Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

ONLINE CONTROL OF A BATCH PROCESSOR WITH INCOMPATIBLE JOB FAMILIES UNDER CORRELATED FUTURE ARRIVALS

John Benedict C. Tajan
Appa Iyer Sivakumar

N3.2-01-36
65 Nanyang Drive
Singapore-MIT Alliance
(Department of Mechanical and Aerospace Engineering, Nanyang Technological University)
SINGAPORE 637460

Stanley B. Gershwin

Room 8-407
77 Massachusetts Avenue
Singapore-MIT Alliance
(Mechanical Engineering Department, Massachusetts Institute of Technology)
Cambridge, MA 02139, USA

ABSTRACT

The oxidation and diffusion ovens in wafer fabrication are batch processors, where only jobs belonging to identical job families can be processed together. In this paper, we compare the performance of a proposed online heuristic based on Model Predictive Control against a popular look-ahead method called NACHM. Simulation results show that the MPC-based heuristic, with properly selected parameters, can have up to 16.67% shorter mean cycle time than NACHM under uncorrelated job arrivals.
Under positively correlated job arrivals, the mean cycle time of jobs passing through the batch processor is almost always significantly reduced for both the MPC-based policy and NACHM. The simulation results also suggest that increased correlation generates less improvement for policies that foresee events longer into the future, as NACHM improves at a faster rate than the MPC-based heuristic. Thus, when the correlation is sufficiently high (0.7) and the traffic intensity is low (0.5), the MPC-based heuristic, which considers events that occur farther into the future, has higher mean cycle time (from 1.92% to 9.47%) than NACHM.
Controlling processors in front of the batch processor with the anticipated needs of the batch processor, successive job arrivals to the batch processor may result in positively correlated job families. Our results highlight the potential benefits of constraining the production of the upstream processor according to the anticipated needs of the batch processor.

1 INTRODUCTION

Both the oxidation and diffusion furnaces are batch processors that can concurrently process more than one job, with the processing time independent of the number of jobs processed. However, not all jobs arriving at the batch processor require identical chemical recipes and furnace temperatures. Thus, not all jobs can be processed together as a batch. Jobs that can be processed together comprise a job family.

When the number of jobs to be processed is finite, minimizing the mean cycle time of jobs passing through a single batch processor is an NP-hard problem (Tajan 2008). Thus, online heuristics meant for infinite horizon problems can also be used for finite horizon problems where the number of jobs to be scheduled is moderately large. In this paper, we compare a Model Predictive Control-based heuristic with NACHM, a popular look-ahead heuristic, under varying levels of correlation between the job families of consecutive arrivals.

2 ONLINE CONTROL OF BATCH PROCESSOR IN WAFER FABRICATION – A REVIEW

In this paper, we only consider the online control of batch processors where jobs belonging to different job families cannot be processed together. The processing time is only dependent on the job family being processed, and not on the job composition or quantity.

(Deb and Serfozo 1973) use dynamic programming formulations for minimizing the average cost per unit time and the expected discounted cost of a batch processor with stochastic processing times and Poisson job arrivals to show the existence of an optimal policy in the form of a minimum batch size. When the holding cost is linear and the processing time distribution is exponential, the optimal minimum batch size can be determined via closed-form equations. Follow-up work by (Aalto 1998 and Aalto 2000) consider compound Poisson arrival processes. When the processing time and arrival rate distributions are general, the optimal policy is no longer guaranteed to be a threshold policy. However, (Avramidis, Healy, and Uzsoy 1998) provide a method to determine the optimal threshold policy (the optimal policy among all threshold policies).

An alternative to threshold policiea are look-ahead policies, which are heuristics that assume that a limited number of future job arrivals can be predicted. Dynamic Batching Heuristic (DBH), by (Glassey and Weng 1991), assumes a single job family, with processing time P. At any Time Instance t that the batch processor is available and only a partial batch is available, DBH is activated. DBH assumes a planning horizon from t to $t+P$. Given the forecasted job arrivals within this horizon, DBH starts a batch at the time instance t' within the interval (t, $t+P$) that minimizes the total waiting time incurred by jobs at the batch processor from t to t'.

Simulation experiments show that DBH outperforms threshold policies, even with moderate errors in the predicted arrival time, as long as the traffic intensity is not low.

(Fowler, Phillips, and Hogg 1992) suggests a similar look-ahead called the next arrival control heuristic (NACH). NACH only considers the first future job arrival. Furthermore, NACH does not specify a future time instance when the batch processor is to be loaded. Rather, NACH decides to wait for the next job arrival, where the process is repeated. When there are moderate prediction errors, NACH slightly outperforms DBH. (Fowler, Phillips, and Hogg 1992) also extend NACH to the case where jobs belong to different job families (we call this extension NACHM). Future knowledge of the first job arrival for each job family is assumed to be known.

NACHM outperforms threshold policies, even under moderate prediction errors. However, the amount of improvement declines as the prediction errors become large. NACHM has been extended to two cases: when there is more than one batch processor in a particular stage (Fowler, Hogg, and Phillips 2000), and when the downstream processor requires considerable set-up times. (Solomon, et al. 2002).

Subsequent look-ahead heuristics include (Weng and Leachman 1993), which assume jobs have a unit holding cost, and propose the Minimum Cost Rate (MCR) heuristic in an attempt to minimize the rate cost is incurred by the batch processor. (Weng and Leachman 1993) propose a minimum cost rate (MCR) heuristic for the same problems in (Fowler, Phillips, and Hogg 1992). MCR, like DBH, does not postpone decisions; it instead determines the best time to process a batch at each instance it is executed. (Robinson, Fowler, and Bard 1995) propose the Rolling Horizon Cost Rate (RHCR) heuristic, which is a combination of the cost rate calculations of MCR and the decision postponement of NACHM. (Robinson, Fowler, and Bard 1995) is also expanded to form the RHCR-S heuristic, which considers the expected waiting time of jobs in front of the downstream serial processor.

(Duenyas and Neale 1997) analyze the problem of minimizing the average holding cost per unit time of a single batch processor with incompatible job families; job families have exponential processing times. Even when there are only two job families, each with a Poisson arrival process, the optimal policy can have a complicated form and a heuristic policy is proposed. This heuristic can be easily adapted to reflect knowledge of future arrivals.

The previously mentioned look-ahead heuristics consider only cycle time-related objective functions; (Gupta, Sivakumar, and Ganesan 2004) apply look-ahead method to optimizing earliness/tardiness-related objectives. Instead of determining analytically the optimal time instance and batch composition, the authors propose using Conjunctive Simulated Scheduling to evaluate the options available to the idle batch processor.

3 PROBLEM STATEMENT

A batch processor can process up to Q jobs simultaneously, and the processing time is independent of the number of jobs being processed. There are n jobs to be processed at the batch processor, where n is unknown. The earliest time a Job i can be processed is at Time Instance r_i. Each Job i belongs to a Job Family j, where $j = 1$ to m. Only jobs belonging to the same family can be batched together. The processing time of a batch is dependent only on the job family that is currently being processed. The objective is to minimize the mean cycle time for all jobs passing through the system, with the cycle time of Job i equal to its Completion Time c_i minus its Arrival Time r_i. . The batch processor is assumed to be initially available, and jobs are indexed in increasing order of their release times.

4 ONLINE CONTROL OF A BATCH PROCESSOR

Because the number of jobs n is unknown, the number of jobs processed by the batch processor is uncertain. We require an online algorithm, which is executed regularly, with only a limited number of decisions made at each execution. Online algorithms typically have to run much faster than offline algorithms, as online algorithms are repeatedly executed in real time.

In this section, we discuss two online algorithms: the MPC-based heuristic we develop, and NACHM, the benchmark we compare the MPC-based heuristic against.

4.1 NACHM Control Scheme

In NACHM, a decision has to be made when (a) a job arrives, or (b) the batch processor finishes. When a job arrival corresponding to Family j occurs, the 'push logic' is used. In the 'push' logic, only jobs belonging to Family j are considered. If the total waiting time incurred if a batch of Family j jobs are immediately processed is higher than the waiting time incurred if the processor waits for the next arrival of Family j job, then the processor waits. If the re-

verse is true, then the processor processes a batch of Family *j* jobs.

When the batch processor finishes processing, the 'pull logic' is used. The 'pull' logic is detailed below:

- If there is at least one full batch, then the full batch with the weighted shortest processing time (WSPT) is chosen for processing. The weight of a particular Job Family *i* is the total number of jobs in front of the batch processor that does not belong to Job Family *i.*
- If only partial batches exist, then the 'push' logic is executed for each job family.
- If the 'push' logic returns a decision to process for every job family, then WSPT is used to select which job family to process. Conversely, if the 'push' logic suggests waiting for the next job arrival for all job families, the system waits for the next job arrival.
- If the decisions from the 'push' logic functions are not unanimous for all job families, then the total waiting time incurred by all jobs if the recommended decision of the 'push' logic is followed for that job family is computed. The corresponding decision that results with the lowest total waiting time is the output of the 'pull' logic.

We choose NACHM as a benchmark for several reasons:

- The logic behind NACHM is easy to understand.
- The definition of job arrival horizons of NACHM and the MPC-based heuristic differ. NACHM assumes knowledge of *m* job arrivals, one job arrival for each family. In contrast, the MPC-based algorithm takes, at most, the next *L* job arrivals, regardless of their job family.
- NACHM considers all job families in making a decision. In contrast, the MPC-based algorithm may consider only a subset of job families.
- NACHM only considers the waiting time incurred due to the first batch, while the MPC-based heuristic considers the waiting time incurred in emptying the batch processor queue and horizon.

4.2 Model Predictive Control (MPC)

One possible method of coping with algorithm processing time constraints is Model Predictive Control (MPC) (Bertsekas 2005). At each instance a decision has to be made, model predictive control optimally solves a deterministic problem with a shorter horizon. This will output a series of controls, one control for each instance a decision has to be made. Only the first control is implemented, the rest are discarded. This process is repeated at each instance a decision has to be made.

To implement MPC, a base method of solving small instances of the problem is required. (Tajan 2008) provides two base methods for solving small problem instances: an integer linear programming model and a dynamic programming model. Either model can be easily modified to be used as a kernel for MPC.

4.3 Development of MPC-based heuristic

From numerical experiments, an increase of five job arrivals or one job family corresponds to roughly an order of magnitude of increase in the computational time required to solve a finite- horizon problem to optimality, using the dynamic programming model in Tajan (2008). Thus, MPC needs to truncate the problem in two dimensions to create the smaller problem instance: the number of future job arrivals L and the number of job families f. Thus, the MPC-based heuristic has two parameters and the exact variant can be distinguished via the parameters *(f, L)*.

At any Instance *t* that the batch processor is idle, MPC predicts the arrivals for the next *L* jobs into the batch processor. Assuming that the batch processor can process up to $\chi > f$ job families, then $\chi - f$ job families are ignored by the MPC-based heuristic. Only the *f* job families whose batches have the shortest weighted processing times (given the current queue composition at the batch processor) are considered. This selection criterion is based on the optimal policy for a single batch processor with no future arrivals, which dictate that batches are to be processed according to the weighed shortest processing time rule (with the number of jobs inside a batch serving as the batch weight).

Let the set of job families selected for consideration be S_f. If, at Time Instance *t*, only $\psi < f$ job families have jobs in front of the batch, then the ψ job families with jobs queued join S_f. We then augment S_f by counting the number of jobs belonging to a particular job family within the *L* future job arrivals. Let f_i be the job family with the i^{th} most jobs arriving within the horizon. Starting from $i=1$, if $f_i \notin S_f$, then $S_f = S_f \bigcup f_i$, and the process continues until $i=m$, or until $|S_f| = f$, whichever comes first. Thus, the smaller problem instance to be solved to optimality has $|S_f| \leq f$ job families and a maximum of *L* future job arrivals.

5 COMPARISON OF CONTROL SCHEMES FOR DIFFERENT LEVELS OF CORRELATION

In this section, we discuss the manner in which correlation between successive job arrivals is introduced, before introducing the experimental setup.

5.1 Experimental model

Figure 1 shows the chosen experimental model. A single batch processor with a maximum capacity of *Q* jobs at a time stores jobs in $m = 6$ buffers, each buffer corresponding to a job family. The batch processor experiences a single job arrival stream; thus, the time between successive

job arrivals can be described by a single random distribution. We assume the job arrival stream is a Poisson process; the time between successive job arrivals follow an exponential distribution.

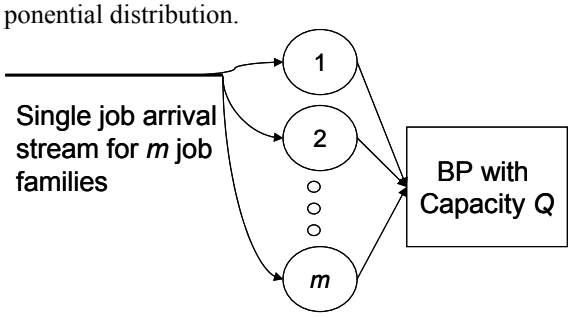

Figure 1: The batch processor BP can process up to Q jobs belonging to the same family, and experiences a single random job arrival stream.

5.2 Correlation between job families of successive job arrivals

When upstream processors do not care about the family sequence of job arrivals to the batch processor, we assume zero correlation between the job families of successive job arrivals. Thus, the job family of a particular job arrival has a uniform distribution.

Positive correlation between the job families of successive arrivals implies that it is likely to find consecutive job arrivals to have identical job families. Let there be m job families. The probability that the job family of x, a_x, is $1 \le z \le m$ equals $1/m$, since a_x is uniformly distributed from 1 to z. Let y be the job that arrives immediately after x. Let $\psi = P(a_y = z | a_x = z)$. ψ is the probability that y belongs to Family z if x belongs to Family z. $P(a_y = g | a_x = z, g \neq z)$ has a uniform distribution across the remaining job families, and is equal to $(1-\psi)/(m-1)$. The correlation coefficient between the job families of two consecutive jobs (represented as random variables X and Y) is related to ψ and m by

$$Corr(X,Y) = \frac{m\varphi - 1}{m-1}.$$

5.3 Experimental Setup

Two variants of the MPC-based heuristic *(f, L) – f* being the job family quantity considered and *L* being the job arrival horizon length - are evaluated, (3, 15) and (4, 10). Based on empirical experimentation, an increase in one family or in five jobs results in an increase in magnitude of the computational time requirement for the Dynamic Programming model used by the MPC-based heuristic (Tajan 2008). Thus, the heuristics (3, 15) and (4, 10) have roughly the same magnitude in computational requirements (based on a MATLAB implementation). These two heuristics are compared against NACHM.

For each simulation run, 2000 jobs are randomly generated, the cycle time from the first 200 jobs to exit the system are discarded. This warm-up period is deemed sufficient using Welch's method (Law and Kelton 2000). The processing time per job family is evenly distributed between three to nine time units. The mean time between job arrivals is determined through the traffic intensity. The traffic intensity *TI* is the dimensionless ratio between the arrival rate and the maximum processing rate. A ratio above 1.0 indicates that jobs arrive at a faster rate than the processor can process, which will lead to system instability. We assume two values for the traffic intensity, 0.5 and 0.8. If λ is the average arrival rate, then the mean time between arrivals, $1/\lambda$, is obtained by:

$1/\lambda$ = (average time to process a batch/(traffic intensity * batch processor capacity)).

We assume two values (four and eight) for the job family quantity. Three levels of correlation coefficients were selected: zero correlation, weak correlation (Corr(X,Y)=0.25), and strong correlation (Corr(X,Y)=0.7).

For each set of simulation parameters, ten simulation experiments are performed. To increase testing power, all three evaluated heuristics share the same arrival stream. Table 1 summarizes the various levels varied for each set of experiments.

Table 1: Experimental Setup Summary

Corr(X,Y)	Job family quantity	Traffic intensity
0, 0.25, 0.7	4, 8	0.5, 0.8

6 DISCUSSION OF RESULTS

In each set of experiments, a paired T-test is used to determine whether the hypothesized differences in the mean cycle time of the proposed MPC-based heuristics and NACHM is significantly different from zero. The confidence interval selected is 95%.

The mean execution time of a single instance of any heuristic does not exceed 10 seconds. However, both MPC-based heuristics take considerably longer times than the NACHM heuristic, due to the need to solve a small dynamic programming problem at each instance.

6.1 Uncorrelated Job Families for Consecutive Job Arrivals

Figure 2 graphically presents the mean cycle time estimates for the three heuristics when job arrivals are uncorrelated. Increasing either the number of job families or the traffic intensity increases the mean cycle time, regardless of policy. Both observations are intuitive: increasing the traffic intensity increases the workload of the processor, while increasing the number of job families mean that job composition of a batch become more restrictive.

In directly comparing (3,15) with (4,10), (3,15) has a significantly longer mean cycle time than (4,10) when there are only four job families. When the number of job families is increased to eight, the performance of the two policies cannot be differentiated with 95% confidence level.

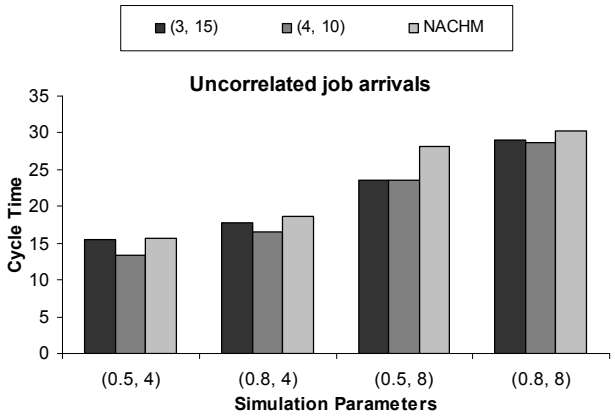

Figure 2: Comparing cycle time for three heuristics when job families of consecutive job arrivals are uncorrelated. The simulation parameters are described as tuples, in the form (TI, job family quantity).

When there are only four job families, (3, 15) removes one job family from consideration. This can adversely affect performance. As an example, assume that Job Families One, Two and Three each have a single job in front of the batch processor, and Job Family Four has none. However, the next three job arrivals belong to Job Family Four, and these three jobs will all arrive very soon. (3, 15) might recommend processing a small batch of either Family One, Two or Three, whereas (4, 10) correctly decides to wait for the next three arrivals of Family Four jobs.

When the job family quantity is increased to eight, the effect of one less job family considered is reduced. Firstly, the probability that a job arrival sequence has a dominant job family is reduced with a larger number of job families. Secondly, the probability that a crucial job family is ignored by (3,15) and is not ignored by (4,10) (Event A) is 25% with four job families and 12.5% with eight job families.

Based on comparing (3, 15) and (4, 10), we recommend the following guidelines in selecting the parameters L and f for the MPC-based heuristic:

- If the number of job families is relatively small, it is best to select f to be equal to the number of job families, at the cost of reducing L or relaxing the computational time constraints.
- If the number of job families is large, then small reductions in f will not have a large adverse effect on the performance of the heuristic.

In comparing NACHM against both MPC-based heuristics (3, 15) and (4, 10), the mean cycle time for NACHM is significantly longer than either MPC-based heuristic for all four parameter sets. Table 2 contains the estimated percentage reduction in cycle time if the control scheme is switched from NACHM to either MPC-based heuristic. The results suggest that the MPC-based heuristic outperforms NACHM when the job families of consecutive job arrivals are uncorrelated.

Table 2: Paired T-test results for uncorrelated job arrivals

Traffic Intensity	Job Family Quantity	Estimated % cycle time reduction - (3, 15)	Estimated % cycle time reduction - (4, 10)
0.8	8	4.45%	5.38%
0.8	4	4.2%	11.69%
0.5	8	16.66%	16.19%
0.5	4	1.21%	13.84%

To be sure that the improvement of the MPC-based heuristics are not due to their possibly longer horizons, we perform an auxiliary comparison between the MPC-based heuristic (4,8) and NACHM when there are eight job families. For each traffic intensity (0.5 and 0.8), ten simulation runs are performed. There does not exist an instance where (4,8) will have a longer horizon (in terms of the number of job arrivals) than NACHM. Since the actual number of job families running through the system is twice that of the number of job families considered by (4,8) at any instance, the mean number of future job arrivals considered by (4, 8) is four, half that of NACHM. However, Table 3 shows that (4, 8) still has significantly lower mean cycle time than NACHM. When the traffic intensity is low, (4, 8) has 16.9% lower mean cycle time, and when the traffic intensity is high, 5.17% lower mean cycle time. This experiment shows that the improved mean cycle time by the MPC-based heuristics cannot be wholly attributed to a possibly longer horizon length than NACHM.

Table 3: Comparing (4, 8) with NACH-MM when there are eight job families

Traffic Intensity	$\mu_{(4,8)}$	μ_{NACHM}	$\mu_{NACHM} - \mu_{(4,8)}$	95% Confidence interval
0.5	23.58	28.39	4.81	(4.35, 5.37)
0.8	29.32	30.92	1.6	(0.72, 2.48)

To the best of the authors' knowledge, all of previously proposed look-ahead methods 'optimize' only the waiting time incurred due to the next batch. By ignoring the remaining batches that need to be processed by the batch processor, the previously proposed look-ahead policies do not fully utilize the available information. For example, if the current batch processor queue has jobs from three job families (Figure 3), there must be at least four

batches that need to be processed by the batch processor, before the queue is emptied. The MPC-based heuristic will consider the cycle time incurred for processing all three batches, while NACHM (and other look-ahead methods) merely look at the cycle time incurred for the first batch to be processed. We believe that the MPC-based heuristic has improved performance over NACHM because of this further forward thinking.

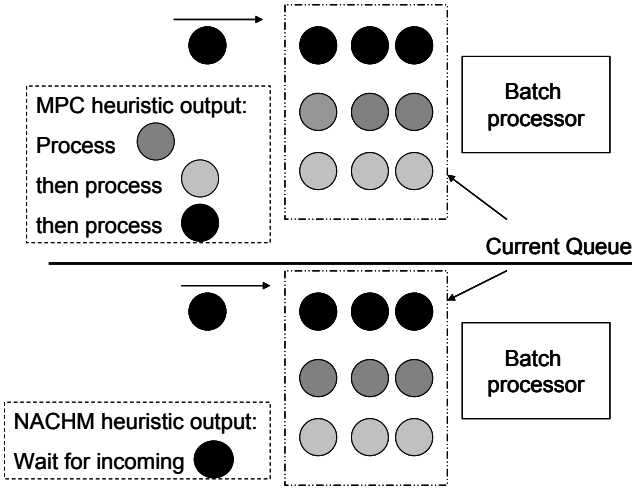

Figure 3: The difference between the two policies is that the MPC-based heuristic will provide a complete sequence to empty out the buffer, while NACHM (and other look-ahead methods) will only consider the best decision with regards to only the first batch to be processed.

6.2 Correlated Job Families for Consecutive Job Arrivals

We divide the discussion into two sections. First, we discuss the effect of increased correlation on the *relative* performances of the heuristics, then we discuss the effect of increased correlation on the absolute performances of the heuristics.

6.2.1 Relative performance of heuristics

Figure 4 and Figure 5 illustrate the mean cycle times of the three heuristics under increased correlation of job families for successive job arrivals. (3,15) still has significantly higher cycle time than (4,10) when there are only four job families. When there are eight job families, the mean cycle times generated by (3,15) and (4,10) generally cannot be distinguished within 95% confidence interval. The only exception is when the correlation coefficient is high (0.7), and the traffic intensity was low (0.5), where (4,10) has superior performance to (3,15). High correlation coefficients increases the probability that consecutive job arrivals will be from the same job family. If (3,15) chooses to consider only Job Families One, Two and Three, if the next four job arrivals all belong to Job Family Four, the heuristic is tricked into thinking that there are no job arrivals in the near future. This might lead to the heuristic recommending that the batch processor should process a batch, even if waiting for the four jobs from Job Family Four to arrive reduces average cycle time. Increasing *f* from three to four increases the probability that some job families would be selected according to future job arrivals.

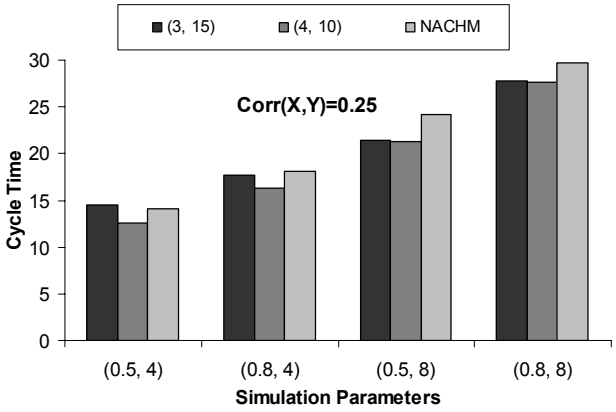

Figure 4: Estimated mean cycle time for the three heuristics when Corr (X,Y) = 0.25. NACHM has lower cycle time than (3,15) for setting (0.5, 4).

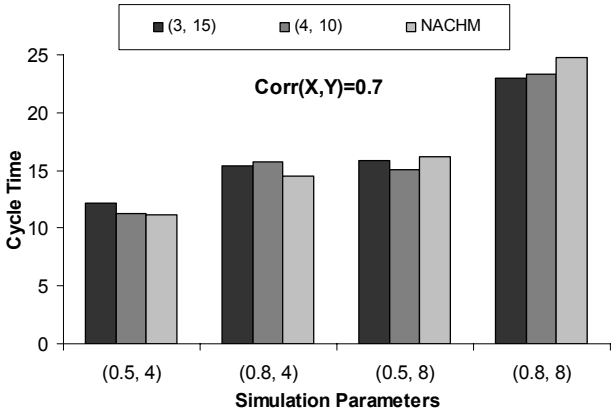

Figure 5: Estimated mean cycle time for the three heuristics when Corr (X,Y) = 0.7. NACHM has lower cycle time than either MPC-based heuristic at low traffic intensity.

However, the results also suggest that (3,15) benefits more than (4,10) from the introduced correlation when there are only four job families. This is due to the need of (3,15) to exclude one job family in its analysis. When Corr(X,Y) is low, the current WIP levels at the batch processor (a function of past job arrivals) are less representative of the future job arrivals expected, than when Corr(X,Y) is high. Since the MPC-based heuristic uses the current WIP levels to determine which job family to exclude from its analysis (for the case of (3,15)), high correlation benefits

(3,15) more than (4,10), as it makes (3,15) less likely to ignore a crucial job family in its analysis.

The relative performance of (4,10) (as measured by the percentage reduction in mean cycle time over NACHM) generally worsens as the correlation coefficient is increased. Furthermore, when the traffic intensity is 0.5 and Corr(X,Y)=0.7, NACHM has significantly lower cycle time than both MPC-based heuristics (3,15) and (4,10). This effect is magnified when there are more job families, as some job families are ignored by the MPC-based heuristics, whereas NACHM does not ignore any job family. Table 4 shows the expected percentage reduction in the mean cycle time when switching from NACHM to either MPC-based heuristic. A negative value indicates that switching to that particular MPC-based heuristic results in increasing the mean cycle time, while N.S. means we could not differentiate the mean cycle times with 95% confidence level.

Table 4: Estimated percentage cycle time reduction due to adoption of MPC-based heuristic over NACH-MM when job arrivals have positively correlated job families

Corr (X,Y)	Traffic Intensity	Job Family Quantity	Est. % reduction - (3, 15)	Est. % reduction - (4, 10)
0.25	0.8	8	6.24%	6.7%
0.25	0.8	4	2.13%	10.32%
0.25	0.5	8	11.70%	11.70%
0.25	0.5	4	**-2.59%**	11.19%
0.7	0.8	8	7.44%	5.97%
0.7	0.8	4	***N.S.***	6.6%
0.7	0.5	8	**-6.62%**	**-8.76%**
0.7	0.5	4	**-9.47%**	**-1.92%**

6.2.2 Absolute performance of heuristics

We hypothesize that increasing correlation reduces the mean cycle time, regardless of heuristic. To test this hypothesis, we use a T-test to determine the statistical significance of the observed reduction in mean cycle time due to the increase in correlation. A 95% confidence level is still used, and the null hypothesis is that the mean cycle times under job arrivals with varying levels of correlation in their job families are equal (H_0: $\mu_0 \neq \mu_1$). The alternative hypothesis is H_1: $\mu_0 > \mu_1$; we ignore the possibility that correlation will increase mean cycle time. Two sets of comparisons are performed: comparing uncorrelated with lowly correlated job families among successive job arrivals, and comparing lowly correlated with highly correlated job families among successive job arrivals. Table 5 contains the estimated reduction in mean cycle time if low correlation between the job families of successive job arrivals were introduced, Table 6 contains the estimated incremental reduction in cycle time if the correlation was increased to a high level. In both tables, positive values indicate increasing correlation resulted in reduced mean cycle time.

Table 5: Estimated reduction in mean cycle time as Corr(X,Y) moves from 0 to 0.25

Heuristic	TI	job family qty.	Est. % reduction in cycle time	Probability null hypothesis is true
(3, 15)	0.8	4	0.62%	**0.37**
(4, 10)	0.8	4	1.22%	**0.25**
NACHM	0.8	4	2.72%	2.97×10^{-2}
(3, 15)	0.8	8	3.85%	3.28×10^{-2}
(4, 10)	0.8	8	3.40%	4.64×10^{-2}
NACHM	0.8	8	2.02%	2.39×10^{-2}
(3, 15)	0.5	4	6.07%	2.49×10^{-8}
(4, 10)	0.5	4	6.78%	9.21×10^{-9}
NACHM	0.5	4	9.56%	1.56×10^{-11}
(3, 15)	0.5	8	9.20%	1.36×10^{-9}
(4, 10)	0.5	8	9.70%	2.9×10^{-8}
NACHM	0.5	8	14.29%	3.27×10^{-10}

Table 6: Estimated reduction in mean cycle time as Corr(X,Y) moves from 0.25 to 0.7

Heuristic	TI	job family qty.	Est. % reduction in cycle time	Probability null hypothesis is true
(3, 15)	0.8	4	10.40%	7.76×10^{-5}
(4, 10)	0.8	4	7.16%	2.56×10^{-3}
NACHM	0.8	4	10.86%	1.71×10^{-5}
(3, 15)	0.8	8	17.45%	5.7×10^{-8}
(4, 10)	0.8	8	15.72%	2.1×10^{-6}
NACHM	0.8	8	16.37%	1.59×10^{-7}
(3, 15)	0.5	4	15.86%	1.04×10^{-10}
(4, 10)	0.5	4	9.50%	2.98×10^{-8}
NACHM	0.5	4	21.14%	4.83×10^{-15}
(3, 15)	0.5	8	27.71%	1.08×10^{-15}
(4, 10)	0.5	8	26.24%	5.74×10^{-14}
NACHM	0.5	8	40.12%	3.19×10^{-17}

In general, the mean cycle time is significantly reduced when the amount of correlation between the job families of successive job arrivals is increased. The only cases where the T-test failed to find a significant difference were when the number of job families were low (four), the traffic intensity was high (0.8), and the correlation coefficient was increased from 0 to 0.25, for the two MPC-based heuristics. Furthermore, the cycle time reduction is more substantial (in terms of percentage reduction) when the base scenario (Corr(X,Y) = 0) has higher cycle time. Thus, the reduction in mean cycle time is greater when the number of job families is high or when the traffic intensity is low.

The results also suggest that NACHM will benefit more than the MPC-based heuristic. The introduction of correlation causes job arrivals for a particular job family to come in spurts. This alleviates the problem of estimating individual job family arrival rates when NACHM selects its job horizon, since it becomes more likely to see spurts of job arrivals with the same job family. In addition, the deterioration due to only optimizing the first batch to be processed is reduced. If the next X arrivals all belong to Job Family i, then it is likely that Job Family i would be the dominant factor in determining the control policy at the batch processor, whether we optimize the first batch, or until the queue is emptied. This effect is more pronounced when the traffic intensity is low, since the time instances between jobs are long. Thus, we get to see greater relative improvement for NACHM over the MPC-based heuristic when positive correlation is introduced. At high correlation coefficients and low traffic intensity, NACHM becomes superior to the MPC-based heuristics.

If it is possible to increase the job family correlation of successive job arrivals significantly, then the reduction in cycle time obtained through this method is generally significantly larger than the benefit of using a more sophisticated control policy. This suggests that controlling the arrival pattern of jobs into the batch processor can conceivably result in larger improvements in the batch processor performance, even with a simplistic batch processor control policy.

7 CONCLUSION

We develop an MPC-based heuristic for the infinite horizon problem of minimizing the mean cycle time of a batch processor with incompatible job families and future job arrivals. The finite horizon problem is strongly NP-Hard, making heuristics necessary even for problems with moderate size. The MPC-based heuristic *(f, L)* has two parameters: the number of job families considered f, and the number of future job arrivals L. When the total number of job families is low, letting f be equal to the number of job families, at the expense of shorter L or longer computational time per iteration, will improve heuristic performance significantly. When the total number of job families is high, the exact value of f becomes less crucial.

We compare the performance of the MPC-based heuristic (4,10) against a popular look-ahead heuristic, NACHM. When the job families of successive job arrivals are uncorrelated, the MPC-based heuristic has significantly lower mean cycle time than NACHM, especially when the job arrival rate is low.

When processors upstream of the batch processor intend to process jobs that can quickly form batches, the job families of arrivals the batch processor experiences may be positively correlated. Our experimental results show that increasing correlation generally causes a significant reduction in the mean cycle time observed, regardless of the batch processor heuristic. Furthermore, the experienced cycle time reduction is typically larger with more job families.

In comparing the improvement between the MPC-based heuristics and NACHM, the results support the hypothesis that increased correlation causes smaller improvements for policies that foresee events farther into the future. When the correlation is sufficiently high, (4, 10) may even have worse performance than NACHM, for a limited set of system parameters.

Our results suggest two ways the mean cycle time can be reduced at the batch processor. First, one can use the MPC-based heuristic, which outperforms NACHM for a large proportion of the system parameters evaluated. However, this heuristic entails larger computational costs, particularly for larger values of f and L. The second way to reduce cycle time is to control the upstream processors, such that the arrival distribution has positively correlated job families. Increasing correlation in the job families results in a significant reduction in mean cycle time for all system parameters evaluated, regardless of batch processor policy. Furthermore, for the correlation coefficients evaluated, the amount of cycle time reduction obtained from increasing correlation typically dwarfs the magnitude of the cycle time reduction obtained from switching to the MPC-based heuristic.

ACKNOWLEDGMENTS

The work presented is sponsored by the Singapore MIT Alliance (SMA).

REFERENCES

Aalto, S. 1998. Optimal control of batch service queues with compound Poisson arrivals and finite service capacity. *Mathematical Methods of Operations Research* 48:317-335.

Aalto, S. 2000. Optimal control of batch service queues with finite service capacity and linear holding costs. *Mathematical Methods of Operations Research* 51: 263-285.

Avramidis, A. N., K. J. Healy, and R. Uzsoy. 1998. Control of a batch-processing machine: a computational approach. *International Journal of Production Research* 36:3167-3181.

Bertsekas, D. P. 2005. *Dynamic programming and optimal control.* 3rd ed. USA: Athena Scientific.

Deb, R. K. and R. F. Serfozo. 1973. Optimal control of batch service queues. *Advances in Applied Probability* 5:340-361.

Duenyas, I. and J. J. Neale.1997. Stochastic scheduling of batch processing machine with incompatible job families. *Annals of Operations Research* 70:191-220.

Fowler, J. W., G. L. Hogg, and D. T. Phillips. 2000. Control of multiproduct bulk server diffusion/oxidation processes. Part 2: Multiple servers. *IIE Transactions* 32:167-176.

Fowler, J. W., D. T. Phillips, and G. L. Hogg. 1992. Real-time control of multiproduct bulk-service semiconductor manufacturing processes. *IEEE Transactions on Semiconductor Manufacturing* 5:158-163.

Glassey, C. R., and W. W. Weng. 1991. Dynamic batching heuristic for simultaneous processing. *IEEE Transactions on Semiconductor Manufacturing* 4:77-82.

Gupta, A. K., A. I. Sivakumar, and V. K. Ganesan. 2004. Look ahead batching to minimize Earliness/Tardiness measures in batch processes. In *2004 IEEE Conference on Robotics, Automation and Mechatronics* 2:1101-1106. Singapore: Institute of Electrical and Electronics Engineers, Inc.

Law, A. M. and W. D. Kelton. 2000. *Simulation modeling and analysis*. 3rd ed. New York: McGraw-Hill, Inc.

Mathirajan, M. and A. I. Sivakumar. 2006. A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. *International Journal of Advanced Manufacturing Technology* 29:990-1001.

Robinson, J. K., J. W. Fowler, and J. F. Bard. 1995. The use of upstream and downstream information in scheduling semiconductor batch operations. *International Journal of Production Research* 33:1849-1869.

Solomon, L., J. W. Fowler, M. Pfund, and P. H. Jensen. 2002. The inclusion of future arrivals and downstream setups into wafer fabrication batch processing decisions. *Journal of Electronics Manufacturing* 11:149-159.

Tajan, J.B. 2008. Control of manufacturing systems with downstream batch processor. PhD. Thesis, School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore.

Weng, W. W. and R. C. Leachman. 1993. An improved methodology for real-time production decisions at batch-process work stations. *IEEE Transactions on Semiconductor Manufacturing* 6:219-225.

AUTHOR BIOGRAPHIES

JOHN BENEDICT TAJAN is a post-doctoral fellow for the School of Information Systems for Singapore Management University. He received his B.S. in Manufacturing Engineering and Management from De La Salle University, Philippines, and both his S.M. and PhD. in Innovation in Manufacturing Systems and Technology from Nanyang Technological University, under the Singapore-MIT Alliance programme. His current research interests include control of systems with batch processors and decentralized methods for resource allocation.

APPA IYER SIVAKUMAR is an Associate Professor in the School of Mechanical and Aerospace Engineering (MAE) at the Nanyang Technological University, Singapore and a Faculty Fellow of Singapore - Massachusetts Institute of Technology (MIT) Alliance (SMA-MST programme) He was at Gintic Institute of Manufacturing Technology, Singapore prior to this appointment. His research interests are in the area of OR, Optimization, Advanced Manufacturing Systems engineering, Discrete Event Simulation, Scheduling, Logistics, Supply chain design, and Research Methodology. He is the Chairman of the NTU Undergraduate Research Programme (URECA) and the Chairman of the MAE Engineering Innovation and Design (EID) programme. He received a Bachelors of Engineering in Manufacturing Systems Engineering and a PhD in Manufacturing Systems Engineering from University of Bradford, UK. He has been the Technical Committee Chairman of ICCIM and co-edited the proceedings of the 3rd and 4th International Conference on Computer Integrated Manufacturing (ICCIM ’95 and ICCIM’97).

STANLEY B. GERSHWIN is a Senior Research Scientist at the MIT Department of Mechanical Engineering. He received the B.S. degree in Engineering Mathematics from Columbia University, New York, New York, in 1966; and the M.A. and Ph.D. degrees in Applied Mathematics from Harvard University, Cambridge, Massachusetts, in 1967 and 1971. He has been previously affiliated with the Bell Telephone Laboratories, the C. S. Draper Laboratory and MIT Laboratory for Information and Decision Systems (LIDS). He was Professor of Manufacturing Engineering at the Boston University College of Engineering (half time) in 1986-1987. Dr. Gershwin currently teaches an MIT course in Manufacturing Systems Analysis (2.852).

Dr. Gershwin is the author of Manufacturing Systems Engineering (Prentice-Hall, 1994) and numerous papers in international journals. His research interests include real-time scheduling and planning in manufacturing systems; hierarchical control; dynamic programming in hybrid (discrete and continuous state) systems; decomposition methods for large scale systems; approximation techniques.

Dr. Gershwin and his students have performed research projects and consulted for such companies as Boeing, General Motors, Polaroid, Hewlett Packard, Johnson & Johnson, and United Technologies. He is the MIT Group Leader of the Leaders for Manufacturing Program research Group 5, "Design and Operation of Manufacturing Systems." Dr. Gershwin is an IEEE Control Systems Society Distinguished Lecturer and a Fellow of the IEEE. He is affiliated with MIT's Laboratory for Manufacturing and Productivity, Leaders for Manufacturing Program, and the Operations Research Center.