHED-STAR: A Price Based Shop Scheduling Module," *Journal of Manufacturing and Operations Management*, 1, 131–181.

PETRAKIAN, R. G. (1991), *Shop Floor Control in Wafer Fabs using Predictions and Pricing*, Unpublished PhD dissertation, University of California, Berkeley, Berkeley, CA.

PRITSKER, A. A. B., L. J. WATTERS AND P. M. WOLFE (1969), "Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach," *Management Science*, 16, 1, 93–108.

ROUNDY, R., HERER, Y., AND S. TAYUR (1989), "Price-directed Scheduling of Production Operations in Real Time," *Proceedings of the 15th Conference on Production Research and Technology*, Berkeley, CA.

WEIN, L. M. (1988), "Scheduling Semiconductor in Wafer Fabrication," *IEEE Transactions on Semiconductor Manufacturing*, 1, 3, 115–130.

WOLFF, R. W. (1989), *Stochastic Modeling and the Theory of Queues*, Prentice Hall, Englewood Cliffs, NJ.