

A Review of Real-Time Control Strategies for Furnace Batch Sizing in Semiconductor Manufacturing

JENNIFER K. ROBINSON

FabTime Inc., 325M Sharon Park Drive #219, Menlo Park, CA 94025

JOHN W. FOWLER

Department of Industrial and Management Systems Engineering, Arizona State University, Tempe, AZ 85283

JONATHAN F. BARD

Graduate Program in Operations Research, Department of Mechanical Engineering, University of Texas, Austin, TX 78713

This paper presents a review and evaluation of a number of strategies for the control of batch queueing processes in the semiconductor manufacturing industry. Special attention is paid to policies that include real-time information about the state of the manufacturing facility, particularly the expected timing of future arrivals. The studies evaluated highlight the potential benefits that might stem from taking full advantage of real-time information. These studies clearly show that using knowledge of future arrivals can, in many cases, lead to improvements in waiting time, even in the presence of prediction errors. Several avenues for future research are also discussed.

Keywords: Batch control policies, real-time information, semiconductor manufacturing

INTRODUCTION

This paper reviews and evaluates several strategies for the control of bulk service (batch) queueing processes in the semiconductor manufacturing industry. Particular emphasis is placed on those policies that incorporate detailed knowledge of the expected timing of future arrivals. This real-time information is available (from shop floor control systems) in most wafer fabrication facilities (fabs), but is seldom exploited in decision-making. Use of real-time information can yield improvements in operation.

The process for manufacturing integrated circuits consists of four basic steps; wafer fabrication, wafer probe, assembly or packaging, and final testing. Wafer fabrication has traditionally been the most costly phase of semiconductor manufacturing. It involves a complex sequence of processing steps that must be performed in a clean-room environment to reduce the threat to the wafers of particle contamination. Fabrication begins with a smooth (typically silicon) wafer upon which hundreds of circuits are layered through successive operations. During processing, a wafer goes through each of several primary processing steps many times. Production of a particular type of circuit requires a specific sequence of processing steps (deterministic with the exception of rework), with unique processing times at each step for that product type. Each processing step normally follows a very strict “recipe,” so that processing times are essentially constant.

Some of the processing steps are performed on individual wafers, others on lots (groups) of wafers, and still others on batches (collections) of lots. In this paper operations performed on individual wafers or lots of wafers are referred to as *serial* steps, while those performed on groups of lots are called *batch* steps. A lot generally consists of 24 or 48 wafers, while a typical batch contains up to six or eight lots. The collection of lots into batches results in a non-smooth product flow. Control of a fabrication facility generally involves the monitoring of different production sequences and processing time recipes for hundreds of products.

A typical performance measure used in evaluating the impact of control strategies on a single workstation is the expected waiting time of products in queue, which usually is minimized. Factory level strategies, on the other hand, attempt to minimize cycle times (i.e., time in system) or maximize throughput. Since processing times for particular products tend to be constant, minimizing waiting time in queue helps to reduce cycle time of wafers through the system. Long cycle times are particularly undesirable in the semiconductor industry because the yield of the process is inversely proportional to the amount of time wafers spend in production [1]. This declining yield occurs because the more time wafers spend in the fab, the more likely they are to become contaminated.

As indicated above, a common feature of several process steps is that lots are processed in batches in what is generally referred to as a furnace tube. There is a limit to the number of lots that may be processed in a particular batch, due to the limitations of the tube involved. These limitations may be physical or may depend on the physics of the process. Processing time is usually independent of the number of lots in a batch, and once a batch process begins, it cannot be interrupted to allow other lots to join. From a local perspective, when a tube is available and full loads are waiting, the decision to process a batch is obvious, since no advantage can be gained at that work area by waiting (although a decision may still be needed concerning which product type to process). However, when there is a tube available and only partial loads of products are waiting, a decision must be made to either start a (partial) batch or wait for more products to arrive.

Several control strategies have focused on the particular decision of choosing the batch size to be processed at a specific workstation at a given time. Such real-time strategies may be applied to either single or multiple tubes, and may consider either single or multiple products. Queueing systems of this sort, which can process more than one customer at a time, are known as bulk service queueing systems. A number of factors influence the operation of bulk service queueing systems, including the intensity and distribution of arrivals to the server(s), the number and mix of products, the maximum batch sizes of products that may be processed in a tube, and the processing times of products.

In the next section a number of studies concerning the general bulk service queueing problem, without particular reference to semiconductor manufacturing, are reviewed. These works document the development of the *general bulk service rule*, also called the *minimum batch size* (MBS) rule. This rule requires that a batch be made as soon as a server is available and the queue reaches a specified minimum level (i.e. threshold). Next, several studies that directly address the semiconductor manufacturing industry are examined. Finally, four control strategies that incorporate knowledge of future arrivals into the decision structure are more critically evaluated.

GENERAL BULK SERVICE QUEUEING SYSTEMS

A significant amount of research appears in the literature concerning batch processing, mostly in the area of queueing theory. None of this work includes knowledge of the expected state of the facility. Instead, decisions are made based solely on the current state of the system. Four papers which illustrate the evolution of the bulk service queueing problem are described in detail below, followed by a brief discussion of more recent work in the area of bulk service queueing systems.

On Queueing Processes with Bulk Service

Bailey [2] authored the first study of a bulk service queueing system. In his model, single customers arrive to a single server according to a Poisson process, and are served in batches. The size of each batch is a fixed maximum, where the implied minimum is one. Bailey derives the probability structure of the service intervals for both random and constant service times, and finds an equation for average waiting time. This is the mean value of the distribution of customers' waiting time. He next derives a rough inequality for the average waiting time, finding good upper and lower bounds given the parameters of the arrival and service distributions. Ultimately, he provides a practical, though imprecise, guide to finding batch sizes that avoid very long waiting periods.

Bulk Queues with Poisson Input

Neuts [3] uses Bailey's model as a starting point and goes on to propose his "general rule" for bulk service queues. He assumes that customers arrive singly according to a homogeneous Poisson process of rate λ , and are served in groups according to the following policy. If there are less than L customers waiting, wait until there are L present, then serve them all. If there are L or more but less than K (the capacity of the server), serve all of them together. If there are K or more waiting, serve K . The rest of the customers wait. Service times of successive groups are assumed to be conditionally independent given the batch sizes, but may depend on their magnitudes. The order of service is considered irrelevant in this study, which Neuts refers to as the $M/G_{L,K}/1$ system.

Waiting Time Distribution in a Poisson Queue with a General Bulk Service Rule

Medhi [4] presents a particular instance of Neuts' general rule, where the service times are distributed exponentially. He derives waiting time distributions for this system by studying the probability density function of the steady state number in the queue. For simplicity, he assumes that the service time distribution is independent of batch size, and has mean $1/\mu$. He proceeds to derive the steady state equations of the system, and the steady state equations for the waiting time in queue for an arrival. He finds the moments of the distribution, and shows that Little's Law [5] holds in the case of the general bulk service rule. Like the

papers previously discussed, Medhi's work provides a starting point from which other researchers can investigate specific examples. For example, Powell and Humblet [6] call the general rule a "vehicle holding strategy." They propose a numerical method to find the moments of the queue length distribution.

Optimal Control of Batch Service Queues

Deb and Serfozo [7] extend Neuts' work by deriving a cost function and investigating optimal policies for the MBS rule. They consider a batch queue in which each batch size is subject to control. Their idea is that policies which minimize the expected continuously discounted cost per unit time over an infinite time horizon are of the form of MBS as described by Neuts. The problem then involves computing the threshold value that minimizes the cost function. In Deb and Serfozo's model, customers arrive by a Poisson process with mean λ . Service times are independent and identically distributed, and do not depend on batch size. The process is reviewed whenever either a server becomes free or a customer arrives and the server is free. At that point, either no customers are served or a batch is served.

Deb and Serfozo's work has more potential for direct implementation than that of Neuts. The authors show that an MBS strategy can be used to minimize either the total discounted or average operating costs of a queue, and demonstrate how to find optimal values for L . No tests are included, however, to show how these methods work in practice. In particular, there is no examination of the effect on cost of using non-optimal values of L . Later work by Glassey and Weng [8] shows that using non-optimal values of L can result in significant deviations from optimality. However, Deb and Serfozo's results do show that if the optimal value of L is used, a better control policy cannot be found using only information about the current state of the system.

Other Work on Bulk Service Queueing Systems

A number of other authors have addressed the problem of bulk service queueing systems. Crabhill *et al.* [9] provide a bibliography of early research concerning the optimal design and control of queues. Chaudhry and Templeton [10] dedicate an entire book to bulk arrival and bulk service queues. Both Bahary and Kolesar [11] and Bitran and Tirupati [12] address the issue of multiple product types served by bulk queueing systems. Neuts and Nadarajan [13] extend Neuts' work (described above) to include multiple servers.

Makis [14] considers batch service queueing systems with finite and infinite service capacities, where costs are charged for servicing the customers and for holding them in the system. Das [15] studies a single server system where the customers arrive according to a Poisson process and are served in batches under an MBS rule with constant service rate. Chaudhry, *et al.* [16] use an analytical approach to show how to numerically evaluate steady-state probabilities and moments for number in system at different time epochs for bulk and nonbulk queues. Brière and Chaudhry [17] study single-server bulk-service queues, where the capacity of the single server is a random variable. Sim and Templeton [18] consider a transportation system with a number of vehicles (with capacity constraints) that take passengers from a source to various destinations. While all of the papers discussed above give some indication of analytical results for bulk service queueing systems, none takes into account the semiconductor manufacturing industry or real-time control systems.

WORK SPECIFIC TO THE SEMICONDUCTOR MANUFACTURING INDUSTRY

A substantial body of research concerning flow control in the semiconductor manufacturing industry has appeared in recent years. With a focus on bottlenecks, Wein [19] assesses the impact of several different input

mechanisms and scheduling rules on the performance of semiconductor fabs. He concludes that order release has a large impact on wafer fab operations. Glassey and Resende [20] present a closed loop job release mechanism, called starvation avoidance, for use in high volume fabs where output is limited by machine capacity rather than by outside conditions. Lozinski and Glassey [21] continue along the same lines, presenting a graphical tool for inventory and production control that is designed to help shop floor personnel make decisions consistent with a starvation avoidance policy. These authors all point to the importance of scheduling the bottleneck machine to minimize starvation but none of them deals with the accompanying impact of the batch size decision. An important contribution could be made by reducing starvation of the bottleneck machine through informed batch size decision-making.

Some work has also been done concerning sequences of batch and serial machines in semiconductor manufacturing. Gurnani, Anupindi, and Akella [22], for example, study the impact of loading policies for the batch machine on a two-stage serial-batch system. de Haut de Sigy [23] develops a stochastic dynamic programming model to determine the loading policy of a batch machine following a serial machine in a single product system. Neither paper explicitly treats the impact of the batch machine on a subsequent serial processor.

The queuing theoretic studies described in the previous section all rely solely on information concerning the current state of the system. Many semiconductor fabs, however, now have sophisticated shop floor control systems. These systems contain information about the current state of each lot, as well as the length of time the lot has been in that state. This information could easily be used to predict the timing of future arrivals. As the following papers show, effective use of such information can lead to significant decreases in flowtimes through the system.

METHODS INCORPORATING KNOWLEDGE OF FUTURE UPSTREAM EVENTS

This section contains a detailed review and evaluation of four recently developed control strategies: Glassey and Weng's [8] *dynamic batching heuristic*; Fowler, Phillips, and Hogg's [24] *next arrival control heuristic*; Weng and Leachman's [25] *minimum cost rate heuristic*; and Robinson, Fowler, and Bard's [26] *rolling horizon cost rate heuristic*. Both merits and weaknesses of the four methods are identified for the purpose of highlighting the potential for research in this area.

Dynamic Batching Heuristic

Glassey and Weng [8] demonstrate how to reduce the average waiting time of lots at a batch workstation. Their paper looks at a particular decision, dynamic batch sizing, to answer questions about how to use global information about the state of a factory, and how much improvement might be realized by using it. They look at the single product-single server case with constant processing times for particular operations and attempt to minimize the expected waiting time of lots in queue. The purpose of the paper is to compare the results obtained when using an optimal MBS policy with the results generated when using their own adaptive rule, the dynamic batching heuristic (DBH). When using DBH, the authors begin by predicting the arrival times of the next few lots. For this reason DBH is also referred to as a look-ahead strategy.

Glassey and Weng attempt to minimize the expected total waiting time of both the lots already in queue and the next few lots that are expected to arrive. The decision to make a batch implies that any new arrivals that come before the batch is finished processing will have to wait. On the other hand, the decision to wait for more arrivals implies that the lots already in the queue will have to wait. DBH is used to decide at what point

the machine should begin processing a new batch in an effort to minimize the overall average waiting time of both lots already in queue and lots scheduled to arrive.

The test model used is a single machine, operating without breakdowns, with a tube capacity of five lots of wafers for all products and a processing time of 25 time units for all products. The average waiting time in queue (for both the DBH and MBS policies) is a function of two parameters, the coefficient of variation of the interarrival process and the traffic intensity. Traffic intensity is defined as:

$$\text{arrival rate} / [(\text{service rate} \times (\text{tube capacity}))] \quad (1)$$

The authors test both uniform and Poisson interarrivals over 100,000 time units in 10,000 time unit blocks, and compare the standardized average delay for the two control strategies (DBH and MBS) over a range of traffic intensities between 0.1 and 0.9. They run simulations using a simulation language called the Berkeley Library of Objects for Control and Simulation (BLOCS), which is written in Objective-C. They give no indication of computation times for their runs, though one would expect given the increase in necessary computations, that DBH would require greater CPU time than MBS.

They first obtain some experimental results for the MBS rule alone. They show that MBS equal to one is optimal in light traffic, indicating that in this case the tube should start service whenever there are any lots waiting. Conversely, they show that the minimum size batch should be a full load in very heavy traffic. They proceed to examine the impact on system performance of using non-optimal minimum batch sizes. Although Deb and Serfozo document a method for finding the optimal minimum batch size, they (Deb and Serfozo) include no results concerning the impact of non-optimal choices. Glassey and Weng indicate that the system may perform poorly when there has been a bad choice of minimum batch size, particularly in light traffic. They also show that since MBS equal to one is optimal in light traffic and performs almost as well as any other batch size in heavy traffic, the best choice for minimum batch size in time-varying or unknown traffic is MBS equal to one.

Next, DBH is developed, using the following notation:

C	Tube capacity
t_0	Time epoch that the tube is idle and the number in the queue is positive
t_i	Arriving epoch of the next i^{th} lot after t_0
q	Number of lots in queue at epoch t_0
T	Processing time
L	Look-ahead number. Assumes that the next L arrival epochs are known with certainty at t_0

The DBH formulation is based on the following logical conditions:

1. The time of loading the tube is either at the time that it becomes idle and there are lots waiting, or, if it has been idle, at the time when some lot arrives.
2. When $q \geq C$, the machine starts service right away. Waiting will only result in more delay under such circumstances.

Since it becomes difficult to predict the later arrivals in a long planning horizon, Glassey and Weng propose a heuristic for operating the tube in a planning horizon that is equal to the processing time (T) of the product. The overall heuristic is included as Figure 1.

The number of predicted arrival epochs is always less than or equal to $C - 1$, where C is the tube capacity, because it is assumed that there is at least one lot waiting if a decision is being made and that once a full load is waiting a batch will be made. Therefore $L \leq C - 1$. The last possible arrival considered at time t_0 is the earlier of the last arrival that takes place during processing time T or the arrival that results in a full load. At time t_0 , if the number of lots in queue, q , is greater than or equal to the capacity of the furnace, then a load of C lots is started immediately. Otherwise, the heuristic decides how many lots for which to wait by calculating the areas under the curve of cumulative number of lots in queue versus time. This curve includes the net delay for both lots already in queue and lots that will arrive during the planning horizon.

Starting q lots at time t_0 causes no delay for lots already in queue. However, a lot arriving at epoch t_i , provided t_i is within the planning horizon, will wait for at least $(T + t_0 - t_i)$, the time from arrival until the lot being processed is finished. Waiting for the lot that arrives at t_i and then starting a batch creates no delay for the arriving lot but causes the q lots already waiting to each be delayed by $(t_i - t_0)$. The loading time is chosen that leads to the largest overall savings in waiting time. The corresponding epoch i is:

$$i = \operatorname{argmax} (j x(T + t_0 - t_j) - q x(t_j - t_0) : 0 \leq j \leq j_{\max}) \quad (2)$$

where j_{\max} is the last arrival considered for the selected value of L .

Glassey and Weng demonstrate the heuristic for $L = C - 1$, meaning that the algorithm considers all arrivals up to a full load. They then document the effect of using different values for this look-ahead number, L , and find that choice of look-ahead number does not affect average queue time for regular arrivals (uniform arrivals with range equal to mean). However, in the case of Poisson arrivals they indicate that each additional lot considered gains about half the remaining potential improvement. In light of this observation, they speculate that the performance is a convex function of the look-ahead number with decreasing marginal returns on additional information.

They also examine the sensitivity of the heuristic to prediction errors, which are assumed to be normally distributed with mean zero and standard deviation s_e . Letting s_a be the standard deviation of the interarrival times and assuming that $s_e = k * s_a$, where k is a user-specified constant, the authors find that even with large prediction errors ($k = 0.5$), the dynamic batching heuristic outperforms the MBS rule whenever traffic intensity is greater than or equal to 0.2. This suggests that DBH is robust to prediction errors.

In general, they find that DBH yields significant improvements over MBS whenever traffic intensity is moderate (0.30 to 0.70), reducing waiting times by approximately 50%. In extreme conditions – both very heavy and very light traffic intensities – DBH attains less improvement. This diminished improvement occurs because in extreme cases DBH generally makes the same decision as the optimal MBS rule. In heavy traffic, full loads are almost always processed while in light traffic single lots are usually loaded. Glassey and Weng also find that increased variance of the arrival process reduces waiting times when the arrival epochs can be predicted, at least in medium traffic.

Glassey and Weng were the first to publish work that incorporates the use of knowledge of future arrivals into the batching decision. They demonstrate a clear improvement over the results attained using MBS. They also obtain some useful results regarding the effect of non-optimal minimum batch sizes.

Some potential weaknesses to Glassey and Weng's control strategy follow:

1. The algorithm may be difficult to implement because of the need to select a look-ahead number.

2. DBH only applies to the single-product, single-server case. It is not clear how to extend the algorithm to multiple products and multiple servers, which are probably more realistic in a factory setting. Determining the best value of L might be somewhat difficult in the multi-product case.
3. Ultimately it would be useful to know how the batching decisions made using DBH affect the factory as a whole. DBH may reduce wait time in queue at a particular workstation, but could increase the variability of the workstation output process and disrupt flow further along in the system.

Next Arrival Control Heuristic

Fowler *et al.* [24] base their work on a number of the assumptions used by Glassey and Weng. They, too, develop and test a control strategy for bulk service queueing systems, giving particular emphasis to the batch size decision in semiconductor manufacturing. Their objective is to minimize the average time that lots spend waiting to be processed, incorporating knowledge of future arrivals into the decision. Unlike Glassey and Weng, however, Fowler *et al.* derive heuristics for both the single and multiple product cases. Both cases utilize a single server. They evaluate their control strategies using simulation, and compare their steady state results with results generated using both MBS and DBH strategies.

Like Glassey and Weng, Fowler *et al.* assume that all processing times are constant and independent of the number of lots in the batch, and that whenever a full load of any product is available a batch is started immediately. They begin their research by modifying the control strategy used by Glassey and Weng, who found that about half the potential benefit in using DBH is gained by looking ahead only to the next arrival. They (Fowler *et al.*) note also that the further ahead one looks, the greater the potential impact of the decision on arrivals that occur outside of Glassey and Weng's time horizon of T time units. Therefore, they modify DBH to consider only the next arrival, and make the decision at that point on whether to start the batch immediately or wait for the next arrival. If the decision is to wait, the inventory in front of the batch machine builds and the decision logic is repeated at the time of the next arrival (i.e., a rolling horizon). This policy is different from the decision structure used by Glassey and Weng in which a decision is made at time t_0 for which specific arrival to wait. Fowler *et al.*'s variation allows more information about the impact of the decision on future arrivals to be used by extending the planning horizon to consider arrivals that take place after $t_0 + T$. It is also more responsive to system changes such as prediction errors that take place after t_0 . The modified heuristic is called the next arrival control heuristic (NACH).

In using NACH, Fowler *et al.* identify two decision points. They define a "push" decision as the decision made when lots arrive to the queue and a furnace tube is available for processing. A "pull" decision is the decision made when a furnace tube becomes available and there is at least one lot waiting to be processed. At each decision point, NACH evaluates the delay incurred by the lots already in queue if they are forced to wait for the next arrival, as well as the delay incurred by the next arrival if a batch is started immediately. If the delay incurred by waiting is greater, a batch is started right away. Otherwise, no batch is made and the decision is repeated at the time of the next arrival. In the single product case, NACH is used for both the push and pull decisions. This parallels Glassey and Weng's calculation of (2), except that i is always equal to 0 or 1.

The control strategy is more complex for the multiple products case, because the decision to make a batch of one-product results in additional wait time for all of the other products. Fowler *et al.* only consider the impact of the decision on the arriving product type in the push decision, leaving the contention between the products to the pull decision. The decision logic of the multi-product control strategy is outlined as Figure 2.

The authors use *weighted shortest processing time* (WSPT) to decide between those products with full loads available, where the weight for each product is its processing time. They indicate that WSPT has been shown

to minimize the mean flowtime [27]. When no full loads are available, the pull decision logic gets fairly complicated. First NACH is used for each product that has lots already waiting in queue, taking into account only the product currently under consideration. If the individual NACH evaluations all indicate to wait, the decision is to wait. If they all indicate to make a batch, WSPT is used to decide among the products and a batch is made. However, if some of the NACH evaluations indicate to wait, and others to make a batch, the interactions of the different products must be considered. The total delay to all products caused by following the NACH evaluation for each product is calculated and the product and associated action which cause the minimum overall delay are selected.

Fowler *et al.* include extensive experimental testing of their control strategy, as well as of the MBS and DBH strategies. They strengthen the simulation conditions used by Glassey and Weng by increasing run length to allow for more batches of data to be collected. All simulations are performed using SIMAN with FORTRAN subroutines. For the single product case, they make comparisons with Glassey and Weng's results, using the same sequences of random numbers to test all control strategies. The use of common random numbers allows comparison under a similar arrival pattern so that any differences can be attributed to differences in the control strategies rather than to random variation.

The authors begin by replicating Glassey and Weng's results for DBH and MBS, and are able to obtain very similar numbers. They then test the same case using their NACH-based control strategy. They find only small differences between DBH and NACH results in the cases without prediction errors, and slight improvement from using NACH in most of the error prediction cases. This improvement in the error cases might be expected, considering that the NACH strategy makes decisions sequentially, and thus has more frequent access to updated arrival predictions. The exception is in the case of large errors and light traffic, in which case DBH performs better. Both NACH and DBH outperform MBS in most cases, the exception being the case where there is virtually no waiting.

The comparisons with MBS are very similar to those made by Glassey and Weng. Since Glassey and Weng's heuristic does not extend to the multi-product case, Fowler *et al.* compare their results for that case to results obtained using MBS only. It should be noted that the MBS results are for the optimal choice of threshold values. Comparisons are based on a factorial design of experiments that incorporates seven factors, all at either two or three levels. The factors considered are listed in Table 1. Fowler *et al.* use analysis of variance to compare the performance of the two control strategies under the different levels of the other factors. They find that on the average (across all other factor level combinations) NACH outperforms MBS. The effect is more pronounced at a traffic intensity of 0.50 than at a traffic intensity of 0.80, which is consistent with the results for the single product case. As the traffic intensity climbs to 0.80, NACH acts like MBS. Cases with exponential interarrival times show more improvement than those with uniform interarrival times. This makes sense, considering that NACH is more responsive to change than MBS.

They next compare the performance of the two control strategies when errors in the prediction of future arrivals are included. They find that in all cases observed the NACH-based strategy performs better than the MBS strategy. In general, the percent difference between the strategies decreases with increase in k . In other words, as more error is present, less improvement is gained through knowledge of future arrivals, which is consistent with one's intuition. However, even with error, improvement can be shown; the error comparison is used to confirm the robustness of the NACH-based control strategy.

Fowler *et al.* conclude that their research demonstrates the potential benefit in using knowledge of future arrivals to help make batching decisions for bulk service queueing systems. Their data support the fact that NACH performs at least as well as MBS and often yields considerable improvement. An additional advantage to using NACH over optimal MBS policies is that NACH-based strategies are parameter-free, while MBS strategies require setting the threshold parameter(s) for every set of system conditions. Since NACH reacts to changes in system conditions, it is more adaptive than MBS to change. It is less clear that

NACH outperforms DBH, but it appears to perform at least as well. Again, NACH has the advantage over DBH of being parameter-free. Also, NACH is able to handle the multiple products case, which Glassey and Weng do not consider.

Nevertheless, several shortcomings of the work become apparent on closer inspection:

1. Two different decision criteria are used in the multiple product pull decision, NACH and WSPT. It might be more intuitively appealing to have a single decision criterion. Also, the validity of using individual NACH evaluations for each product without considering the interactions and then calculating interaction effects based on the results of the individual evaluations, might be questioned.
2. Clearly, the authors are not using all of the available knowledge of future arrivals. They only consider the next arrival and then decide whether or not to make a batch. The problem with this approach is that a situation can occur where NACH is used to evaluate the next arrival and indicates that it is better to make a batch now than to wait for the arrival. However, there might be a second arrival scheduled to take place immediately after the first, so that there would be less total waiting time incurred by waiting for both arrivals than by making a batch right away. NACH will not give the better solution in this case.
3. As mentioned earlier under the DBH results, the impact of NACH-based decisions on the system as a whole needs to be assessed. Although such decisions generally result in shorter average waiting times in queue than MBS decisions, this improvement might be gained at the expense of smoothness of output. It should be noted here that in addition to the multiple products-single server work described above, Fowler [28] also considers the single product-multiple server and multiple product-multiple server cases.

Minimum Cost Rate Heuristic

Weng and Leachman [25] address the same problem examined by Glassey and Weng and by Fowler *et al.* However, their minimum cost rate heuristic (MCR) has some noticeable differences from the two control strategies previously described. MCR seeks to minimize the holding cost per unit time, which is like minimizing the weighted (by cost) number of lots in queue. Only when it is assumed that there is a unit holding cost per unit time of one for all products does MCR minimize the number of lots in queue. In this case, average time in the queue is minimized (by Little's Law).

In the single product-single server case, MCR is an extension of DBH. The primary difference is that instead of using a fixed horizon for look-ahead (the processing time), MCR uses the processing time plus prior waiting time. Weng and Leachman use the same basic notation used for DBH, and also introduce the following:

$TC(t_i)$ Total holding cost from t_0 to $(T + t_i)$ if the furnace is loaded at t_i

$CR(t_i)$ Holding cost per unit time (cost rate) from t_0 to $(T + t_i)$ if the furnace is loaded at t_i

The look-ahead number, L , is no longer necessary. As with DBH and NACH, the start time will only be at the current time, t_0 , or at future arrival epochs. If the operation starts at t_i , the furnace will not be free again until $(t_i + T)$, meaning that the scheduling horizon is $(t_i + T - t_0)$, during which time all lots except those in process are waiting. MCR requires calculation of the holding cost per unit time for all possible loading epochs (up to the point where there is a full load of the product waiting). The reason for evaluating all possible loading epochs is that the cost rate function is not strictly decreasing. For example, waiting for the second arrival can, in some cases, yield a smaller value of the cost rate function than making a batch after the first arrival, particularly if the arrival times are spaced very irregularly. In this manner, Weng and Leachman

overcome one of the shortcomings of NACH, and to a lesser extent DBH (when L is not equal to C-1). Their algorithm is shown Figure 3. Note that as with DBH, MCR first determines how many arrivals for which to wait and then waits until that many have arrived before making another decision.

The cost rate is calculated as follows. First, the total holding cost experienced during the scheduling horizon is calculated as

$$TC(t_i) = q(t_i - t_0) + \sum_{j_{i_0} < t_j \leq t_i} (t_i - t_j) + \sum_{j_{i_i} < t_j \leq t_i + T} (t_i + T - t_j) \quad (3)$$

where $q \leq C$. Then, the holding cost per unit time, or cost rate, is

$$CR(t_i) = TC(t_i) / (t_i + T - t_0) \quad (4)$$

Weng and Leachman next extend the cost rate function to the multiple product case. This extension involves finding the best loading time for each product according to the minimum cost rate of all possible loading epochs and then choosing the product with the overall minimum cost rate. The following additional notation is introduced:

$t_{0,j}$ t_0 for product j (time a decision is to be made - the same for all products)

$t_{i,j}$ Arrival epoch of the i^{th} lot of product j after time $t_{0,j}$

C_j Furnace capacity for product j

P Product set

q_j The queue length of product j at time epoch $t_{0,j}$

Q Subset of **P** such that $j \in \mathbf{Q}$ if $q_j \geq C$

J Number of products: $J = |\mathbf{P}|$

T_j Process time of product j

$TC_k(t_{i,j})$ Total holding cost of product k from $t_{0,j}$ to $(T_j + t_{i,j})$ if product j is loaded into furnace at $t_{i,j}$, where $k = 1, \dots, J$

$CR_j(t_{i,j})$ Holding cost per unit time (cost rate) from time $t_{0,j}$ to $(T_j + t_{i,j})$ if product j is loaded into the furnace at $t_{i,j}$, where $j = 1, \dots, J$

The holding cost of the product considered for loading, product j , at time $t_{0,j}$, is

$$TC_j(t_{0,j}) = \max(q_j - C_j, 0) \times T_j + \sum_{n_{0,j} < t_{n,j} \leq t_{0,j} + T_j} (t_{0,j} + T_j - t_{n,j}) \quad (5)$$

and at time $t_{i,j}$ is

$$TC_j(t_{i,j}) = q_j(t_{i,j} - t_{0,j}) + \sum_{n_{0,j} < t_{n,j} \leq t_{i,j} + T_j} (t_{i,j} - t_{n,j}) + \sum_{n_{i,j} < t_{n,j} \leq t_{i,j} + T_j} (t_{i,j} + T_j - t_{n,j}) \quad (6)$$

for $i > 0$. Note that $(q_j + i)$ is always less than or equal to C_j . For the other products $k \neq j$, the holding cost is

$$TC_k(t_{i,j}) = q_j(t_{i,j} + T_j - t_{0,k}) + \sum_{n_{0,k} < t_{n,k} \leq t_{i,j} + T_j} (t_{i,j} + T_j - t_{n,k}) \quad (7)$$

for $i \geq 0$. The holding cost per unit time for product j being loaded at $t_{i,j}$ is

$$CR_j(t_{i,j}) = \sum_{k=1}^J TC_k(t_{i,j}) / (t_{i,j} + T_j - t_{0,j}) \quad (8)$$

The algorithm used to decide when the furnace should begin processing is given in Figure 4. Note that when full loads are available, MCR is used to decide between them and a batch is always made at time t_0 . This is to prevent the case where the furnace is forced to wait when there are full loads available for processing. In any case, the product with the minimum cost rate is chosen. Ties are broken in favor of the product with the shortest processing time.

Weng and Leachman use simulation to compare MCR with MBS, DBH, and NACH. They consider a single furnace with a capacity of five lots (and no breakdowns), to which the time between lot arrivals is exponentially distributed. As before, arrival rates are defined implicitly by the traffic intensity at the furnace.

They first test the single product case, where the processing time of a lot is 25 time units. Note that this is the same case tested previously by Glassey and Weng. They compare average queue lengths for MBS, DBH, NACH, and MCR at various traffic intensities. All of their simulations are performed using BLOCS. The authors find that the performances of DBH, NACH, and MCR are all very close, with DBH yielding essentially the same results as MCR. All of the look-ahead cases outperform MBS.

They next consider the multiple product case, where there are four products each of which needs a processing time of 60, 120, 180, and 240, respectively. Arrival patterns for each product at the furnace are the same and are exponentially distributed. The authors compare the average queue lengths resulting from use of MBS, NACH, DBH (referencing unpublished research done by Glassey and Weng to extend DBH to the multiple product case), and MCR. They conclude that MCR performs better than the other three methods, while DBH and NACH both outperform MBS. In light traffic, NACH performs better than DBH, while the converse is true in heavier traffic. This probably occurs, they explain, because in heavy traffic NACH performs essentially as WSPT, but in lighter traffic the flexibility of NACH allows responsiveness to more information. They also note that MBS and DBH have fairly uniform queue lengths by product, while NACH has more varied queue lengths. They indicate that this is because methods based on shortest processing time (which NACH implicitly is) tend to distribute waiting times in proportion to processing times.

Weng and Leachman conclude that they “have shown how to use arrival information to improve dispatching decisions on a batching workstation.” In particular, MCR demonstrates several advantages over the strategies previously discussed. Although it yields no noticeable improvement over DBH and NACH in the single product case, it apparently reduces queue lengths in the multiple product case. MCR is also somewhat easier to follow than NACH because it requires the same essential algorithm for both the single and multiple product cases, i.e. all decisions are made based on the cost rate function. MCR catches some potential improvement which is not incorporated into Fowler *et al.*'s control strategy by looking ahead at all future loading epochs. It is to be expected that evaluating all possibilities will yield some improvement over the performance of NACH, which only looks at the next arrival.

However, there are a number of issues left unresolved by Weng and Leachman's research, including the following:

1. MCR requires more computations than NACH. This is because every time a decision is made, the cost rate function is calculated for all products and all possible loading epochs. The authors fail to

show that the small improvement in queue length afforded by looking at all possible future loading epochs justifies the increased computations.

2. Like DBH, MCR decides at each decision point exactly when the next batch will be processed. This may cause problems when it comes to implementation, particularly when there is error in the arrival predictions.
3. While they do make observations concerning the average queue lengths that result from using the different control strategies, Weng and Leachman make no attempt to examine the resulting output processes. By introducing the cost rate function, MCR could be increasing the variability of the output leaving the furnace. This might cause problems downstream, particularly for future serial operations.

Rolling Horizon Cost Rate Heuristic

Robinson *et al.* [26] present a heuristic that is essentially a combination of NACH and MCR. The cost rate function used in MCR is incorporated into the rolling planning horizon used in NACH. In other words, at each decision point, a cost function is calculated for each possible loading epoch, up to full loads, for each product. The product and loading epoch that yield the minimum value of the cost rate function are identified. If the resulting epoch corresponds to t_0 , the decision is made to process a batch immediately. Conversely, if the resulting epoch corresponds to a time other than t_0 , the decision is made to wait for the next arrival. At the time of the next arrival the cost rate functions are again calculated and another decision may be made to wait. In this way, the heuristic takes advantage of the simple structure of the cost rate function and also the rolling time horizon used in NACH. This heuristic is called the rolling horizon cost rate heuristic (RHCR).

Robinson *et al.* use the same notation introduced by Weng and Leachman. The cost rate and holding cost functions for single products are as described in (5)-(6). The scheduling algorithm used is depicted in Figure 5. RHCR for the single product case is tested against MBS, NACH, and MCR, using an experimental design similar to that of Glassey and Weng. Here a product arrives at a furnace with a capacity of five lots and a processing time of 25 time units per batch, with exponentially distributed interarrival times. Simulation is used to compare policies, with a full factorial experiment at nine traffic intensities (ranging from 0.1 to 0.9) and two levels of coefficient of error (0.0 and 0.3) for each of the four control strategies.

The results reported for this experiment, broken down by level of error in arrival predictions, are shown in Figure 6. They indicate that the overall average delay for RHCR is not significantly different from that of NACH (at the 95% confidence level). However, RHCR does appear to outperform both MBS and MCR. When there is no error in the prediction of future arrivals, RHCR and MCR yield almost identical results. When error is present the performance of MCR degrades somewhat, and both NACH and RHCR have shorter delays. This makes sense, since RHCR uses more (potentially erroneous) information than NACH uses but mitigates the impact by repeating the decision process at each new arrival.

Next, the authors extend RHCR to the multiple product case. The total cost rate and holding cost functions are given in (5)-(8). The scheduling algorithm used for the rolling horizon case is included as Figure 7. RHCR is tested against MBS, NACH, and MCR, using an experiment similar to one used by both Fowler *et al.* and Weng and Leachman. Factors tested include: processing times, tube capacities by product, product mix, arrival traffic intensity, and coefficient of error in arrival prediction. The results of this experiment are shown in Figure 8.

Robinson *et al.* find that the minimum overall delay is observed when using NACH, followed by (in order of performance) RHCR, MCR, and MBS. When there is no error in the prediction of future arrivals, RHCR has

the shortest delay, followed by NACH, MCR, and MBS. However, when there is error present, NACH outperforms RHCR. This makes sense when one considers that RHCR uses more of the potentially inaccurate information than NACH does. Even though RHCR recalculates the cost rates at each new decision point, some damage has already been done by the false data. MCR, which does not recalculate the cost rates once the minimum has been selected, is most damaged by prediction errors.

The authors take their analysis one step further and introduce an extension that looks at a small system of a batch machine followed by a serial machine. They extend the cost rate functions used to account for delays incurred while waiting at the serial machine and consider the time in the entire system instead of just the time in queue. They find that slight improvements are possible at light to moderate traffic intensities. These improvements vanish under high traffic conditions, apparently because the heuristic sometimes makes decisions that force the serial processor to be starved for long periods of time. The serial results were reported to hold for uniform instead of exponential interarrival times. Their conclusion is that attention should be focused on control of the batch step rather than including information on the subsequent serial step.

The results of Robinson *et al.* consistently support the work of Glassey and Weng, Fowler *et al.*, and Weng and Leachman, and reflect the small marginal improvement that is possible by taking the best of the prior approaches. The authors make two primary contributions. First, they respond to some of the shortcomings of prior methods by incorporating the rolling horizon used in NACH into the cost rate function used in MCR. They also take another step towards integrating batch size decisions into the overall control policy of the fab by extending their heuristic to take into account a downstream serial processor.

However, the authors leave the following issues unresolved:

1. Semiconductor manufacturing in reality consists of repeated sets of batch and serial steps, where a product visits the same workstation several times during processing. To validate their conclusions for the wafer fab as a whole, the authors should extend their algorithms to encompass multiple loops of batch and serial processors, or at least explore how the algorithms perform on a system basis.
2. The improvements seen in using RHCR are gained at the expense of increased computation time. It is not clear that the decrease in cycle time warrants the increased cost of computation.
3. RHCR is only derived for single servers at each workstation. The results will have more credibility if the algorithm is extended to allow for multiple servers.

COMPARISON OF BATCH SIZE CONTROL STRATEGIES

The papers described above illustrate an evolution in the use of real-time information to make intelligent batch size decisions. All four real-time strategies appear to offer improvements over strict use of a minimum batch size policy. A summary of the primary characteristics of the control strategies discussed is included as Table 2. When no error is present, cost-rate based strategies appear to yield slightly smaller delays than NACH-based strategies. However, the performances of DBH and MCR tend to degrade with increased error in arrival predictions. The results from RHCR support the idea that it is worthwhile in these cases to incorporate the rolling horizon introduced by Fowler *et al.*

CONCLUSIONS

The use of control strategies for bulk service queueing systems has evolved tremendously since Bailey first studied the problem. Neuts, Deb and Serfozo, and Medhi all contributed to the development of the minimum batch size strategy, which is fairly easy to implement and thus widely used today. Deb and Serfozo showed that if the optimal value for the minimum batch size (which must be recalculated whenever system conditions change substantially) is used, a better control policy cannot be found using only information about the current state of the system.

However, in many of today's factories, including semiconductor manufacturing facilities, information is available about the current state of the system. DBH, NACH, MCR, and RHCR all provide means for exploiting this knowledge with the intention of reducing time spent waiting in queue. It seems clear from observing the work of Glassey and Weng, Fowler *et al.*, Weng and Leachman, Robinson *et al.* that using knowledge of future arrivals can lead to improvements in waiting time, even in the presence of prediction errors.

A number of avenues for future research remain. The generalization of the results to the wafer fabrication process as a whole will have more validity if the algorithms are extended to encompass multiple loops of batch and serial processing steps. This will give an indication of the impact of re-entrant flow on the system. It is also desirable to extend the heuristics besides NACH to include multiple servers at the different process steps. The research performed so far has dealt with a very simplified system. As more of the complications of the wafer fabrication process are incorporated into the analysis, the results should provide more benefit in real-world situations.

References

1. L. Wein, "On the relationship between yield and cycle time in semiconductor wafer fabrication," *IEEE Transactions on Semiconductor Manufacturing*, vol. 5, no. 2, 156-158, 1992.
2. N. T. Bailey, "On queueing processes with bulk service," *Journal of the Royal Statistical Society B*, vol. 16, no. 1, pp. 80-87, 1954.
3. M. F. Neuts, "A general class of bulk queues with poisson input," *Annals of Mathematical Statistics*, vol. 38, no. 3, pp. 759-770, 1967.
4. J. Medhi, "Waiting time distribution in a poisson queue with a general bulk service rule," *Management Science*, vol. 21, no. 7, pp. 777-782, 1975.
5. J. D. C. Little, "A proof of the queueing formula: $L = \lambda W$," *Operations Research*, vol. 9, no. 3, pp. 383-387, 1961.
6. W. B. Powell and P. Humblet, "The bulk service queue with a general control strategy: theoretical analysis and a new computational procedure," *Operations Research*, vol. 34, no. 2, pp. 267-275, 1986.
7. R. Deb and R. F. Serfozo, "Optimal control of batch service queues," *Advances in Applied Probability*, vol. 5, no. 2, pp. 340-361, 1973.
8. C. R. Glassey and W. W. Weng, "Dynamic batching heuristic for simultaneous processing," *IEEE Transactions on Semiconductor Manufacturing*, vol. 14, no. 2, pp. 77-82, 1991.
9. T. B. Crabhill, D. Gross, and M. J. Magazine, "A classified bibliography of research on optimal design and control of queues," *Operations Research*, vol. 25, no. 2, pp. 219-232, 1977.
10. M. L. Chaudhry and J. G. C. Templeton, *A First Course in Bulk Queues*. New York: John Wiley & Sons, 1983.
11. E. Bahary and P. Kolesar, "Multilevel bulk service queues," *Operations Research*, vol. 20, no. 2, pp. 406-420, 1972.
12. G. R. Bitran and D. Tirupati, "Approximations for product departures from a single-server station with batch processing in multi-product queues," *Management Science*, vol. 35, no. 7, pp. 851-878, 1989.
13. M. F. Neuts and R. Nadarajan, "A multiserver queue with thresholds for the acceptance of customers into service," *Operations Research*, vol. 30, no. 5, pp. 948-960, 1982.
14. V. Makis, "Optimal control characteristics of a queueing system with batch services," *Kybernetika*, vol. 21, no. 3, pp. 197-212, 1985.
15. P. Das, "A general class of bulk service queue with constant service time," *Pure and Applied Mathematica Sciences*, vol. 25, no. 1-2, pp. 95-99, 1987.
16. M. L. Chaudhry, B. R. Madill, and G. Brière, "Computational analysis of steady-state probabilities of $M/G^{a,b}/1$ and related nonbulk queues," *Queueing Systems*, vol. 2, no. 1, pp. 93-114, 1987.
17. G. Brière and M. L. Chaudhry, "Computational analysis of single-server bulk-service queues, $M/G^Y/1$," *Advances in Applied Probability*, vol. 21, no. 1, pp. 207-225, 1989.
18. S. H. Sim and J. G. C. Templeton, "Steady state results for the $M/M^{(a,b)}/c$ batch-service system," *European Journal of Operational Research*, vol. 21, no. 2, pp. 260-267, 1985.

19. L. Wein, "Scheduling semiconductor wafer fabrication," *IEEE Transactions on Semiconductor Manufacturing*, vol. 1, no 3, pp. 115-130, 1988.
20. C. R. Glassey and M. G. Resende, "Closed-loop job release control for VLSI circuit manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, vol. 1, no. 1, pp. 36-46, 1988.
21. C. Lozinski and C. R. Glassey, "Bottleneck starvation indicators for shop floor control," *IEEE Transactions on Semiconductor Manufacturing*, vol. 1, no. 4, pp. 147-153, 1988.
22. H. Gurnani, R. Anupindi, and R. Akella, "Control of batch processing systems in semiconductor wafer fabrication facilities," *IEEE Transactions on Semiconductor Manufacturing*, vol. 5, no. 4, pp. 319-328, 1992.
23. R. J. de Haut de Sigy, "Loading control policy for a batch machine," Report No. LMP 90-001, Laboratory for Manufacturing and Productivity, Massachusetts Institute of Technology, 1990.
24. J. W. Fowler, D. T. Phillips, and G. L. Hogg, "Real time control of multiproduct bulk service semiconductor manufacturing processes," *IEEE Transactions on Semiconductor Manufacturing*, vol. 5, no. 2, pp. 158-163, 1992.
25. W. W. Weng and R. C. Leachman, "An improved methodology for real-time production decisions at batch-process work stations," *IEEE Transactions on Semiconductor Manufacturing*, to appear, 1993.
26. J. K. Robinson, J. W. Fowler, and J. F. Bard, "The use of upstream and downstream information in scheduling semiconductor batch operations," Report No. ORP93-06, Graduate Program in Operations Research, Technical Report Series, The University of Texas at Austin, 1993. To appear in *International Journal of Production Research*.
27. K. R. Baker, *Introduction to Sequencing and Scheduling*. New York: John Wiley & Sons, 1974.
28. J. W. Fowler, "Strategic control of multichannel bulk service diffusion/oxidation processes," Ph.D. dissertation, Texas A&M University, College Station, Dec. 1990. Also available as SRC Publication T91016.

Author Biographies

Jennifer Robinson is co-founder and chief operating officer of FabTime Inc., a provider of wafer fab cycle time reduction software and services. Previously, she worked as an independent consultant in semiconductor manufacturing. Her clients in that industry have included SEMATECH, Digital Equipment Corporation, Seagate Technology, and Siemens AG. Jennifer holds a B.S. (1989) degree in civil engineering from Duke University and an M.S. (1992) degree in Operations Research from the University of Texas at Austin, and a Ph.D. degree in Industrial Engineering from the University of Massachusetts at Amherst (1998). Her research interests center on factory productivity measurement and improvement.

John Fowler is Assistant Professor in the Department of Industrial and Management Systems at Arizona State University. He has also worked extensively in the semiconductor industry, as a member of the Operation Modeling Department at SEMATECH. His research interests are in the area of stochastic models for manufacturing systems.

Jonathan Bard is Professor of operations research and industrial engineering at The University of Texas at Austin. His research interests are the design and analysis of manufacturing systems, hierarchical optimization, and the development of algorithms for large-scale mixed integer programs.

```
If the tube becomes idle, then  
    if there are customers in the queue, then  
        let  $t_0$  be the time epoch that the tube becomes idle;  
        start decision heuristic (described in the text);  
    else  
        wait until a lot arrives and let  $t_0$  be the arrival epoch;  
        start decision heuristic (described in the text);  
    endif  
endif
```

Figure 1. Glassey and Weng's Dynamic Batching Heuristic

PUSH LOGIC:

Evaluate with NACH

PULL LOGIC:

If full loads are available, **then**

use WSPT (described in the text);

else

evaluate each product with NACH;

if all products should wait, **then**

wait;

if all products should start now, **then**

use WSPT;

else

calculate:

total delay caused by starting a batch of product j now for those products whose NACH evaluation indicates to start now (see eq. 3);

total delay caused by waiting for the next arrival of product j for those products whose NACH evaluation indicates to wait (see eq. 4);

select minimum and take appropriate action;

Figure 2. Fowler, Phillips and Hogg's Next Arrival Control Heuristic

Step 0: Set $CR(t_i) = \infty$ for all i .

Set $i = 0$.

Step 1: Calculate $CR(t_i)$ (see equations 5 and 6).

Set $i = i + 1$.

If $(q + i) \leq C$, **then**

go to Step 1;

else

go to Step 2;

endif

Step 2: Stop. Load $(q + n)$ lots of the product into the furnace at time t_n , where $n = \arg \min_k \{CR(t_k): 0 \leq k \leq (C-q)\}$.

Figure 3. Weng and Leachman's Heuristic for Single Products

Step 0: Set $CR_j(t_{i,j}) = \infty$, for all i and j .
Set $\mathbf{Q} = \emptyset$, $j = 1$, and $i = 0$.

Step 1: **If** ($q_j \geq C$), **then**
 calculate $CR_j(t_{0,j})$ (see equations 7-10);
 set $\mathbf{Q} = \mathbf{Q} \cup \{j\}$, $j = j + 1$; **go to** Step 1
endif

Step 2: **If** ($q_j + i \leq C$), **then**
 calculate $CR_j(t_{i,j})$ (see equations 7-10);
 set $i = i + 1$; **go to** Step 2
else
 $CR_j^{\min} = \min_i \{CR_j(t_{i,j}) : 0 \leq i \leq (C - q_j)\}$,
 $\text{index}(j) = \arg \min_n \{CR_j(t_{n,j}) = CR_j^{\min}\}$;
 set $j = j + 1$, $i = 0$;
 if $j \leq J$, **then**
 go to Step 1
 else
 go to Step 3
 endif
endif

Step 3: **If** ($\mathbf{Q} = \emptyset$) **then**
 load product $j = \arg \min_k \{CR_k^{\min} : k \in \mathbf{P}\}$ at time $t_{\text{index}(j),j}$;
else
 load a full batch of product $j = \arg \min_k \{CR_k^{\min} : k \in \mathbf{P}\}$ at $t_{0,j}$;
endif

Figure 4. Weng and Leachman's Heuristic for Multiple Products

Step 0: Set $CR(t_i) = \infty$ for all i .

Set $i = 0$.

Step 1: Calculate $CR(t_i)$ (see equations 5 and 6).

Set $i = i + 1$.

If $(q + i) \leq C$, **then**

go to Step 1;

else

go to Step 2;

endif

Step 2: Stop.

Set $n = \arg \min_k \{CR(t_k) : k \in \mathbf{P}\}$.

If $n = 0$ **then**

If $0 < q \leq C$ **then**

load q units of product at time t_0 ;

else if $q > C$ **then**

load C units of product at time t_0 ;

endif

else

wait for the next arrival;

go to Step 0;

endif

Figure 5. Robinson, Fowler, and Bard's Heuristic for Single Products

AVERAGE AT	MBS	NACH	MCR	RHCR	OVERALL AVG.
TI=0.1	0.2252	0.1599	0.1773	0.1652	0.1819
TI=0.2	0.3655	0.2232	0.2324	0.2224	0.2609
TI=0.3	0.4379	0.2703	0.2742	0.2664	0.3122
TI=0.4	0.4803	0.3113	0.3116	0.3056	0.3522
TI=0.5	0.5141	0.3515	0.3516	0.3463	0.3908
TI=0.6	0.5579	0.4017	0.4015	0.3973	0.4396
TI=0.7	0.6301	0.4794	0.4812	0.4776	0.5171
TI=0.8	0.7855	0.6339	0.6382	0.6358	0.6734
TI=0.9	1.2651	1.1118	1.1103	1.1067	1.1485
k=0.0	0.5847	0.4217	0.4166	0.4168	0.4599
k=0.3	0.5846	0.4545	0.4675	0.4551	0.4904
OVERALL AVG.	0.5846	0.4381	0.4421	0.4359	0.4752

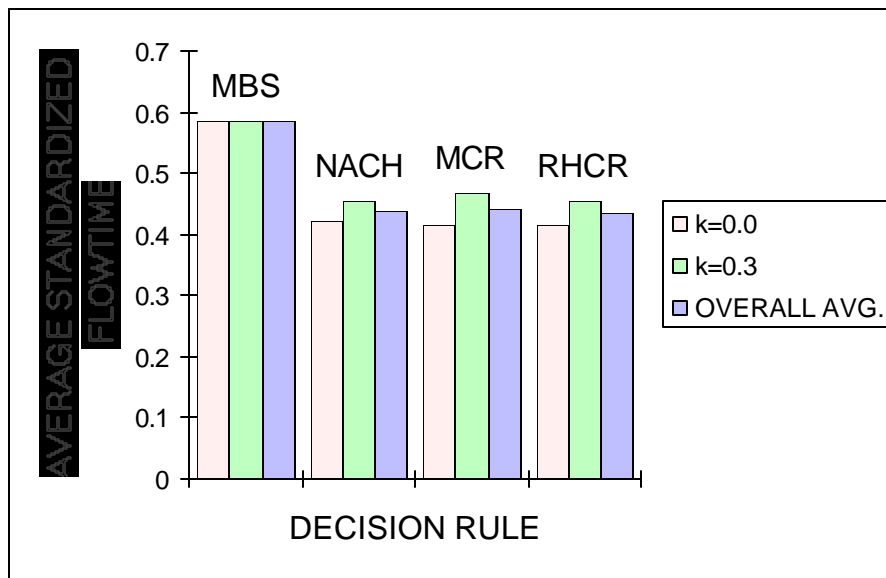


Figure 6. Average Standardized Flowtimes over all Treatment Combinations for Single Products, Using MBS, NACH, MCR, and RHCR (from Robinson *et al.* 1993).

Step 0: Set $CR_i(t_{i,j}) = \infty$, for all i and j .
Set $\mathbf{Q} = \emptyset$, $j = 1$, and $i = 0$.

Step 1: **If** ($q_j \geq C$), **then**
calculate $CR_i(t_{0,i})$ (see equations 7-10);
set $\mathbf{Q} = \mathbf{Q} \cup \{j\}$, $j = j + 1$; **go to** Step 1
endif

Step 2: **If** ($q_j + i \leq C$), **then**
calculate $CR_i(t_{i,j})$ (see equations 7-10);
set $i = i + 1$; **go to** Step 2
else
 $CR_{j,\min} = \min_i \{CR_i(t_{i,j}) : 0 \leq i \leq (C - q_j)\}$; (minimum CR for j)
 $\text{index}(j) = \arg \min_n \{CR_n(t_{n,j}) = CR_{j,\min} : 0 \leq n \leq (C - q_j)\}$;
set $j = j + 1$, $i = 0$;
if $j \leq J$, **then go to** Step 1
else go to Step 3
endif
endif

Step 3: **If** ($\mathbf{Q} = \emptyset$) **then**
 $j = \arg \min_{k \in \mathbf{P}} \{CR_{k,\min}\}$; (product with min CR)
If, $\text{index}(j) = 0$, **then** (if the min occurs at $t_{0,j}$)
load all available product j at $t_{0,j}$;
else
wait for the next arrival; **go to** Step 0
endif
else
 $j = \arg \min_{k \in \mathbf{Q}} \{CR_{k,\min}\}$,
load a full batch of product at time t_0 ;
endif

Figure 7. Robinson, Fowler, and Bard's Heuristic for Multiple Products

AVERAGE AT:	MBS	NACH	MCR	RHCR	OVERALL AVG.
TI=0.5	1.6305	1.3216	1.3552	1.3284	1.4089
TI=0.8	2.9459	2.7658	2.7266	2.7352	2.7934
CAPACITY=HIGH	2.3531	2.1626	2.2164	2.2049	2.2342
CAPACITY=SAME	2.2302	1.8079	1.8321	1.8061	1.9191
CAPACITY=LOW	2.2814	2.0996	2.1162	2.0845	2.1454
TIME=HIGH	2.3254	2.1626	2.2164	2.1769	2.2203
TIME=SAME	1.8222	1.7402	1.7185	1.7041	1.7462
TIME=LOW	2.7169	2.1695	2.2466	2.2145	2.3369
MIX=DOMINANT	2.2443	1.9733	2.0112	1.9839	2.0532
MIX=EQUAL	2.3321	2.0751	2.1098	2.0797	2.1491
k=0.0	2.2877	2.0153	2.0162	2.0102	2.0823
k=0.3	2.2887	2.0331	2.1048	2.0534	2.1199
OVERALL AVG.	2.2882	2.0241	2.0605	2.0318	2.1012

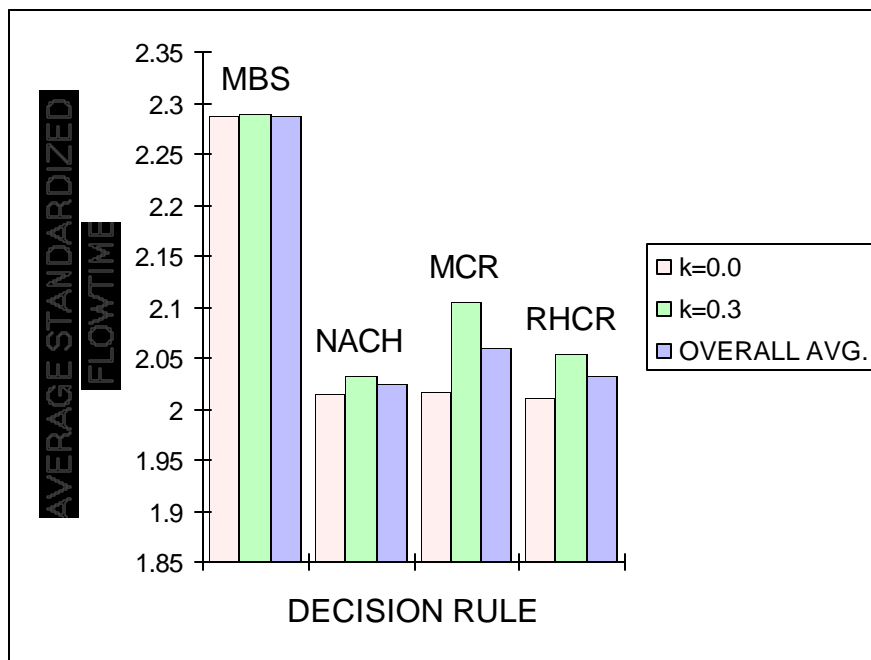


Figure 8. Average Standardized Flowtimes over all Treatment Combinations for Multiple Products, Using MBS, NACH, MCR, and RHCR (from Robinson *et al.* 1993).

	Factor
1.	Control strategy (MBS or NACH).
2.	Number of products (2 or 5).
3.	Interarrival time distribution (exponential or uniform).
4.	Product mix (equal mix or some products dominate).
5.	Capacity by product (all same, dominated by products with high capacity, or dominated by products with low capacity)
6.	Processing time by product (all same, dominated by products with high processing times, or dominated by products with low processing times)
7.	Traffic intensity (0.50 or 0.80, as defined in (1))

Table 1. Factors Used to Compare NACH and MBS, for Multiple Products

Control Strategy	Use Of Real-Time Information?	Approach	Rolling Horizon?	# Future Arrivals Considered
MBS	No	If less than specified minimum batch size is waiting, wait; else, make a batch of up to the specified maximum.	No	0
DBH	Yes	Choose the arrival with the minimum overall delay using areas under the curve of net delay.	No	L (up to C-1)
NACH	Yes	Consider next arrival. If processing now causes less delay, process now, else, wait for the next arrival and repeat the decision process.	Yes	1
MCR	Yes	For each arrival compute the total delay and divide by the time window to get the cost rate. Select the loading condition with the minimum cost rate and wait for that arrival.	No	Up to full loads
RHCR	Yes	Compute the cost rate for all arrivals. If the minimum is for the present time, process now. Otherwise wait for the next arrival and repeat decision process. This heuristic has also been extended to include downstream information.	Yes	Up to full loads

Table 2. Control Strategies for the Batch-Size Decision.