Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

REAL-TIME ADAPTIVE CONTROL OF MULTI-PRODUCT MULTI-SERVER BULK SERVICE PROCESSES

Durk-Jouke van der Zee

Production Systems Design Group
Faculty of Management and Organization
University of Groningen
P.O. Box 800, 9700 AV, Groningen
THE NETHERLANDS

ABSTRACT

Batching jobs in a manufacturing system is a very common policy in most industries. Main reasons for batching are avoidance of setups and/or facilitation of material handling. Batch processing systems often consist of multiple machines of different types for the range and volumes of products that have to be handled. Building on earlier research in aircraft industry, where the process of hardening synthetic aircraft parts was studied, we discuss a new heuristic for the dynamic scheduling of these types of systems. It is shown by an extensive series of simulation experiments that the new heuristic outperforms existing heuristics for most system configurations.

1 INTRODUCTION

"To start the machine now or to wait for a next customer to arrive", that is the question which stresses the essence of the control task for many batch processing systems. The trade-off includes the balancing of logistic costs (e.g. stock keeping, machine utilization) on the one hand and customer service (e.g. lead-times, lead-time uncertainty) on the other hand. As such it is a very common problem found in many industries. The type of systems we study here consist of one or multiple ovens as they can be found in e.g. the aircraft industry and semiconductor manufacturing, cf. Uzsoy et al.(1994), Fowler et al. (1992, 2000), and Hodes et al. (1992). Typically, different types of ovens are available reflecting the required processing conditions (e.g. temperature, pressure), product characteristics (e.g. volume, dimensions) or operating costs (e.g. setup costs) or simply a historical growth pattern. In this article we discuss the development of strategies which help the planner solve the dynamic control problem efficiently. Our primary focus is on look-ahead strategies, i.e., rules that *adapt* their decision to forecast information on *near future arrivals*. Using a procedural approach we propose a new control strategy for planning batch shop activities efficiently.

The paper is organized as follows: in Section 2 we briefly review literature. In Section 3 system characteristics are described in some detail. The construction of the new rule is addressed in Section 4. Section 5 presents the results of a simulation study, indicating the potential of the new rule. Finally, in Section 6, conclusions are summarized.

2 LITERATURE OVERVIEW

The described systems are known in literature as bulk queuing systems. Bulk queuing systems are characterized by the fact that customers arrive in groups and/or are served in groups. In Van der Zee et al. (1997) it is shown how control strategies for bulk queuing systems may be classified according to the amount of information which is known on future arrivals of customers. Three typical situations can be distinguished: (1) No information available, (2) Full knowledge of future arrivals and (3) A limited number of near future arrivals are known or predicted.

The first category of control strategies concerns those strategies that base their decision on local information only. The most important example of such a strategy is the Minimum Batch Size rule (MBS), cf. Neuts (1967). According to this strategy a batch starts service as soon as at least a certain fixed number of customers is present. While the above types of strategies assume zero information on future arrivals, full knowledge of such arrivals is supposed to be available when it comes to deterministic machine scheduling, see e.g. Uzsoy et al. (1994). In this article we focus on the third category of control strategies: the so-called *look-ahead strategies*. Glassey and Weng (1991) were among the first to introduce this type of strategies for (semi-conductor) batch processing systems, which are characterized by the fact that they assume a few near future arrivals to be known and/or predicted. 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 proposed new look-ahead strategies in order to deal with several extensions. Important extensions concern multiple products, multiple servers and alternative criterions for optimization. See Robinson et al. (1995) and Van der Zee et al. (2001) for an overview.

3 SYSTEM DESCRIPTION AND DECISION STRUCTURE

In this section we describe the batch shop studied in some detail. As a starting point for our discussion Figure 1 is used.

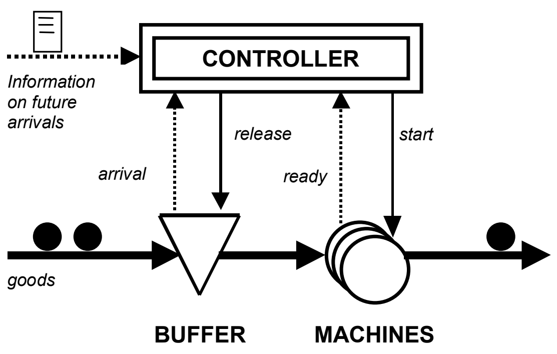


Figure 1: Control of a Batch Shop

Next to a *controller*, the batch shop consists of a *buffer* with an unlimited storage capacity and a number of *servers* (machines). Here we study the case of single arrivals. Machines m, with m = 1..M, process products j, j = 1..N batch-wise. It is demanded that batches are made up of the same type of products, which is related to the required processing conditions. Machines may also be of different types. Differences between machine types are reflected by the required service times ($T_{m,j}$) or the allowed batch sizes $C_{m,j}$ (think of e.g. volume restrictions). The time needed for setup and the transportation of products between the buffer and a server is considered to be included in the service time. Hence, set-up activities are sequence-independent. The service time needed to complete a job is fixed, i.e., it depends on machine (m) and product (j), but it is independent of the batch size.

The above description defines the static, i.e., time independent, shop characteristics. Let us now discuss the decision structure. As a starting point in our discussion we define a framework for decision making (compare Figure 2). Two types of events govern shop dynamics: the actual arrival of products, and their departure, i.e., the completion of a machine job. These events correspond to *decision moments* at which the controller is triggered to initiate some action (compare Figure 1), given availability of both machine(s) and product(s). The decision maker bases his decision on information available on the shop status contained in the *information base*. The information base is supposed to contain the following data on these machines and products: (1) Queue length (q_j) for each product j at the decision moment t_0 (j = 1..N); (2) For each machine m, the moment $t'_m \geq t_0$ when the machine is available (again) (m = 1..M), and (3) For each product j (j = 1..N) the present and successive future arrivals $t_{k,j}$, ordered through the index k according to moment of arrival, up to some specified information horizon.

Essentially, two decision options are open to the decision maker with regard to a specific machine m (that is available at t_0): (1) The scheduling of a job at the decision moment, and (2) Postponement of the scheduling decision to the next decision moment. Note how the scheduling of a job implies the release of products from the buffer and the start of an operation at the machine (compare Figure 1). As a *criterion for optimization* we adopted the minimization of logistic costs per item on the long term. Logistic costs are assumed to be made up of waiting costs (e.g. related to storage of products in the buffer), processing costs and setup costs. The minimization of average flow time may be regarded as an important special case in which setup costs are zero, and waiting costs and processing costs are linear.

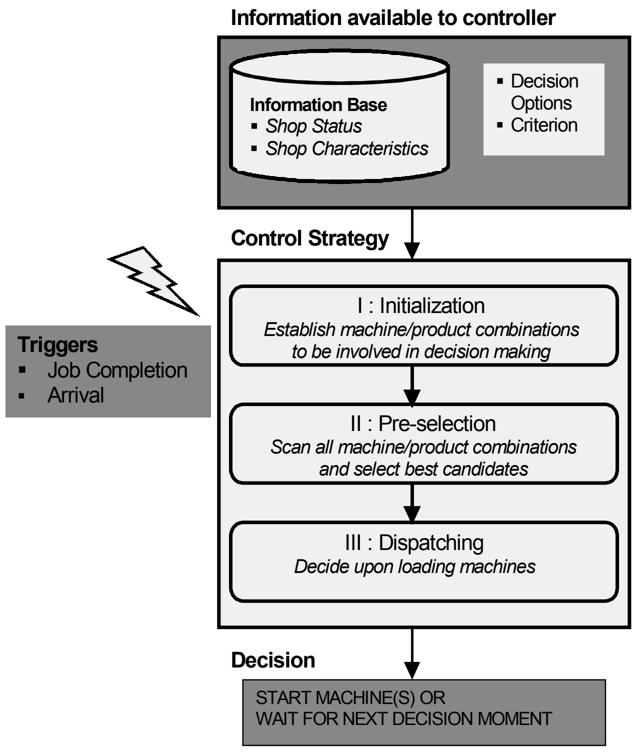

Information available to controller
Information Base
▪ Shop Status
▪ Shop Characteristics
▪ Decision Options
▪ Criterion
Control Strategy
Triggers
▪ Job Completion
▪ Arrival
I : Initialization
Establish machine/product combinations to be involved in decision making
II : Pre-selection
Scan all machine/product combinations and select best candidates
III : Dispatching
Decide upon loading machines
Decision
START MACHINE(S) OR WAIT FOR NEXT DECISION MOMENT

Figure 2: A Framework for Decision Making

4 RULE CONSTRUCTION

The above description of the shop floor sets the context for the decision problem. Let us now consider this problem in some more detail. We assume a procedure for decision making which concerns three sequential steps: initialization, pre-selection and dispatching. *Initialization* is meant to establish the set of machine/product combinations, which is to be involved in decision making. In the second step, *pre-selection*, the aim is to reduce the combinatorial problem by selecting the most promising machine/product combinations. *Dispatching* concerns the question whether machines available at the decision moment should be loaded right now, or whether it is better to wait for a next decision moment. The trade-off involves a comparison of logistic costs for both possibilities for each of the selected machine/product combinations.

Above we characterized control strategies in general terms. Let us now adopt this new vocabulary to discuss the *new Dynamic Scheduling Heuristic* (DSH) in general terms. For a more detailed discussion see Van der Zee et al. (2001):

I: Initialization: No a priori restrictions are assumed with regard to the set MP of machine/product combinations to be involved in the decision.

II: Pre-selection: For the new heuristic a scanning procedure has been developed, in which all machines are involved in the decision, in principle. The scanning procedure aims at selecting the "best" product for each of the machines available at the decision moment t_0, given the criterion for optimization. The procedure consists of two phases: (A) The building of a *virtual schedule* (VS), which assigns products to machines, and (B) The reduction of the schedule to those machines available at t_0 (VSR). By building a virtual schedule involving *all machines* it is striven for the best use of machines, given their characteristics and the information on products. The schedule is called "virtual" because its meaning is restricted to a certain decision moment. This is a direct consequence of its dependence on a very limited knowledge on future product arrivals. VS is built iteratively in four steps:

1. Mark those elements of MP (see I: Initialization) for which there is a full load and for which the machine is available at t_0. Call them MP_1.
2. Compute throughput for all elements in MP.
3. Sort MP. Give priority to MP_1, then consider maximum throughput.
4. Reduce MP by eliminating the machine/product combinations where the respective machine and/or product has been part of a combination before (following the sorting order).

In the first step full loads are considered for the same reasons as explained earlier. As a second criterion for prioritizing machine/product combinations a throughput related rule is used. As a definition of throughput we use:

$$TH_{m,j} = \frac{min(q_j, C_{m,j})}{t'_m + T_{m,j} - t_0} \tag{1}$$

Note how throughput in equation (1) is influenced by queue length (q_j), machine availability (t'_m) and capacity ($C_{m,j}$) and service times ($T_{m,j}$). Given the sorting of step 3, in step 4 it is accounted for the fact that a machine can only be assigned one job at the same time. Unique combinations are searched for, trying to realize a high throughput. The resulting set equals VS.

In phase (B) VSR is simply found by considering only those machine/product combinations in VS for which the machine is available at t_0. There is one exception to this rule, however, in case the first element of VS concerns a full load, only this element is passed on to the Dispatching step. Subsequently, the Pre-selection is re-run, succeeding the update of the information base concerning the full load (which of course will start with certainty). By allowing for this loop it is reckoned with the fact that a full load for a product may be an indication of long queue length and/or a high arrival rate, which in turn may call for the assignment of multiple machines for the same product.

III: Dispatching: For each machine/product combination (m,j) in VSR it is decided whether a job should start at the decision moment. For those cases where there is no full load, estimated logistic costs per item (TC) are compared for the situation in which the job would be started right away (t_0) and the situation in which one waits for $t_{1,j}$, i.e., the next arrival (for the same product j). Using the notation introduced above, TC has been defined as:

$$\begin{gathered} TC_{m,j}(t_0) = \frac{TV_{m,j}(t_0)}{q_j} \\ TC_{m,j}(t_{1,j}) = \frac{TV_{m,j}(t_{1,j})}{min(q_j + 1, C_{m,j})} \\ \text{with:} \\ TV_{m,j}(t_0) = \Phi_{m,j} + \sum_{t_0 < t_{k,j} < H^0_{m,j}} (H^0_{m,j} - t_{k,j}) \\ TV_{m,j}(t_{1,j}) = \Phi_{m,j} + q_j(t_{1,j} - t_0) + \sum_{t_{1,j} < t_{k,j} < H^1_{m,j}} (H^1_{m,j} - t_{k,j}) \\ \text{where } H^0_{m,j} = t_0 + T_{m,j}; H^1_{m,j} = t_{1,j} + T_{m,j} \end{gathered} \tag{2}$$

In equation (2) $TV_{m,j}$ computes setup costs ($\Phi_{m,j}$) and estimated waiting costs for both alternatives. Waiting costs are determined by computing waiting times for products j up to the cost horizon $H^d_{m,j}$. As a natural consequence of the fact that a machine/product combination is studied in isolation, $H^d_{m,j}$ is set to the moment machine m becomes available again.

5 SIMULATION STUDY

In order to demonstrate the potential of the new Dynamic Scheduling Heuristic a simulation study was carried out. In this section the design of the simulation study and its outcomes are discussed. The experiments concern a job shop that consists of four machines in parallel. Two series of experiments were carried out (I,II). In Table 1 an overview is given of these series.

Table 1: Design of the Simulation Study

No	Factor	I	II
1	No Machines/No Products	4/1,2,4,6,8	
2	Product Mix	Equal	
3	Capacity per Product (C_j)	5	
4	Capacity per Machine (C_m)		7,7,3,3
5	Service Time per Product (T_j)		25
6	Service Time per Machine(T_m)	20,20, 33 1/3,33 1/3	
7	Work Load (w)	0.3;0.6;0.9	
	Number of experiments	75	75

We test the performance of the Minimum Batch Size Rule (adapted for multi server environments, cf. Van der Zee 2001), the existing Next Arrival Control Heuristic (Fowler 1992,2000) and Dynamic Job Assignment Heuristic (Van der Zee 1997) and the DSH heuristic for five product/machine combinations. The adapted Minimum Batch Size rule (MBSX) prioritizes the longest queue. As long as products are available a new batch is loaded right away. As such it makes no use of knowlegde on future arrivals. The MBSX rule serves as a benchmark in this study. The Next Arrival Control Heuristic (NACH) and Dynamic Job Assignment Heuristic (DJAH) address shop configurations consisting of multiple identical machines. Next to the DJAH rule we study a truncated version of the DJAH rule, named T-DJAH. It does only consider next arrivals if they are within the cost horizon, i.e., the next product has to arrive before a next machine becomes available. In this way it is dealt with situations in which DJAH appears to be somewhat too greedy. These cases mainly concern situations where multiple products have to be handled at low traffic intensities. (in our examples 30%). Typically, queue length per product is small under these circumstances. An additional item may therefore have a very significant impact on estimated waiting costs *per item*. As a consequence the dispatching decision may be postponed even in those cases where the next arrival is not expected "soon". Although both look-ahead strategies mentioned may simply be adapted to deal with alternative machine types (see Van der Zee 2001), they do not consider machine related characteristics in shop optimization. The choice for alternative rules allows us to gain insight in the relative performance of the new DSH rule relating to its use of information on future arrivals and shop configuration, i.e., machine and product characteristics.

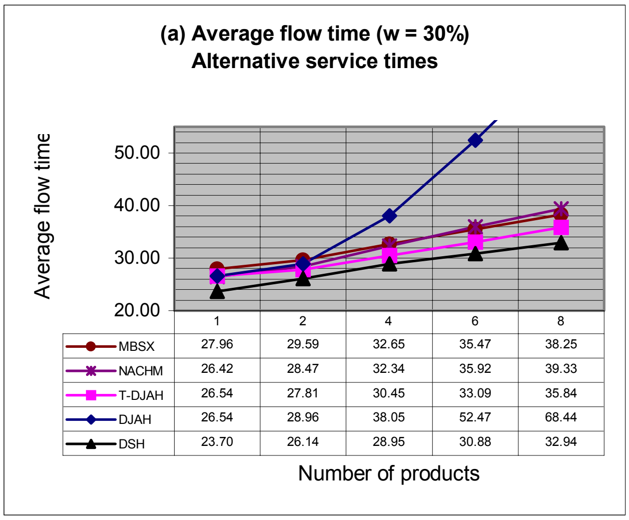
In our simulation study we considered Poisson arrivals. The criterion for optimization that was adopted, was the minimization of average flow time. The software package, which was used to carry out the simulation experiments, is Simple++ (Technomatix). The principles of object orientation underlying Simple++ make it a flexible and efficient tool for model building and setting up experiments.

Results for the simulation study are presented in Figure 3. In the first example (I) machines can be classified in two types: "fast" and "slow". Fast machines require a service time of 20 time units for each product, while slow machines require 331/3 time units, but all machines have the same capacity 5. In the same way an alternative setting (II) is studied, where a distinction is made in two "large" machines with a maximum batch size of seven products and two "small" machines which allow for a batch size of three products.

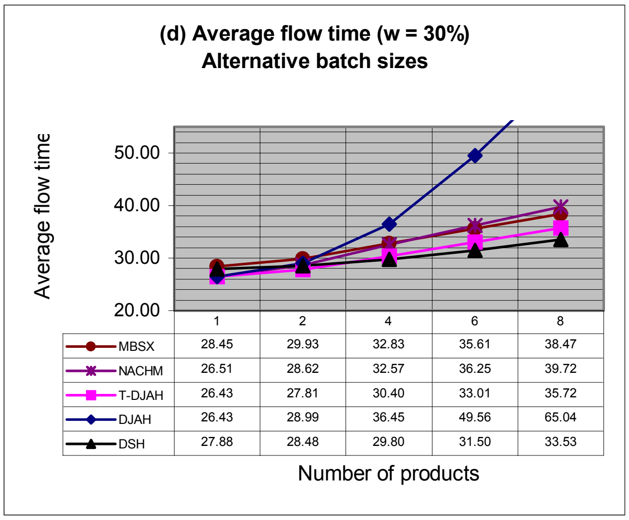
Simulation results for these experiments are presented in Figures 3a,b,c and 3d,e,f respectively. The results in the Figure 3 clearly indicate the improvement of system performance that is obtained if a look-ahead strategy is used instead of MBSX, which bases its decision on local information only. In general, improvements as a percentage of average flow time are large for low traffic intensities and smaller for high traffic intensities. This is due to the saturation effect, which was described by Glassey et al. (1993):

- A large fraction of the time, decision options are limited to the loading of full batches. As a consequence the 'look-ahead' heuristics and the 'greedy rule', i.e., MBSX, more often make the same decision.
- The larger the queue, the less likely it is that the total waiting time will be reduced by delaying the start until the next arrival.

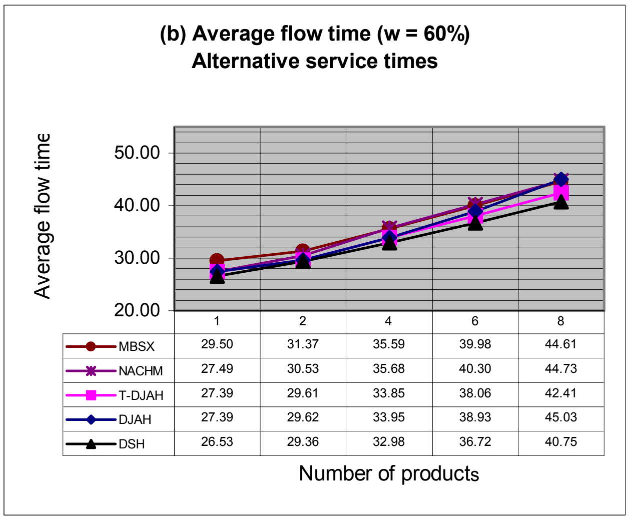
The results for series I show that DSH realizes an improvement of system performance of up to 10% for low ar rival rates and up to 5% for moderate traffic intensities.



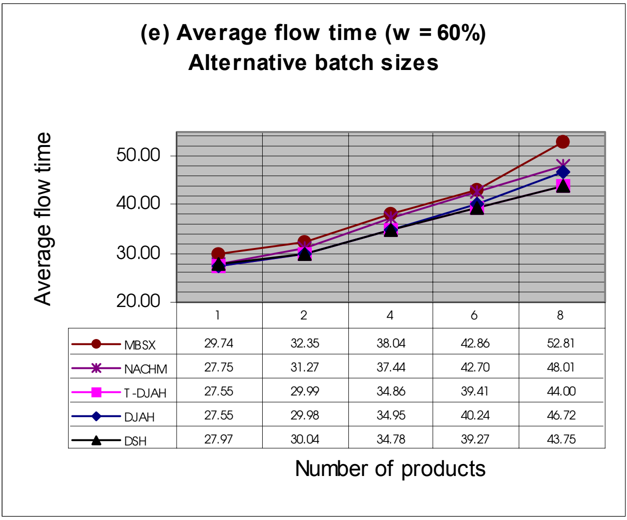
	1	2	4	6	8
● MBSX	27.96	29.59	32.65	35.47	38.25
✱ NACHM	26.42	28.47	32.34	35.92	39.33
■ T-DJAH	26.54	27.81	30.45	33.09	35.84
◆ DJAH	26.54	28.96	38.05	52.47	68.44
▲ DSH	23.70	26.14	28.95	30.88	32.94



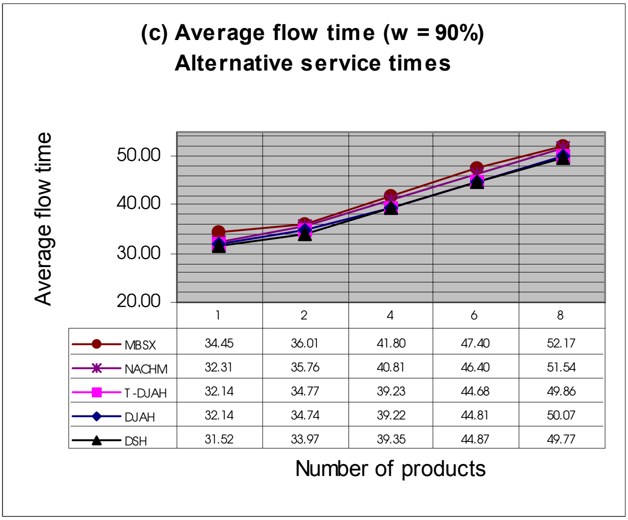
	1	2	4	6	8
● MBSX	28.45	29.93	32.83	35.61	38.47
✱ NACHM	26.51	28.62	32.57	36.25	39.72
■ T-DJAH	26.43	27.81	30.40	33.01	35.72
◆ DJAH	26.43	28.99	36.45	49.56	65.04
▲ DSH	27.88	28.48	29.80	31.50	33.53



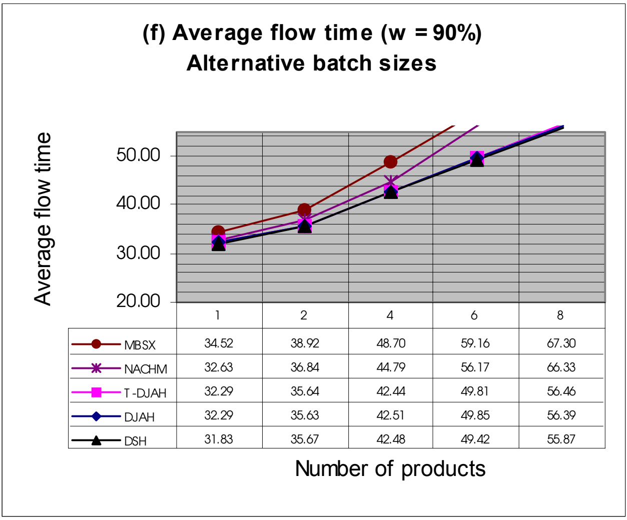
	1	2	4	6	8
● MBSX	29.50	31.37	35.59	39.98	44.61
✱ NACHM	27.49	30.53	35.68	40.30	44.73
■ T-DJAH	27.39	29.61	33.85	38.06	42.41
◆ DJAH	27.39	29.62	33.95	38.93	45.03
▲ DSH	26.53	29.36	32.98	36.72	40.75



	1	2	4	6	8
● MBSX	29.74	32.35	38.04	42.86	52.81
✱ NACHM	27.75	31.27	37.44	42.70	48.01
■ T-DJAH	27.55	29.99	34.86	39.41	44.00
◆ DJAH	27.55	29.98	34.95	40.24	46.72
▲ DSH	27.97	30.04	34.78	39.27	43.75



	1	2	4	6	8
● MBSX	34.45	36.01	41.80	47.40	52.17
✱ NACHM	32.31	35.76	40.81	46.40	51.54
■ T-DJAH	32.14	34.77	39.23	44.68	49.86
◆ DJAH	32.14	34.74	39.22	44.81	50.07
▲ DSH	31.52	33.97	39.35	44.87	49.77



	1	2	4	6	8
● MBSX	34.52	38.92	48.70	59.16	67.30
✱ NACHM	32.63	36.84	44.79	56.17	66.33
■ T-DJAH	32.29	35.64	42.44	49.81	56.46
◆ DJAH	32.29	35.63	42.51	49.85	56.39
▲ DSH	31.83	35.67	42.48	49.42	55.87

Figure 3: Simulation Results - Minimizing Average Flow Time per Item for a Job Shop Consisting of Non-Identical Machines for Work Loads of 30%,60% and 90%. Differences Concern Service Times (a,b,c) and Allowed Batch Sizes (d,e,f)

Here one clearly observes the effects of a pre-selection procedure that aims at improving throughput by giving priority to faster machines. Confirmation of this proposition can be found in Table 2. The table presents the percentage of products handled per machine type. The results in Table 2 clearly indicate how DSH improves on system performance by making good use of faster machines. Improvements are largest for situations in which a small number of products are processed at low traffic intensities. This is a direct consequence of the larger flexibility following from excess machine capacity that can be efficiently filled, given the relatively large queue lengths (for small N).

Table 2: Percentage of Products Processed per Machine Type (fast/slow)

N = 1

w	30%		60%		90%	
Rule	fast	slow	fast	slow	fast	slow
MBSX	65.2	34.8	64.3	35.7	62.9	37.1
NACHM	63.2	36.8	64.0	36.0	62.9	37.1
T-DJAH	62.0	38.0	63.9	36.1	62.7	37.3
DJAH	62.0	38.0	63.9	36.1	62.7	37.3
DSH	97.2	2.8	74.7	25.3	63.2	36.8

N = 4

w	30%		60%		90%	
Rule	fast	slow	fast	slow	fast	slow
MBSX	63.2	36.8	63.2	36.8	62.3	37.7
NACHM	63.2	36.8	63.0	37.0	62.6	37.4
T-DJAH	63.0	37.0	62.6	37.4	62.3	37.7
DJAH	69.7	30.3	62.4	37.6	62.3	37.7
DSH	85.3	14.7	71.1	28.9	63.8	36.2

N = 8

w	30%		60%		90%	
Rule	fast	slow	fast	slow	fast	slow
MBSX	63.2	36.8	62.8	37.2	63.5	36.5
NACHM	63.2	36.8	62.8	37.2	63.2	36.8
T-DJAH	62.8	37.2	62.2	37.8	62.2	37.8
DJAH	74.5	25.5	62.6	37.4	62.1	37.9
DSH	80.8	19.2	71.8	28.2	64.0	36.0

Let us now discuss the influence of maximum batch size on system performance. Results in Figure 3 indicate that DSH outperforms the other heuristics for most settings. Only for N=1 NACHM, DJAH perform best at low and moderate traffic intensities. In these situations DSH tends to focus too much on a few machines (equal to the number of products), while neglecting the possibility of involving other machines.

6 CONCLUDING REMARKS

In this article the new Dynamic Scheduling Heuristic (DSH) for on-line scheduling of multi-server batch operations has been discussed. The heuristic distinguishes itself from existing control strategies by taking an integral approach in which it builds a schedule for all machines, instead of just focussing at the machines available at the decision moment. In this way complex situations may be addressed where the controller has to find an efficient match between product and machine characteristics and their availability. It has been shown by an extensive simulation study that the new approach results in significant improvements in system performance.

REFERENCES

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

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

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

Glassey, C.R., F. Markgraf, and H. Fromm, H..1993. Real time scheduling of batch operations. In *Optimization in Industry: mathematical programming and modeling techniques in practice*, ed. T.A. Ciriani, and R.C. Leachman, Chichester:Wiley: 113-137.

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

Neuts, M. F. 1967. A general class of bulk queues with Poisson input. *Annals of Mathematical Statistics* 38: 759-770.

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.

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

Van der Zee, D. J., A. van Harten, and P. C. Schuur. 1997. Dynamic Job Assignment Heuristics for Multi-Server Batch Operations-A Cost-Based Approach. *International Journal of Production Research* 35: 3063-3093.

Van der Zee, D. J., A. van Harten, and P. C. Schuur. 2001. On-Line Scheduling of Multi Server Batch Operations. *IIE Transactions*, 33: 569-586.

AUTHOR BIOGRAPHY

DURK-JOUKE VAN DER ZEE is an assistant Professor at the Faculty of Management and Organization at the University of Groningen. He received M.S. and Ph.D degrees from the University of Twente, The Netherlands. His research interests include simulation methodology and applications, shop floor control systems and flexible manufacturing systems. His e-mail and web addresses are `<d.j.van.der.zee@bdk.rug.nl>` and `<www.bdk.rug.nl/medewerkers/d.j.van.der.zee>`.