Dynamic scheduling of batch processing machines with non-identical product sizes

Durk-Jouke van der Zee

Dynamic scheduling of batch processing machines with non-identical product sizes

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2005-IJPR-0476.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	14-Feb-2006
Complete List of Authors:	van der Zee, Durk-Jouke; University of Groningen, Production Systems Design Group
Keywords:	DISPATCHING RULES, SIMULATION, BATCH SCHEDULING
Keywords (user):	

Dynamic scheduling of batch processing machines with non-identical product sizes

D.J. van der Zee

Assistant Professor
Production Systems Design Group
Faculty of Management and Organization
University of Groningen
P.O. Box 800
9700 AV Groningen
The Netherlands

Phone: +31 – 50 - 363 4687 / +31 – 50 - 363 7491
Fax: +31 – 50 – 363 2032
e-mail: d.j.van.der.zee@rug.nl

Abstract

In this paper we consider dynamic scheduling for batch processing machines. Our research is motivated by the burn-in ovens found in semiconductor manufacturing. So far research in this field mainly concentrated on control strategies that assume batches to be homogeneous, i.e., products should all belong to the same family. However, burn-in ovens may allow for simultaneous processing of alternative families of products. Families differ from each other with respect to product volume. We propose a new scheduling approach that addresses these situations. The objective is to minimize average flow time per product for the batch operation. Our so-called look-ahead strategy adapts its scheduling decision to shop status, which includes information on a limited number of near future arrivals. The potential of the new strategy is demonstrated by an extensive simulation study.

Key words

Scheduling, batch processing machines, simulation

Dynamic scheduling of batch processing machines with non-identical product sizes

Abstract

In this paper we consider dynamic scheduling for batch processing machines. Our research is motivated by the burn-in ovens found in semiconductor manufacturing. So far research in this field mainly concentrated on control strategies that assume batches to be homogeneous, i.e., products should all belong to the same family. However, burn-in ovens may allow for simultaneous processing of alternative families of products. Families differ from each other with respect to product volume. We propose a new scheduling approach that addresses these situations. The objective is to minimize average flow time per product for the batch operation. Our so-called look-ahead strategy adapts its scheduling decision to shop status, which includes information on a limited number of near future arrivals. The potential of the new strategy is demonstrated by an extensive simulation study.

Key words

Scheduling, batch processing machines, simulation

1. Introduction

In many industries batch servers are used for efficient processing. Main reasons for batching, i.e., the grouping of a number of jobs which may be processed simultaneously, are the avoidance of set-ups or the facilitation of material handling. Examples include transportation systems (e.g. charter airline flights, shuttle buses and elevators), service environments (e.g. restaurants and mail shipment) and manufacturing systems (e.g. furnaces, plating baths).

In this paper we study a particular model of a batch processing machine motivated by the burn-in ovens found in the final testing phase in semiconductor manufacturing, see Uzsoy et al. (1992,1994), Chandru et al. (1993a), Chandra and Gupta (1997). The purpose of the final testing phase is to check the circuits for defects. Burn-in ovens are used in this phase to subject the circuits to thermal stress for an extended period of time. In this way latent defects may be discovered that would otherwise be found in the operating environment. The ovens allow for integrated circuits belonging to different families to be batched, i.e., processed simultaneously. Circuits may differ with respect to their size (Uzsoy 1994). Batch sizes are restricted by e.g. the physical sizes of the oven, or process constraints. Batch processing machines quite similar to burn-in ovens are the box ovens used in the manufacture of multi-layer ceramic capacitors (Koh et al. 2004). Other examples concern the ovens used for hardening synthetic aircraft parts (Hodes et al. 1992), and the furnaces used for giving industrial gas turbine blades the required mechanical properties (Oduoza 2002).

Our research focuses on the development of strategies for on-line scheduling of the aforementioned batch servers. Effective burn-in operation scheduling is considered of significant importance for lead-time reduction because it is frequently a bottleneck. This is due to long processing times relative to other operations – days vs. hours (Azizoglu and Webster 2000, Bhatnagar et al. 1999). Note that the semiconductor industry is very much driven by cycle times (Cigolini et al. 1999, Fowler et al. 2000). Typically, the rules to be developed should be both fast, i.e., computationally efficient, and responsive. The need for a responsive strategy follows from the unpredictability of the arrival pattern of products to be processed at the burn-in stage. This is caused by the characteristics of preceding manufacturing stages, like unequal processing times, test

failures, re-entrant flows and machine breakdowns, cf. Uzsoy et al. (1992). They harm the original release plan in terms of the product mix and the timing of operations.

In recent years, many strategies for on-line scheduling of batch servers have been developed, cf. e.g. Uzsoy et al. (1994), Van der Zee et al. (1997), Fowler et al. (2000). These strategies mainly address settings in which batches are assumed to consist of identical products only. Only Neale and Duenyas (2003) and Van der Zee (2004) study the case of compatible product families. However, they focus on product families that differ with respect to their requirements to processing times. In addition we will propose a *new control strategy* in this article - the Dynamic Job Assignment Heuristic for Compatible product families with non-identical Sizes (DJAH-C-Size). It addresses the model of a batch processing machine as it was introduced above. The rule may be characterized as a so-called look-ahead strategy, as it includes a limited amount of forecast data on near-future arrivals in deciding on the contents of the next batch to be loaded.

To test DJAH-C-Size a simulation study has been designed. In this study we consider performance of DJAH-C-Size relative to other strategies in terms of the average flow time realized per product. The study includes benchmark strategies like the First Come First Serve rule (FCFS), and existing look-ahead strategies.

The remainder of the paper is organized as follows: in the next section we will review related literature and outline contributions of this paper. In Section 3 we describe the assumptions, data, definitions and objective that underlie our study. The new heuristic rule will be introduced in Section 4. To establish the potential of the new rule a simulation study was designed. In Section 5 we discuss the alternative rules and configurations that are studied in terms of experimental factors and their ranges.

Outcomes of this study are analyzed in Section 6. Finally, conclusions are summarized in Section 7.

2. Review of literature

The control of batch processing machines used in semiconductor manufacturing received much attention in literature. Among the reasons mentioned for this particular interest are the role of batch machines as bottleneck resources, the high costs of equipment, and the significant role played by this industry in the overall economy (Azizoglu and Webster 2000).

Let us now classify the developed strategies for controlling batch processing machines. In queueing theory threshold strategies are studied which relate the decision to schedule a batch to a certain minimum queue length. An important example of such a strategy is the Minimum Batch Size rule (MBS), which was introduced by Neuts (1967). According to this strategy a batch starts service as soon as at least a certain fixed number of customers is present. Using a dynamic programming approach, Deb and Serfozo (1973) showed how this critical load should be chosen in order to minimize the expected discounted cost over an infinite horizon. If the cost of serving is set to zero and the cost of waiting is linear, minimizing the expected averaged cost is equivalent to minimizing the average flow time.

While the above strategies base their decision on local information only, full knowledge of future arrivals is assumed to be available in the field of deterministic machine scheduling. Two models of batch processing machines appear to be dominant in this field. The first model concerns the case of *incompatible product families*.

According to this model only products belonging to the same family may be processed simultaneously. This may e.g. be due to the different requirements set by each family with respect to processing conditions. Within each family each product requires the same processing time. The second model considers batch processing of *compatible product families*. In this case it is assumed that products belonging to alternative families may be processed simultaneously.

One of the early examples of scheduling rules for the first model is given by Ikura and Gimple (1986) who consider a single job family. They have addressed the problem of determining whether a schedule in which all the due dates can be met exists, given dynamic job arrivals. Later on, batch server models with multiple incompatible job families have been studied under a variety of conditions, see e.g. Uzsoy (1995) and Dobson and Nambimadon (2001). Compatible product families are addressed by Lee et al. (1992) who propose heuristics for the problem of minimizing makespan on parallel burn-in ovens. Also they address the problems of minimizing the number of tardy jobs and maximum tardiness for which an optimal algorithm is proposed. Chandru et al. (1993a,b), study the problem of minimizing total completion time within a similar context. Both Lee et al. and Chandru et al. address product families with equal sizes. Uzsoy (1994), Ghazvini and Dupont (1998), Azizoglu and Webster (2000), Melouk et al. (2004) and Koh et al. (2004, 2005) focus on product families with non-identical sizes. Uzsoy et al. (1992, 1994) summarize planning and scheduling models for this industry. Other surveys are supplied by Webster and Baker (1995) and Potts and Kovalyov (2000).

A third category of so-called look-ahead strategies has been developed based on the observation that in many practical cases the assumptions underlying deterministic machine scheduling are not met. In those cases the amount and quality of data on future

arrivals do not allow for a deterministic approach, cf. Glassey and Weng (1991), Duenyas and Neale (1997), Fowler et al. (2000). Typically, look-ahead strategies assume that only a limited number of near future arrivals are known and/or predicted. Glassey and Weng (1991) were among the first to introduce this type of strategies for (semiconductor) batch processing systems. They discuss the practical usability of a dynamic programming approach to find a sequence of loading times of given lots, in such a way that total delay is minimized. They argue that this approach fails for reasons of computational feasibility, and availability and quality of data on future arrivals. Therefore they present a Dynamic Batching Heuristic (DBH). This heuristic decides when to start a production cycle thereby aiming for a minimal average flow time. The planning horizon in DBH is just one service time. DBH proves to perform better than MBS, based upon the knowledge of just a few arrivals. Starting from the single product single machine shop discussed by Glassey and Weng other authors propose new look-ahead strategies in order to deal with several extensions. The first extension of the DBH rule concerns product families, cf. Fowler et al. (1992), Weng and Leachman (1993), Robinson et al. (1995), Duenyas and Neale (1997). Differences between products concern the required service time and/or maximum allowed batch size. Fowler et al. (2000) and Van der Zee et al. (1997, 2001) propose heuristics that cover the multiple machine case. All look-ahead strategies mentioned focus on the control of batch servers with *incompatible* job families. So far, only Neale and Duenyas (2003), and Van der Zee (2004) developed look-ahead strategies for batch processing machines with compatible product families. However, they study families with different requirements with respect to processing times, and identical product sizes.

Our research contribution concerns the development and testing of a new look-ahead strategy for the control of batch servers with compatible product families. As an objective we consider the minimization of the average flow time per product. We see the new control strategy as a practical means for shop floor control that offers some important advantages relative to existing approaches. Firstly, it may help to improve system performance relative to other heuristics. Next to existing look-ahead strategies, these heuristics include local strategies that do not consider future product arrivals. Secondly, it is capable of supporting speedy decisions making good use of little information on future jobs. Also the new heuristic is quite robust with respect to forecast data. In this way we address the limitations of many control strategies from the field of deterministic scheduling with respect to data requirements and response time.

3. Problem description

In the introduction to this paper several real-world examples of batch processing machines are mentioned. More in particular, we considered control of the burn-in ovens found in semiconductor manufacturing. In this section we take a more general focus by considering a particular model of a batch processing machine, which is characterized by compatible product families with non-identical product sizes. We start by describing the batch shop in detail. Building on this description we present a framework that characterizes the scheduling problem faced by the planner in general terms. In the next section the framework will be used to define the new control strategy. As a starting point for our discussion we will use figure 1.

[insert figure 1 about here]

3.1 *Batch shop*

The shop consists of a batch server, and a buffer. For the sake of simplicity and clarity of understanding we assume products to arrive in lots of size one. Extension of our approach towards situations with lot sizes greater than one is straightforward, cf. Van der Zee et al. 1996. Products are stored in the buffer until the time they are processed. The buffer is assumed to have an unlimited storage capacity. Multiple families (j) of products are considered, with $j \in J = \{1,2,..,N\}$. Families differ from each other with respect to product size (s_j). Processing time (T) is fixed and includes set-up and transport times. Hence, set-up activities are sequence independent. Batches may consist of products belonging to different families. The batch size is restricted, i.e., $\sum_{i \in B} s_{f(i)} < C$, with f(i) specifying the family of product i in batch B, and C equal to the maximum allowed batch size. The latter characteristic may e.g. be related to volume restrictions.

3.2 *A framework for decision-making*

The problem of the planner concerns the scheduling of jobs for the batch processing machine as introduced in the previous subsection. As an objective we consider the minimization of average flow time per product in the long run. Average flow time per product (F) is defined as:

$$F = \frac{\sum_{i=1..I} fl_i}{I} \qquad (1)$$

with

$$fl_i = w_i + T$$

In computing flow time for a product i (fl_i) we distinguish between two elements:

- The required processing time (T).
- The waiting time that accumulates before product i is loaded into the machine (w_i).

Note that as processing time is assumed to be fixed minimization of average flow time per product implies minimization of average waiting time per product.

Let us now define the scheduling problem for the planner in general terms using the objective as a starting point. Therefore we present a framework for decision-making (figure 2). The framework characterizes control strategies, i.e., scheduling routines, in terms of their triggers, information availability and usage, and decision structure. Implementation of this framework in practice assumes the availability of a shop floor control system. For example, most semiconductor companies employ computerized shop floor control systems. Among others they can provide reasonably accurate predictions of future arrivals times to a station, cf. Fowler et al. (2000).

[insert figure 2 about here]

Triggers

Two types of events govern shop dynamics: product arrivals and the completion of a batch job. Both events generate new information for the planner. As such these events correspond to decision moments, i.e., moments at which a planner may make the decision to load the machine. Obviously, new jobs are only released if both machine and products are available. Note that the arrival of information on future arrivals is not considered as a decision moment.

Information

Next to the static shop characteristics such as the required processing time and alternative product sizes (see Subsection 3.1), the planner has the following information on shop status at his disposal at a decision moment (t_0):

- Local information on queue lengths for each family j (q_j), with $j \in J = \{1,2,..,N\}$.
- For each family j the successive future arrival times $t_{a,j}$ ordered through the index a according to the moment of arrival, up to some specified look-ahead horizon LH.

For look-ahead strategies only *forecast information* on product arrivals within the *look-ahead horizon* is assumed to be available. This information may be incomplete or uncertain. Typically, existing strategies relate the look-ahead horizon to a number of future arrivals and/or a fixed time period, cf. Van der Zee et al. (1997). This implies that there is no uniform definition of the look-ahead horizon available in literature. A common denominator in defining the look-ahead horizon is, however, that the amount and/or quality of information on future product arrivals restrict the scheduling activity to the *next machine cycle*.

Decision options

Essentially two decision options are open for the planner:

- The loading of products into the batch machine at the decision moment (t_0).
- Do nothing, i.e., wait for a next decision moment.

The choice of options implies a rolling horizon approach in scheduling the batch machine. This approach does not allow for the scheduling of the batch machine at a future product arrival. After all, new or better quality information on future arrivals may become available in the meantime. Consequently, a better decision may be possible. For more background on this issue see previous research by e.g. Fowler et al. (1992) who studied robustness of look-ahead strategies with respect to forecast error.

Decision structure

Following the framework, a control strategy is described in terms of a procedure made up of three sequential steps (see figure 2):

I Look-ahead strategies will have to reach the goal of flow time minimization in the long run by adopting a *reduced objective*. After all, the look-ahead horizon is limited, see above. A natural way to reduce the objective is to consider minimization of product flow times (waiting times) for the next machine cycle, cf. equation (1). A disadvantage of such a reduction lies in the fact that it may make the strategy rather shortsighted – future consequences of current decisions are hardly considered. Previous research by Fowler et al. (1992), Weng and Leachman (1993), Duenyas and Neale (1997) made clear that next to flow times also machine utilization should be considered. Let us consider this point in somewhat more detail, starting with a definition of machine utilization (U(B)):

$$U(B) = \frac{T \sum_{i \in B} s_{f(i)}}{(t + T - t_0)C} \qquad (2)$$

Machine utilization equals the ratio of the work contents as determined by the composition of the batch (B) and available machine capacity. Obviously, U<1. Work

contents is computed as the sum of the products sizes of all products i in B times the processing time (T). Available machine capacity is determined by the cycle length and the allowed batch size (C). Note that cycle length may be different from the required processing time, as the planner may decide to postpone the loading of the next batch until some future moment $t>t_0$. For example, consider a batch consisting of 4 products to be scheduled at $t = t_0 + 6$. Product sizes are 10,20,30,30 respectively. Processing time equals 25 time units. Maximum allowed batch size equals 150. Machine utilization for this cycle is computed as 25*(10+20+20+30)/(150*(6+25)) = 43.0%.

Especially in case of high arrival rates it is important to consider machine utilization. After all, a choice for a batch realizing short flow times but a low utilization, may "eat capacity away" for future machine cycles. This may cause additional delay in future. In some cases capacity losses may even lead to an unstable system (Duenyas and Neale 1997). To address this issue Fowler et al. (1992) propose a "full loads priority rule" for the case of incompatible product families with product size 1 (unit size). The rule prioritizes full loads, i.e., product families j for which cumulative size for those items in queue equals or exceeds the allowed batch size ($q_j \geq C$, given $s_j=1$), over non-full loads ($q_j<C$). Note how full loads act as an indicator of potential future capacity shortages. As such it serves *as a switch for including machine utilization in the reduced objective*. For more details on the full loads rule see Fowler et al. (1992, 2000) and Weng and Leachman (1993). For the case of compatible product families this rule has to be refined, see Section 4.

II The *selection of the best candidate batch* for the next machine cycle (B^*) concerns three phases:

- Batch formation – the composition of a set of candidate batches.
- The computation of costs associated with each candidate.
- Choice of the best candidate batch.

Each candidate batch ($B_{b,t}$) is characterized by its contents and its scheduling moment (t). Here we assume that the scheduling moment equals the earliest moment all products in $B_{b,t}$ have arrived. Next to the decision moment (t_0) alternative scheduling moments concern forecasted product arrivals for all families j ($t_{a,j}$) up to the look-ahead horizon (LH). Depending on the number and size of available products at t alternative batches b = 1,2,….may be considered.

For incompatible product families the set of candidate batches (CB) will typically not be very large. This is due to the demands set on the homogeneity of the batch contents and the restricted batch size, i.e., $\|B_{b,t}\| \leq C$, where $\|B_{b,t}\|$ denotes the cardinality of $B_{b,t}$. On the other hand for compatible product families the combinatorial problem may be of considerable size.

A *cost function* (CF) relates batch contents and its scheduling moment to system performance. Finally, the best candidate batch is chosen from CB as the one that involves minimum costs, i.e., $B^* = \text{argmin}_{B_{b,t} \in CB} CF(B_{b,t})$.

III In the third step a *dispatching decision* is made with respect to the best candidate batch. This concerns the choice among the two decision options, see above. In case the scheduling moment (t) of the best candidate batch B^* equals the decision moment ($t=t_0$) the respective products are released from the buffer. In all other cases, i.e., $t>t_0$, the planner waits for a next arrival.

4. A new look-ahead strategy

In this section we will describe the new look-ahead strategy. As a format for description we use the framework for decision-making as we defined it in the previous section. To guarantee a clear understanding of indexes, parameters and variables used in the construction of the control strategies, we first explain the notation (Subsection 4.1). Next, in Subsection 4.2, we define the new look-ahead strategy.

4.1 *Notation*

In the sequel we will use the following notation:

Indexes

a = index for the forecasted arrival moments = 1,2…

b = batch identifier = 1,2…

i = product identifier = 1,2,…

j = product family = 1,2,..N

Parameters

C = the allowed batch size, i.e., the cumulative size of products that may be loaded in the batch machine at the same time.

J = the set of product families.

LH = the look-ahead horizon.

N = the number of product families.

q_j = the number of products of family j in queue at t_0.

s_j = the size of products in family j.

t_0 = the decision moment, i.e., the moment the scheduling activity is being triggered due to a change of shop status.

$t_{a,j}$ = the time of the a-th arrival after t_0 for products of family j.

T = the required processing time.

Variables

t = a scheduling moment, i.e., a moment considered by the planner for loading the next batch into the machine.

CB = the set of candidate batches.

$B_{b,t}$ = batch b being considered for loading in the batch machine at t, b = 1,2…

B^* = the best candidate batch, i.e., the batch for which loading implies minimum costs.

$f(i)$ = the family product i belongs to.

$P(t)$ = the cumulative queue length over all product families at t.

$CF^0(B_{b,t})$ = weighted waiting costs if products in candidate batch $B_{b,t}$ would be loaded in the machine at t.

$CF^1(B_{b,t})$ = loss of machine efficiency if products in candidate batch $B_{b,t}$ would be loaded in the machine at t. Efficiency losses are related to machine utilization.

$W(B_{b,t})$ = waiting costs if products in candidate batch $B_{b,t}$ would be loaded in the machine at t.

4.2 *Definition of the look-ahead strategy*

In this subsection we will discuss the new look-ahead strategy. We will refer to the heuristic as the Dynamic Job Assignment Heuristic for Compatible product families with non-identical Sizes (DJAH-C-Size). The new rule is partly based on the Dynamic Job Assignment Heuristic (DJAH) that addresses the case of incompatible product families (Van der Zee et al. 1997). First we define DJAH-C-Size in conformity with the *decision structure* introduced in Subsection 3.2. Next we highlight its extensions relative to the DJAH rule.

Step I: Determine which reduced criterion to apply

In Subsection 3.2 it was made clear that a look-ahead strategy should not only focus on flow time minimization for the next machine cycle. Also machine utilization should be considered. Look-ahead strategies addressing incompatible product families solve this issue through a so-called "full loads" rule, see Subsection 3.2. Here, we consider its modification for the DJAH-C-Size rule. Therefore we distinguish between two situations at the decision moment (t_0):

- Cumulative size of products in queue exceeds the allowed batch size, i.e., $\sum_{j \in J} q_j s_j \geq C$.
- Cumulative size of products in queue is less than the allowed batch size, i.e., $\sum_{j \in J} q_j s_j < C$.

In the first case we may apply the aforementioned full loads rule in a straightforward way – we decide to consider machine utilization. However, how to decide on the criterion to apply in the second case? We solved the issue by considering the question

whether it would be possible to include the next product - arriving after t_0 - in the batch. In case of a positive answer we adopt the flow time criterion, otherwise we consider machine utilization.

Step II: Select the best candidate batch

Form candidate batches

We reduce the combinatorial problem associated with batch formation by:

- Restricting the number of alternative scheduling moments (t), and
- Applying alternative rules for batching, i.e., determining batch contents.

In case a flow time criterion is adopted (cf. Step I) we restrict alternative scheduling moments to the next arrival ($t_1=\min_{j\in J} t_{1,j}$). Also $t_1 < t_0+T$. After all, if a better candidate batch would be associated with a later scheduling moment, why not schedule the best candidate batch available at t_0 first? Note how these restrictions make batching a trivial exercise – candidate batches $B_{b,t}$ consist of those products available at t_0 and t_1 respectively, i.e., $B_t=B_{b,t}$. In principle it would be possible to consider more scheduling moments. For example, we may include future arrivals (t) up to the moment cumulative size of available products does exceed the allowed batch size. However, previous research on look-ahead strategies indicated that the inclusion of more scheduling moments (arrivals) in decision making hardly leads to a performance improvement (Fowler et al. 1992). In a series of supplemental simulation experiments we found that performance differences typically are in a range between -0.4 and 0.4%.

Alternatively, in case of a focus on machine utilization, we restrict the number of scheduling moments to be considered in a simple way. Therefore we first determine the best candidate batch $B_{b,t}$ among those candidate batches available at t_0, using the procedure for batch formation and cost functions specified below. The utilization u* that is associated with this batch is used to compute the time frame (TF) for considering alternative scheduling moments:

$$TF = \min(t_0 + T\frac{1-u*}{u*}, t_0 + T) \quad (3)$$

For example, if u* = 0.80, alternative scheduling moments up to t_0 + 0.25T may be considered. Clearly, it does not make sense to consider scheduling moments that are not within t_0+T, see above. The above restriction implies a significant reduction of the number of alternative scheduling moments. An interesting question in this respect is how sensitive system performance would be for the number of future product arrivals to be considered. After all, in many cases costs of postponement will be significant, as cumulative size of items in queue tends to be large. On the other hand, relative gains typically will be small. We come back to this point in Section 6.

This leaves the batching decision. In fact this boils down to solving a classic bounded knapsack problem – how to maximize utilization (profit), starting from those products and their associated sizes (weights) available at the scheduling moment t. Here the term "bounded" refers to the fact that the number of available products per family is restricted – to those available at the scheduling moment. Several algorithms exist to deal with the problem. They reflect a trade-off between quality of the solution and computational efficiency. For example, optimal solutions can be found by using common dynamic programming formulations (DP). However, for larger problems

computation times may become too significant for on-line decision making, cf. Section 6. In these cases approximate rules may help. A first example of such a rule is the greedy sorting rule, cf. Kellerer et al. 2004:

1. Sort all available products k = 1..K in decreasing order of their size, i.e., $s_1 > s_2 > s_3 \ldots > s_{K-1} > s_K$
2. Set k = 1; AC = C; Batch $B_t = \varnothing$
3. If $s_k < AC$ then $B_t = B_t \cup \{k\}$; $AC = AC - s_k$
4. If k = K then Stop else set k = k + 1; go to 3.

According to this rule products k available at t may be added to the batch B_t as long as the required space does not exceed the available space (AC, with $AC \leq C$). Note how B_t resembles the choice of a batch made by the rule, i.e., the one batch chosen from the candidate batches $B_{b,t}$. Another rule is supplied by Martello and Toth (1984). The rule is referred to as MTGS. The idea is to apply the above sorting rule K times by considering product sets {1,..,K},{2,..,K},{3,..,K} and so on, respectively, and take the best solution. Clearly, there is a better chance of finding improved solutions as more alternatives are considered. Many more examples can be given of rules for solving the knapsack problem, see e.g. Kellerer et al (2004) and Martello and Toth (1990) for overviews. In this article we will restrict ourselves to the rules mentioned above: Dynamic programming, the greedy sorting rule, and MTGS. From the abundance of rules available we strove to select a few representative rules with respect to the inherent trade-off between solution quality and computational efforts (see above). However, alternative rules can be incorporated in a straightforward way. In this respect the DJAH-C-Size heuristic is generic.

Compute costs for candidate batches

If *no full loads* are present DJAH-C-Size aims at flow time (waiting time) minimization for the next machine cycle, see step I. Let us consider the respective waiting *costs*, using figure 3 as a starting point in our discussion. The figure shows how the loading of a batch B_{t_1} into the machine causes waiting times for products up to the *cost horizon* t_1+T, i.e., the moment the machine would complete service.

[insert figure 3 about here]

Where figure 3 addresses a specific case, equation (4) captures the general case:

$$W(B_t) = (t - t_0)\sum_{j \in J} q_j + \sum_{t < t_{a,j} \le t+T,\, j \in J} (t + T - t_{a,j}) \tag{4}$$

The first term in equation (4) refers to delay caused for products in queue by postponing the loading of the machine until t. The second term computes waiting times for products arriving during processing. Note how the terms match product categories specified in figure 3. To account for the fact that the number of products in a batch are different we define our cost function as:

$$CF^0(B_t) = \frac{W(B_t)}{P(t)}$$
$$\textit{with}$$
$$P(t_0) = \sum_{j \in J} q_j \tag{5}$$
$$P(t_1) = 1 + \sum_{j \in J} q_j$$

In case *full loads* are present we propose an alternative cost function that relates candidate batches B_t to machine utilization:

$$CF^1(B_t) = 1 - U(B_t) = 1 - \frac{T \sum_{i \in B_t} s_{f(i)}}{(t + T - t_0)C} \quad (6)$$

Equation (6) computes efficiency losses by considering *machine use* relative to *machine capacity*, cf. equation (2).

Choose best candidate batch

The best candidate batch B^* equals the batch $B_{b,t}$ for which minimum costs are computed, i.e., $B^* = \text{argmin}_{B_{b,t} \in CB} CF(B_{b,t})$. If the application of this rule results in a tie, because the same costs are found for more than one candidate batch, a random choice is made.

Step III: Dispatching decision

In case the scheduling moment (t) of the best candidate batch B^* equals the decision moment ($t=t_0$) the respective products are released from the buffer. In all other cases, i.e., $t>t_0$, the planner waits for a next arrival.

Above we defined the new DJAH-C-Size rule. Let us now briefly consider main extensions of the rule relative to existing look-ahead strategies (see Section 2). This may help us to embed the new heuristic into literature and to highlight the way it is dealt with the specific characteristics of the system. The main issue to be dealt with is the

increased scheduling complexity that follows from the flexibility with respect to batch contents – product families may be combined in one batch. We address this problem by:

- The development of a refined rule for deciding whether machine utilization should be considered as a reduced objective (step 1).
- The definition of a generic procedure for batch formation. The procedure allows for the inclusion of alternative rules for solving the knapsack problem (step 2: batch formation). In this way it allows for a choice of rule that reflects the trade-off to be made by the planner with respect to solution quality and computational efficiency. Further, extensions of the packing problem to deal with two or three dimensions are facilitated in this way. We will, however, not address the latter type of problems here.
- The definition of a new cost function for the case machine utilization should be considered (step 2: computing costs).

A final remark concerns the Rolling Horizon Cost Rate heuristic (RHCR), see Robinson et al. (1995). This rule is quite similar to the DJAH rule we used as a starting point for the construction of the new DJAH-C-Size rule. The RHCR rule can be extended in the same way as the DJAH rule. A series of simulation experiments – in line with those mentioned in Section 5 - made clear that such a rule realizes about the same system performance. Typically, DJAH-C-Size outperforms the extended RHCR rule by at most 0.5% for moderate and high arrival rates. For low arrival rates RHCR outperforms the DJAH-Size rule by up to 0.3%.

5. Simulation study

A simulation study was designed to: (i) demonstrate the potential of the new DJAH-C-Size heuristic, and (ii) improve insight in its practical relevance. Table 1 gives an overview of the experimental factors and their settings.

The key factor in the study is the control strategy (factor 1). We consider the First Come First Serve rule (FCFS) as a benchmark. It is a simple rule that is much applied in both practice and theory. It supplies us with an upper bound on system performance. We implement this rule by starting a new batch every time the machine completes a job, unless no products are available in queue. The contents of the batch follows the classic FCFS-order. As an alternative to FCFS we study FCFS-D and FCFS-I. FCFS-D and FCFS-I are equivalent to FCFS, except for the fact that products in queue are ordered according to Decreasing/Increasing size.

The DJAH-C-Size strategy is considered for three alternative knapsack rules: the Dynamic Programming approach(DP), the Greedy sorting Rule (GR), and MTGS, see Section 4. Also we consider the situation in which no such rule is applied (None). In the latter case it is assumed that a batch is being loaded in case of a full load. A full load is assumed to be present if cumulative size of products in queue exceeds the allowed batch size or if the next arrival would not fit into the batch (given the products already present in queue at the decision moment), cf. Section 4. For batch formation a FCFS ordering is applied. This rule is quite similar to the existing DJAH rule. It helps us to obtain insight in the added value of knapsack rules for batch formation.

In all experiments Poisson arrivals were studied (factor 2). Arrival rates are related to the workload ρ for the batch machine (factor 3). Workload for the batch machine is defined as:

$$\rho = \lambda \sum_{j=1}^{N} \frac{p_j s_j T}{C} \quad (7)$$

Workload is computed by considering the overall arrival rate (λ), the share of product families in the product mix (p_j), the product size for each family (s_j), the required processing time (T) and the maximum batch size allowed (C). Note that in general workloads tend to be smaller than utilization levels for the batch machine. This is due to the presence of alternative product sizes and the possibility of delaying the loading of the batch machine.

Factors 4-8 relate to product and machine characteristics. For these factors one default setting is defined, which reflects a particular setting for shop characteristics. These settings are marked boldly in table 1. Alternative system settings are chosen by changing the value for exactly one of these factors. In this way we consider the effects on system performance of alternative settings for the number of product families, product mix, product size and the maximum allowed batch size. Also we tested the heuristics for their robustness, by reducing the percentage of arrivals known within the look-ahead horizon to 80% (factor 9). This is realized by associating a 20% chance with each arriving product that it will not be reported to the planner before its actual arrival. Note how this choice of experiments implies that the experimental design is not fully crossed, i.e., not all possible combinations of factor settings have been studied.

The software package that was used to carry out the simulation experiments is EM-Plant[1]. The principles of object-oriented design underlying this language make it a flexible and efficient tool for model building. The performance for each heuristic was estimated using the batch means method, compare e.g. Hoover and Perry (1986), Law

and Kelton (2000). Each batch concerns 10,000 products. A total of 31 batches is considered for each experiment, where each first batch was discarded to account for any start up bias. Uncorrelatedness was tested using the runs test (see e.g. Hoover and Perry 1986). It showed no significant correlation, given a significance level of $\alpha = 0.05$.

6. Analysis of simulation results

In the previous section the design of the simulation study has been discussed. In this Section we will analyze the outcomes of this study. First we present results for the default settings (Subsection 6.1). Next we consider the experiments with alternative settings for product and machine characteristics (Subsection 6.2). We conclude, in Subsection 6.3, by discussing robustness of the new look-ahead strategy with respect to forecast data.

6.1 *Default settings*

In this subsection we discuss simulation results for the default shop configurations defined in Section 5. The results are displayed in figure 4. Performance differences for the heuristics have been tested for statistical validity using a paired t-approach, cf. Law and Kelton 2000. The tests pointed out that differences greater than 0.25% should be considered significant.

[insert figure 4 about here]

[1] EM-Plant™ Ver.7.0 (Stuttgart : Technomatix)

The data in figure 4 allow us to draw two general conclusions with respect to system performance:

1. Including forecast data in decision-making results in better flow time performance.
2. The efficient use of machine capacity is an important criterion in minimizing flow time.

The first conclusion is based on the comparison of the local strategies (FCFS, FCFS-D, FCFS-I), and the look-ahead strategies (DJAH(..)). For low and moderate workloads differences are in the order of 10% and 20% respectively. These outcomes are in line with earlier findings by e.g. Glassey et al. 1991, Fowler et al. 1992, and Robinson et al. 1995. For higher workloads differences go up significantly (50% and more). Clearly, this is due to the restrictions set on the allowed batch size. Strategies that do not consider efficient use of the system in minimizing flow time tend to do worse. A clear illustration of this effect is given by DJAH(NONE). This strategy realizes good results for workloads up to 50% building on look-ahead data. However, as it does not consider utilization in its batching decision, its performance is worse for higher workloads.

Also it may be concluded that the use of better quality knapsack rules for batching result in better system performance, cf. DJAH(GR),DJAH(MTGS) vs. DJAH(DP). This performance improvement comes at a price – average execution time for the dynamic programming rule is 6 ms for a configuration of 6 products and a workload of 90%, while GR and MS need at most 0.6 ms (Pentium 4 – 2.0Ghz). In this respect it may be worthwhile to mention the results of a series of additional experiments. We found that in case the knapsack evaluation was restricted to the decision moment – by leaving

alternative scheduling moments out - system performance was less than 1% worse, cf. Section 4.

6.2 *Alternative settings for product and machine characteristics*

To study the effect of alternative settings of product and machine characteristics on system performance three series of simulation experiments were carried out. Outcomes of these experiments can be found in tables 2(a)-(e). Let us consider some interesting outcomes of these experiments:

[insert table 2 about here]

Product mix

The default settings assume a balanced product mix, i.e., relative shares of families in the product mix are equal. Here we consider two "unbalanced" product mixes. The first product mix is dominated by products with large and small sizes (table 2(a)). Alternatively, a shop configuration is studied in which two product families with quite similar sizes have the largest shares in the product mix (table 2(b)). Results in both tables make clear that a more homogeneous mix (table 2(b)) outperforms a more heterogeneous mix (table 2(a)). Performance differences are strongly increasing for higher shop loads. This is a clear consequence of the fact that it is easier to realize high machine utilization in case of more homogeneous product sizes. Note how the intermediate position on performance held by the default configuration is in conformity with this reasoning, cf. figure 4.

Allowed batch size

As indicated in table 2(c) an increase of the allowed batch size makes system performance worse for low and moderate workloads. This is probably caused by the higher arrival rate that compensates for the higher machine capacity, cf. equation (7). Hence queues on average will be longer. This makes the relative gains of delaying the loading of the machine smaller. On the other hand for high workloads (70-90%) performance is improved significantly relative to the default configuration. A likely explanation lies in the fact that relative size of the products is smaller given the larger batch size. This makes it easier to realize a higher utilization of the batch processing machine. In this respect it is noteworthy that the local rules succeed in realizing system stability – as made clear by the outcomes for high (90%) workloads.

Number of products

In the default case we consider four product families. Tables 2(d), 2(e) present results for shops in which the number of product families is two and six respectively. A first conclusion we may draw is that main observations for the default configuration largely apply for both configurations. In fact performances differences are small for workloads up to 50%. However, for higher workloads the issue of "packability" arises – how easy is it to form a batch that realizes a high machine utilization. For example, large and deviant sizes typically make it hard to realize a high utilization. Moreover, one not only has to consider "packability" with respect to the current batch, also future batches should be considered (Dobson and Nambimadon 2001). In this respect the outcome for DJAH(DP) in case of two products and a 90% workload is illustrative. For this case DJAH(DP) is outperformed by DJAH(GR) and DJAH(MTGS). This is because of the fact that it allows products with small sizes take the place of products with large sizes.

As a net consequence queue length for the less packable products (large sizes) grows. Alternatively, both other rules prioritize large sizes. Hence they realize a better performance.

6.3 *Robustness*

In practice information on future arrivals will often consist of forecasts. These forecasts may be incomplete and error prone. To get an insight in the robustness of the rules with respect to forecast information a series of experiments was carried out. It concerns the case of missing data. The shop configuration is altered by reducing the number of arrivals reported to the planner to 80% (see Section 5). All look-ahead strategies appear to be quite robust: while information on future arrivals is reduced by 20% system performance is worsened by at most 2.5%. Hence a comfortable performance improvement is left relative to the local strategies.

[insert table 3 about here]

7. Conclusions

In this article we studied the dynamic control of a batch-processing machine in the presence of compatible product families. Families may differ with respect to product size. We propose a new look-ahead strategy to deal with this situation: the Dynamic Job Assignment Heuristic for Compatible product families with non-identical Sizes (DJAH-C-Size). The new rule aims at minimization of the average flow time per product in the long run. It adapts its scheduling decision to shop status, which includes information on a limited number of near-future product arrivals within the look-ahead horizon. The

scheduling activity involves a subtle trade-off between flow time performance for the next machine cycle and the effects the batching decision, i.e., a specific choice of batch contents, may have on long term system performance. In this respect efficient use of the batch machine in terms of utilization appears an important aspect to consider. Here utilization is determined by both the allowed batch size and product sizes.

By an extensive simulation study it has been shown how DJAH-C-Size outperforms alternative heuristics for a large variety of configurations. Most of the benefits are realized for higher workloads (70% and up). Under such conditions the knapsack algorithm incorporated in DJAH-C-Size for supporting the batching decision makes the difference relative to alternative rules. It should be mentioned that DJAH-Size is flexible with respect to the choice of the knapsack algorithm. In this respect it reflects the trade-off between solution quality and computational efficiency, which may be of relevance for adaptive decision-making.

Although this research has answered some questions with respect to dynamic batch shop control, many interesting directions for further research remain. For example, while our research focused on flow time performance, it is worthwhile from a practical perspective to study due date and cost related criterions. Another interesting avenue concerns extensions of the model such as multi-server batch machines, non unit sized jobs (Uzsoy 1994), limitations on buffer size, non uniform processing times (Neal and Duenyas 2003), the possibility of re-entry flows (Glassey et al. 1993), sequence dependent set-up times (Robinson et al. 1995, Van der Zee 2002), and quality constraints that restrict the possibility to postpone loading of products (Hodes et al. 1992, Neal and Duenyas 2003). Finally, there is a need for more case related research in

this field. This may shed more light on the practical use of look-ahead strategies, and the prerequisites for applying them.

Appendix Overview of control strategies

DBH	Dynamic Batching Heuristic (Glassey and Weng 1991).
DJAH	Dynamic Job Assignment Heuristic (Van der Zee et al. 1997).
DJAH-C-Size	Dynamic Job Assignment Heuristic for Compatible product families with non-identical Sizes (this paper).
DJAH(DP)	Dynamic Job Assignment Heuristic for Compatible product families with non-identical Sizes. Dynamic Programming (DP) is used for solving the knapsack problem (this paper).
DJAH(GR)	Dynamic Job Assignment Heuristic for Compatible product families with non-identical Sizes. The Greedy Rule (GR) is used for solving the knapsack problem (this paper, Kellerer et al. 2004).
DJAH(MTGS)	Dynamic Job Assignment Heuristic for Compatible product families with non-identical Sizes. MTGS is used for solving the knapsack problem (this paper, Martello and Toth 1990).
DJAH(NONE)	Dynamic Job Assignment Heuristic for Compatible product families with non-identical Sizes. No application of knapsack rule – a batch is being loaded in case of a full load, cf. Section 5 (this paper).
FCFS	First Come First Serve rule.
FCFS-D	First Come First Serve rule. Products are ordered according to Decreasing size (this paper).
FCFS-I	First Come First Serve rule. Products are ordered according to Increasing size (this paper).
MBS	Minimum Batch Size rule (Neuts 1967).
RHCR	Rolling Horizon Cost Rate Heuristic (Robinson 1995).

References

Avramidis, A.N., Healy, K.J., and Uzsoy, R., 1998, Control of a Batch Processing Machine: A Computational Approach, International Journal of Production Research 36(11), 3167-3181.

Azizoglu, M., and Webster, S., 2000, Scheduling a batch processing machine with non-identical job sizes, International Journal of Production Research 38(10), 2173-2184.

Bhatnagar, R., Chandra, P., Loulou, R., and Qiu, J., 1999, Order Release and Product Mix Coordination in a complex PCB Manufacturing Line with Batch Processors, International Journal of Flexible Manufacturing Systems 11(4), 327-351.

Chandra, P., and Gupta, S., 1997, Managing batch processors to reduce lead-time in a semi conductor packaging line, International Journal of Production Research 35(3), 611-633.

Chandru, V., Lee, C.Y., and Uzsoy, C.Y., 1993a, Minimizing total completion time on batch processing machines, International Journal of Production Research 31(9), 2097-2121.

Chandru, V., Lee, C.Y., and Uzsoy, C.Y., 1993b, Minimizing total completion time on batch processing machines with job families, Operations Research Letters 13(2), 61-65.

Cigolini, R., Comi, A., Micheletti, A., Perona, M., and Portioli, A., 1999, Implementing New Dispatching Rules at SGS-Thomson Microelectronics, Production Planning & Control 10(1), 97-106.

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

Dobson, G., and Nambimadon, R.S., 2001, The batch loading and scheduling problem, Operations Research 49(1), 52-65.

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

Fowler, J.W., Hogg, G.L., and Phillips, D.T., 1992, Control of multiproduct bulk service diffusion/oxidation processes, IIE Transactions 24(2), 84-96.

Fowler, J.W., Hogg, G.L., and Phillips, D.T., 2000, Control of multiproduct bulk service diffusion/oxidation processes. Part 2: Multiple servers, IIE Transactions 32(4), 167-176.

Ghazvini, F.J. and Dupont, L., 1998, Minimizing mean flow times criteria on a single batch processing machine with non-identical jobs sizes, International Journal of Production Economics 55(3), 273-280.

Glassey, C.R., and Weng, W.W., 1991, Dynamic batching heuristic for simultaneous processing, IEEE Transactions on Semiconductor Manufacturing 4(2), 77-82.

Glassey, C.R., Markgraf, F., and Fromm, H., 1993, Real time scheduling of batch operations, in Optimization in Industry: mathematical programming and modeling

techniques in practice, T.A. Ciriani and R.C. Leachman (eds), Wiley, Chichester, 113-137.

Gurnani, H., Anupindi, R., and Akella, R., 1992, Control of Batch Processing Systems in Semiconductor Wafer Fabrication Facilities, IEEE Transactions on Semiconductor Manufacturing 5(4), 319-328.

Hodes, B., Schoonhoven, B., and Swart, R., 1992, On line planning van ovens. Technical Report, University of Twente, The Netherlands (in Dutch).

Hoover, S.V., and Perry, R.F., 1989, Simulation a Problem Solving Approach (Reading MA, Addison Wesley).

Ikura, Y., and Gimple, M., 1986, Efficient Scheduling Algorithms for a single Batch Processing Machine, Operations Research Letters 5(2), 61-65.

Kellerer, H., Pferschy, U., and Pisinger, D., 2004, Knapsack Problems (Berlin, Springer Verlag).

Koh, S.G., Koo, P.H., Ha, J.W., and Lee, W.S., 2004, Scheduling parallel batch processing machines with arbitrary job sizes and incompatible job families, International Journal of Production Research 42(19), 4091-4107.

Koh, S.G., Koo, P.H., Kim, D.C., and Hur, W.S., 2005, Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families, International Journal of Production Economics (In press).

Law, A.M., and Kelton, W.D., 2000, Simulation Modeling and Analysis (Singapore, McGraw-Hill).

Martello, S., and Toth, P., 1984, Worst-case analysis of greedy algorithms for the subset sum problem, Mathematical programming 28, 198-205.

Martello, S., and Toth, P., 1990, Knapsack Problems – Algorithms and Computer Implementations (Chichester, Wiley).

Melouk, S., Damodran, P., and Chang, P.Y., 2004, Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing, International Journal of Production Economics, 87(2), 141-147.

Lee, C.Y., Uzsoy, R., and Martin-Vega, L.A., 1992, Efficient algorithms for scheduling semiconductor burn-in operations, Operations Research 40(4), 764-775.

Neale J.J., and Duenyas, I, 2003, Control of a batch processing machine serving compatible job families, IIE Transactions 35(8), 699-710.

Neuts, M.F., 1967, A general class of bulk queues with Poisson input, Annals of Mathematical Statistics 38(3), 759 – 770.

Oduoza, C.F., 2002, Capacity Management of Heat Treatment Process in a Manufacturing Environment. Proceedings of the 12th International Conference on Flexible Automation & Intelligent Manufacturing, edited by W.G. Sullivan, M.Ahmad, D. Fichtner, W. Sauer, G. Weigert, T. Zerna (München: Oldenbourg Verlag), 878-888.

Potts, C.N., and Kovalyov, M.Y., 2000, Scheduling with batching: a review, European Journal of Operational Research 120(2), 228-249.

Robinson, J.K., Fowler, J.W., and Bard, J.F., 1995, The use of upstream and downstream information in scheduling semiconductor batch operations, International Journal of Production Research 33(7), 1849-1869.

Uzsoy, R., Lee, C.Y., and Martin-Vega, L.A., 1992, A review of production planning and scheduling models in the semiconductor industry, Part I: System characteristics, performance evaluation and production planning, IIE Transactions 24(5), 47-60.

Uzsoy, R., Lee, C.Y., and Martin-Vega, L.A., 1994, A review of production planning and scheduling models in the semiconductor industry, Part II: Shop-floor control, IIE Transactions 26(4), 44-55.

Uzsoy, R., 1994, Scheduling of a single batch processing machine with nonidentical job sizes, International Journal of Production Research 32(7), 1615-1635.

Uzsoy, R., 1995, Scheduling batch processing machines with incompatible job families, International Journal of Production Research 33(10), 2685-2708.

Webster, S., and Baker, K.R., 1995, Scheduling groups of jobs on a single machine, Operations Research 43(4), 692-703.

Weng, W.W., and Leachman, R.C., 1993, An improved methodology for real-time production decisions at batch-process work stations, IEEE Transactions on Semiconductor Manufacturing 6(3), 219-225.

Zee, D.J. van der, Harten, A. van, and Schuur, P.C., 1996, Dynamic job assignment heuristics for multi-server batch operations. Proceedings of the 2nd International Conference on Computer Integrated Manufacturing in the Process Industries, edited by J.C. Fransoo, W.G.M.M. Rutten (Eindhoven, Beta), 558-573.

Zee, D.J. van der, Harten, A. van, and Schuur, P.C., 1997, Dynamic job assignment heuristics for multi-server batch operations - A Cost-Based Approach, International Journal of Production Research 35(11), 3063-3093.

Zee, D.J. van der, Harten, A. van, and Schuur, P.C., 2001, On-line scheduling of multi-server batch operations, IIE Transactions 33(7), 569-586.

Zee, D.J. van der, 2002, Adaptive scheduling of batch servers in flow shops, International Journal of Production Research 40(12), 2811-2833.

Zee, D.J. van der, 2004, Dynamic scheduling of batch servers with compatible product families, International Journal of Production Research 42(22), 4803-4826.

Table 1 Design of the Simulation Study

	Factor	Settings
1	Control Strategy	**FCFS** **FCFS-D** **FCFS-I** **DJAH-C-Size(None)** **DJAH-C-Size(GR)** **DJAH-C-Size(MTGS)** **DJAH-C-Size(DP)**
2	Interarrival Distribution	**Negative Exponential**
3	Workload (%)	**10,20,30,40,50,60,70,80,90**
4	No. Products Families	2,**4**,6
5	Product Mix (%)	**Equal**;(40,10,10,40);(10,40,40,10)
6	Product Size per Product Family	(10,40);**(10,20,30,40)**;(10,15,20,30,35,40)
7	Processing Time	**25**
8	Maximum Allowed Batch Size	**100**; 200
9	Information Quality (% Arrivals unknown)	**0**,20
	Number of Experiments	441

Table 2 Simulation results for alternative system configurations

(a) Product mix (40,10,10,40)

WL(%)	FCFS	FCFS_D	FCFS_I	DJAH(NONE)	DJAH(GR)	DJAH(MTGS)	DJAH(DP)	MIN	MAX
10	29.72	29.72	29.71	27.39	27.40	27.40	27.40	27.39	29.72
20	33.25	33.31	33.23	29.29	29.33	29.32	29.32	29.29	33.31
30	35.81	36.06	35.72	30.99	31.06	31.05	31.04	30.99	36.06
40	37.91	38.64	37.66	32.81	32.83	32.82	32.78	32.78	38.64
50	40.25	42.18	39.69	35.18	34.97	34.91	34.80	34.80	42.18
60	44.00	48.73	42.82	38.98	37.78	37.63	37.33	37.33	48.73
70	51.58	64.22	49.26	46.72	42.34	41.98	41.05	41.05	64.22
80	76.94	123.11	79.74	71.69	51.95	51.12	47.69	47.69	123.11
90	∞	∞	∞	∞	85.55	83.59	65.95	65.95	85.55

(b) Product mix (10,40,40,10)

WL(%)	FCFS	FCFS_D	FCFS_I	DJAH(NONE)	DJAH(GR)	DJAH(MTGS)	DJAH(DP)	MIN	MAX
10	29.69	29.70	29.69	27.36	27.36	27.36	27.36	27.36	29.70
20	33.13	33.14	33.12	29.17	29.19	29.19	29.19	29.17	33.14
30	35.50	35.55	35.46	30.72	30.78	30.77	30.76	30.72	35.55
40	37.31	37.45	37.19	32.28	32.36	32.32	32.29	32.28	37.45
50	39.18	39.51	38.91	34.17	34.20	34.09	33.99	33.99	39.51
60	41.93	42.61	41.41	36.97	36.72	36.41	36.15	36.15	42.61
70	47.13	48.48	46.14	42.31	40.78	39.99	39.31	39.31	48.48
80	62.16	64.34	62.79	57.43	49.26	47.15	45.26	45.26	64.34
90	∞	∞	∞	∞	76.57	69.27	62.07	62.07	76.57

(c) Allowed batch size = 200

WL(%)	FCFS	FCFS_D	FCFS_I	DJAH(NONE)	DJAH(GR)	DJAH(MTGS)	DJAH(DP)	MIN	MAX
10	33.00	33.00	33.00	29.03	29.03	29.03	29.03	29.03	33.00
20	36.11	36.11	36.11	31.26	31.26	31.26	31.26	31.26	36.11
30	37.12	37.13	37.11	32.69	32.70	32.70	32.70	32.69	37.13
40	37.56	37.64	37.52	33.71	33.75	33.75	33.74	33.71	37.64
50	37.96	38.18	37.86	34.65	34.76	34.75	34.70	34.65	38.18
60	38.79	39.35	38.52	35.83	36.05	36.03	35.86	35.83	39.35
70	40.54	41.94	39.92	37.80	38.13	38.11	37.51	37.51	41.94
80	45.20	48.71	43.85	42.64	43.00	42.96	40.75	40.75	48.71
90	71.78	79.97	77.18	69.24	60.76	60.71	49.99	49.99	79.97

(d) Number of products = 2

WL(%)	FCFS	FCFS_D	FCFS_I	DJAH(NONE)	DJAH(GR)	DJAH(MTGS)	DJAH(DP)	MIN	MAX
10	29.72	29.72	29.72	27.39	27.39	27.39	27.39	27.39	29.72
20	33.25	33.31	33.23	29.29	29.29	29.29	29.29	29.29	33.31
30	35.80	36.08	35.70	30.97	30.97	30.97	30.97	30.97	36.08
40	37.83	38.75	37.57	32.75	32.69	32.69	32.69	32.69	38.75
50	40.12	42.63	39.51	35.05	34.68	34.68	34.68	34.68	42.63
60	43.66	50.43	42.39	38.69	37.30	37.30	37.27	37.27	50.43
70	50.74	71.58	48.08	45.81	41.32	41.32	41.21	41.21	71.58
80	72.72	179.74	69.75	67.93	49.17	49.17	48.72	48.72	179.74
90	∞	∞	∞	∞	72.62	72.62	74.12	72.62	74.12

(e) Number of products = 6

WL(%)	FCFS	FCFS_D	FCFS_I	DJAH(NONE)	DJAH(GR)	DJAH(MTGS)	DJAH(DP)	MIN	MAX
10	29.71	29.71	29.71	27.38	27.39	27.39	27.39	27.38	29.71
20	33.24	33.27	33.21	29.26	29.30	29.30	29.30	29.26	33.27
30	35.81	35.93	35.69	30.95	31.02	31.01	31.00	30.95	35.93
40	37.93	38.34	37.61	32.75	32.82	32.76	32.71	32.71	38.34
50	40.36	41.40	39.69	35.11	34.97	34.79	34.66	34.66	41.40
60	44.42	46.79	42.99	38.93	37.97	37.50	37.11	37.11	46.79
70	53.20	59.11	50.46	47.04	43.07	41.80	40.70	40.70	59.11
80	91.35	104.77	101.76	77.72	54.50	50.79	47.14	47.14	104.77
90	∞	∞	∞	∞	99.60	81.52	64.19	64.19	99.60

Table 3 Robustness – Missing data on future arrivals

WL(%)	FCFS	FCFS_D	FCFS_I	DJAH(NONE)	DJAH(GR)	DJAH(MTGS)	DJAH(DP)	MIN	MAX
10	29.71	29.71	29.70	27.77	27.77	27.77	27.77	27.77	29.71
20	33.19	33.23	33.18	29.85	29.89	29.88	29.88	29.85	33.23
30	35.67	35.81	35.60	31.60	31.66	31.65	31.64	31.60	35.81
40	37.63	38.04	37.45	33.32	33.37	33.34	33.29	33.29	38.04
50	39.74	40.72	39.35	35.45	35.32	35.25	35.10	35.10	40.72
60	43.01	45.08	42.22	38.73	38.00	37.87	37.40	37.40	45.08
70	49.50	54.34	48.30	45.29	42.38	42.11	40.72	40.72	54.34
80	69.89	82.33	75.85	65.71	52.32	51.88	47.00	47.00	82.33
90	∞	∞	∞	∞	87.28	86.58	63.26	63.26	87.28

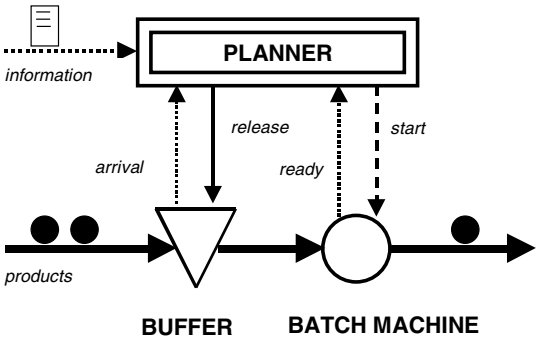


Figure 1 Batch shop

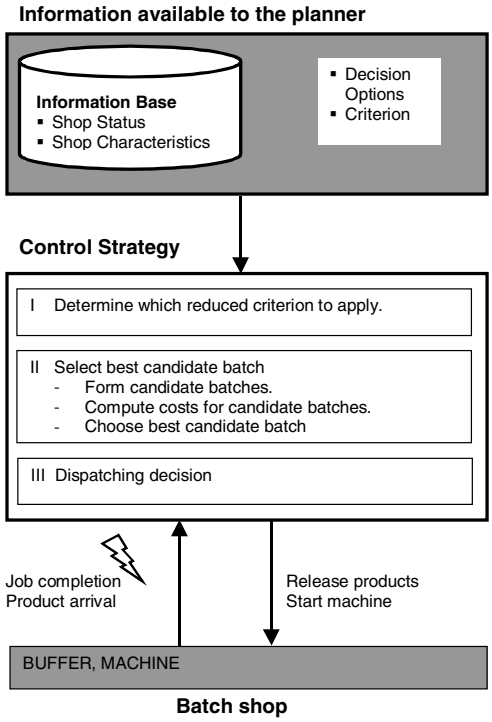


Figure 2 A framework for decision-making

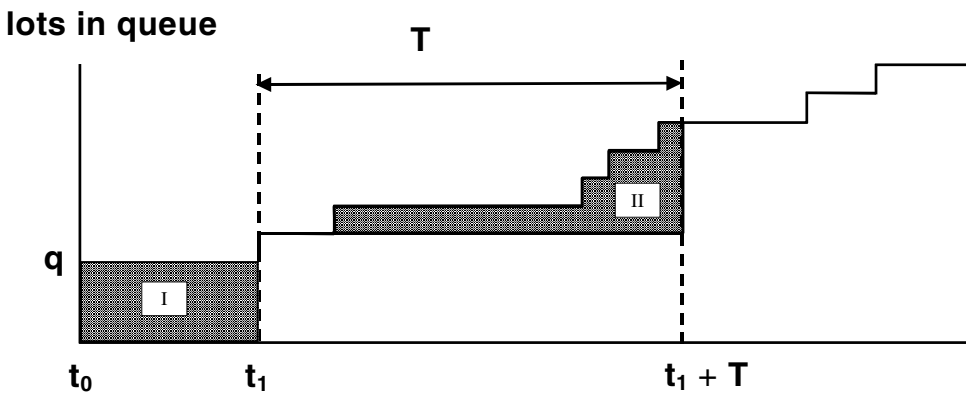
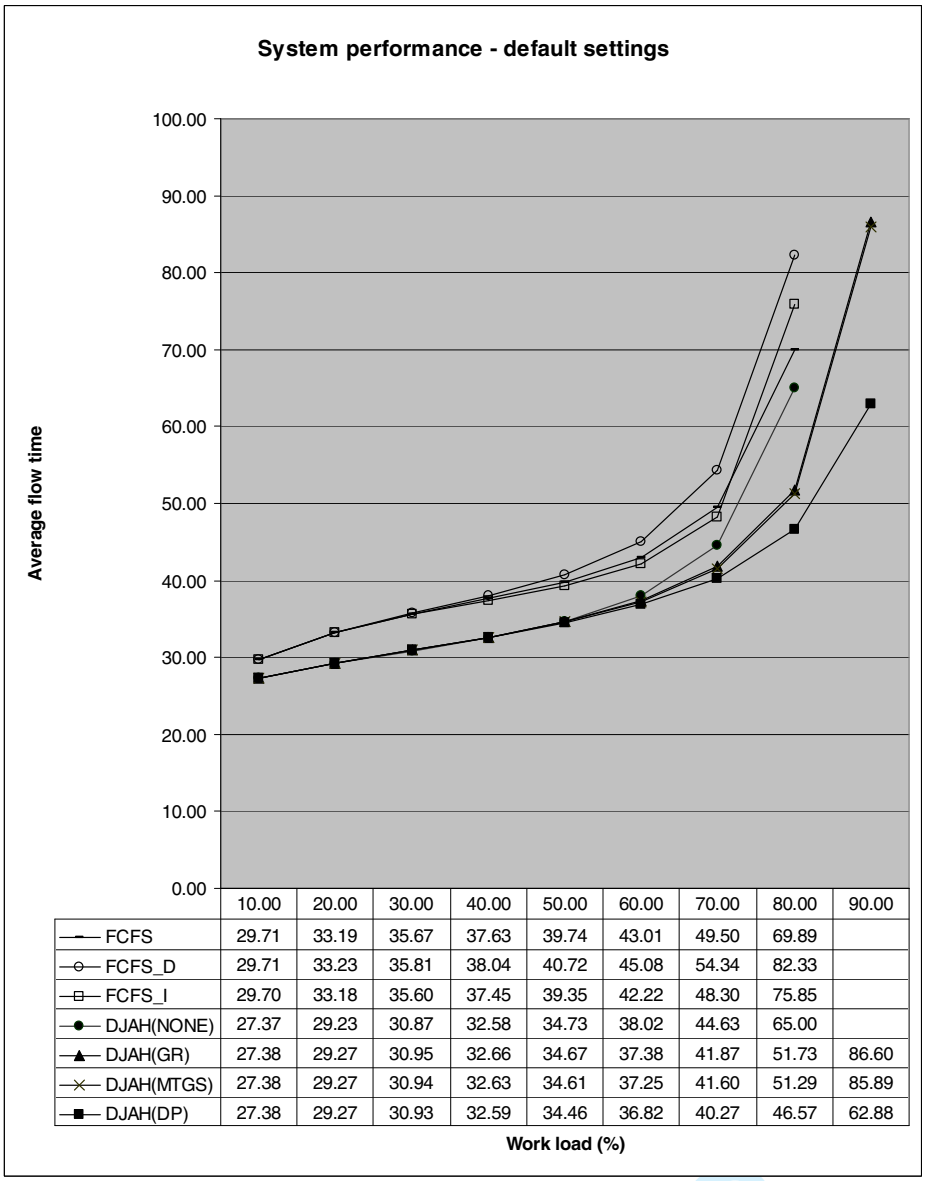


Figure 3 Computation of waiting costs if the machine would be loaded at t_1 with batch B_{t_1} by summing waiting times for those products that:

I Are in queue ($q=\sum_{j\in J} q_j$) and have to wait until t_1

II Arrive during processing



	10.00	20.00	30.00	40.00	50.00	60.00	70.00	80.00	90.00
FCFS	29.71	33.19	35.67	37.63	39.74	43.01	49.50	69.89	
FCFS_D	29.71	33.23	35.81	38.04	40.72	45.08	54.34	82.33	
FCFS_I	29.70	33.18	35.60	37.45	39.35	42.22	48.30	75.85	
DJAH(NONE)	27.37	29.23	30.87	32.58	34.73	38.02	44.63	65.00	
DJAH(GR)	27.38	29.27	30.95	32.66	34.67	37.38	41.87	51.73	86.60
DJAH(MTGS)	27.38	29.27	30.94	32.63	34.61	37.25	41.60	51.29	85.89
DJAH(DP)	27.38	29.27	30.93	32.59	34.46	36.82	40.27	46.57	62.88

Figure 4 Simulation results for default settings