SCHEDULING SEMICONDUCTOR MANUFACTURING PLANTS*

P. R. Kumar†

Abstract

In many large systems, such as manufacturing systems and communication networks, whenever a resource becomes available, one has to decide which of several tasks contending for its attention should be performed next. Such problems are called "scheduling" problems. They are control problems of immense economic importance, and have often been formulated and addressed in an open–loop setting.

To both illustrate the types of problems encountered and to serve as focus, in this article we will address scheduling problems in a technological area of much topical interest – semiconductor manufacturing. Being of relatively recent origin, and organized differently from more traditional manufacturing systems such as flow shops and job shops, they are relatively less explored. They are also of significant economic interest, and much in the public limelight, and thus a fertile area for systems and control researchers. We provide an account of some problems in the area, as well as some suggested solutions.

1 Introduction

There are a large number of control problems with immense economic ramifications, which have traditionally been addressed in the operations research and industrial engineering communities. This area is however not traditionally called "control," but is called "scheduling." Often, scheduling problems have been formulated and addressed in an open–loop setting. Given a (typically) large entity, such as a factory or a network, one has to usually make decisions throughout the plant's lifetime. These decisions typically involve issues such as

*The research reported here has been supported in part by the National Science Foundation under Grant Nos. NSF–ECS–90–25007 and NSF–ECS–92–16487, and the Joint Services Electronics Program under Contract No. N00014–90–J–1270.

†Department of Electrical and Computer Engineering, and the Coordinated Science Laboratory, University of Illinois, 1308 West Main Street, Urbana, IL 61801, Email: prkumar@decision.csl.uiuc.edu.

which of several competing tasks to take up next. The goal of control (i.e., scheduling) is often to run the system more efficiently, for example, to produce parts or deliver messages more expeditiously. This is an area of much current interest, providing many practically important and challenging problems, and is well deserving of close attention from systems and control researchers.

Here we will address scheduling problems in a technological industry which is of much topical interest – semiconductor manufacturing. Semiconductor manufacturing is a relatively recent enterprise, and, as we shall see, it is organized quite differently from more traditional manufacturing operations such as assembly lines or job shops. It provides many economically important challenges.

This brings us to the topic of the article – scheduling manufacturing plants. We begin with a brief description of the semiconductor manufacturing process, and exhibit the structure of the resulting manufacturing plant. We describe the scheduling problems of interest. Based on two fundamental laws from queueing theory, we develop some new scheduling policies based on smoothing the fluctuations of all the flows in the plant. We provide another formulation of our policies as attempting to alleviate total downstream shortfall from each buffer, that is reminiscent of classical integral compensation. We also report briefly on the performance of the scheduling policies, based on an extensive comparative simulation study on large scale plant models.

2 How Semiconductor Wafers are Fabricated and the Resulting Re-Entrant Manufacturing System

One starts with a silicon wafer, a few inches in diameter. The production process consists of imprinting several layers of chemical patterns on the wafer, and the final end product can be regarded as a multilayered sandwich.

Figure 1: Semiconductor manufacturing process.

The processing is done layer by layer; see Figure 1. Each layer in turn requires several steps of individual processing, such as deposition, photolithography, etching, etc. Moreover, many of the steps are repeated at several of the layers.

The machines to perform these individual steps are very expensive. Currently a state of the art plant costs about a billion dollars, and this cost is expected to increase dramatically as feature sizes decrease. The machines are not replicated but revisited by wafers for processing at different layers.

What this gives rise to is a plant of the form shown in Figure 2. It shows a single route followed by wafers. The distinguishing characteristic is that lots re-visit several machines at several stages of their life. This type of a manufacturing system is called a *re-entrant line*.

The main consequence of the re-entrant nature is that wafers at different stages of their life have to compete with each other for the same machines – reminiscent of cars at a traffic light. Thus, wafers can and do spend a considerable portion of their time simply waiting for machines, rather than actually being processed. The manufacturing lead time, the time taken by wafers in their passage from the entry to the exit of the plant, is called the *cycle-time*. It is roughly of the order of 60 days. On the other hand, the actual time that a wafer

is actively being processed may only be about 240 hours, i.e., 10 days. The ratio Cycle-Time/Total Processing Time is called the cycle-time multiplier. In the above example, it is 6. A key goal in semiconductor manufacturing is to reduce the cycle time, i.e., to reduce cycle-time multiplier as close to one as possible.

Figure 2: An aggregated model of a semiconductor manufacturing plant.

This has several important economic ramifications. In R&D Fabrication Lines used for developing new products, reducing the cycle-time reduces the time to design and debug products. Reducing the time to market is very important in the current rapidly changing technological environment.

In production lines, reducing the cycle–time improves the responsiveness to customers. Moreover, there is a result, Little's Law (which by its pervasiveness can be called the Zeroeth Law of Queueing Systems) which states that for a given plant throughput, i.e., production rate, the mean number of wafers in the plant is proportional to their mean cycle-time. Thus, reducing the cycle-time reduces the *work-in-process*. Hence, reducing the cycle–time reduces not only the capital invested in the work-in-process,, but also the clutter on the plant floor.

Finally, there is a technological advantage to reducing cycle-time. The smaller the time that wafers are exposed to contaminants on the factory floor, the greater is the yield of good wafers. This is an important concern in this industry, particularly so whenever new technology is being introduced and yield is typically very low.

Another key concern is to reduce the *variance* of cycle-times. Since the plant may be subjected to several random effects, e.g., random arrivals, random processing times, random machine failures, etc., the cycle-time is also random. If the variance of the cycle-times is small, then one can safely predict when a wafer will complete production and exit the plant. This allows an improved ability to meet due-dates reliably, and thus allows greater coordination with further downstream operations on wafers, e.g., dicing, assembly, etc.

One could summarize the two issues in the following way. A small mean cycle-time allows the plant manager to *make* good promises, while a small variance of cycle-times allows her to *keep* her promises.

Usually, wafers travel through the system in groups called "lots." Each lot may consist of about 20 wafers. For simplicity, we shall call each lot as a "part."

How does one attain these objectives? There are two mechanisms for control. First, one can regulate the arrivals of new parts into the system. Thus one can decide when each new part is "released" into the plant. The control policy for making such decisions is called the *release policy*. Clearly, the release policy will need to satisfy some constraints, e.g., maintaining an average rate of releases.

Now consider the parts already in the plant. At a given machine there may be more than one part waiting for processing by the machine. Then one has to decide which part is to be

processed next, when the machine becomes idle. The control policy governing such decisions is called a *scheduling policy*.

3 How Semiconductor Manufacturing Plants are Different from More Traditional Plants

Broadly speaking, manufacturing systems can be divided into two categories – flow shops and job shops (though of course there are others types of system, e.g., for shipbuilding).

Good examples of flow shops are automobile assembly lines. Here, parts (i.e., cars) have an acyclic (straight line) route through the plant, never visiting any station more than once; see Figure 3. Such flow shops are typically dedicated to producing a single type of part, and in high volume.

Figure 3: A flow shop.

The other major category of manufacturing systems is job shops; see Figure 4. These are

Figure 4: A job shop.

systems which contain a variety of machines of different types. Orders may come, in small volumes, for custom-made parts. Each custom order may involve a different route, which

may revisit a machine, and the aggregate of routes over different orders may look random. Such systems are common in the metal cutting industry.

With the advent of semiconductor manufacturing, we have a new type of system that does not fit in either of the two classes above. Unlike flow shops, the route is re-entrant. However, unlike job shops, often there is more structure and less randomness in the routes.

Clearly any theory of control or scheduling will need to focus on the type of system at hand. Here we will develop scheduling policies which exploit the special structure of re-entrant lines.

4 Little's Law and the Myopically Optimal Last Buffer First Serve Policy

Consider any system, a "black box," into which parts arrive, and from which parts eventually exit. Such a system could be a manufacturing plant or a communication network. Let us suppose that parts arrive to the system at an average rate of λ parts per time unit. Also suppose that parts spend an average amount of time W in the system. Finally, let us suppose that the average number of parts in the system is L, as shown in Figure 5.

Figure 5: Little's Law: $L = \lambda W$.

Then, the average number in the system L, and the average time spent by parts in the system W, are related by the formula,

$$L = \lambda W. \tag{1}$$

This is called Little's Law (see [1]). Due to its widespread applicability, one could call it the Zeroeth Law of Queueing Systems.

Since the arrival rate λ is fixed, we see from (1) that to reduce the mean time W spent in the system, i.e., the mean cycle-time, it suffices to reduce the mean number of parts L in the system. Therefore, Little's Law tells us that the goal of reducing the mean cycle–time is equivalent to the goal of reducing the mean number of parts in the system.

Figure 6: The Last Buffer Priority First Serve Policy: Parts in buffer b_6 are given higher priority than parts in buffer b_3.

One can design a myopically optimal scheduling policy for this equivalent goal. Consider the re-entrant line shown in Figure 6. Let us suppose that parts at the i-th stage of their processing are stored in a buffer labeled b_i. Note that the buffers are labeled in the order that they are visited. Suppose that Machine 3 is idle, and it has to decide whether to process a part in buffer b_3 or a part in buffer b_6. If it processes the part in b_3, then it will end up transferring a part from b_3 to b_4, without affecting the *number* of parts in the system. However, if it processes a part from b_6, then it can reduce the number of parts in the system by one.

Choosing to process a part from b_6 rather than b_3 is "myopic," since it only seeks to achieve an immediate gain by reducing the number of parts in the short–term, rather than by considering the long–term consequences of such expediting.

In any case, by extending this myopic line of reasoning, it is better to give priority to b_4 over b_1 at Machine 1, and to b_5 over b_2 at Machine 2. We can summarize this *buffer priority policy* by rank ordering the buffers as $\{b_6, b_5, b_4, b_3, b_2, b_1\}$, and giving priority at a machine according to this ordering. (Note that even though the relative ordering of buffers at *different* machines, e.g., b_3 and b_4, is not necessary and never used; it is only done so for the convenience of having just one list of all buffers, rather than three separate lists, one for

each machine). Since this ordering is the reverse of the order in which the parts are visited, we call it the *Last Buffer First Serve* (LBFS) scheduling policy.

As noted above above the LBFS policy is a myopically designed policy, and is not optimal. Indeed, we well know from dynamic programming (see [2]) that an optimal policy must strike a balance between short term rewards and long term costs. In our specific context, suppose Machine 1 is idle, and there are no parts in either b_1 or b_4. Then it could be advantageous to give Machine 1 some work to do, especially if it is a bottleneck. Thus, one may prefer to work on a part in b_3 and send it to b_4. However, such compromises are delicate and very difficult to quantify. We will therefore turn to another approach – smoothing fluctuations, later in this article, to obtain good scheduling policies.

Before we return to the problem of designing good scheduling policies, we should note that there are a whole variety of questions that one may ask about systems and specific scheduling policies. For example, is a given scheduling policy stable? How can one quantify its performance? Unfortunately, the now classical works on networks [3, 4, 5, 6], are able to determine explicit closed–form expressions for the steady-state performance, only for certain special types of systems and scheduling policies. The systems of interest here do not appear to possess any simple (or complex) explicit solutions. Thus, recent work has focussed on developing a theory which allows the use of computational methods which can provide guarantees of stability and performance [7, 8, 9, 10, 11]. This may be called the *descriptive* theory. We refer the reader to [12] for an account of some results in this direction.

Let us now proceed with the *prescriptive* aspects of the problem, i.e., how to *control* such systems.

5 Reducing the Variance of Lateness and Variance of Cycle-Time

We shall first consider the problems of reducing the *variance of lateness* and *variance of cycle-time*. Then we will report very briefly from an extensive simulation study based on large plant models [13]. Last, we shall attempt to explain what we have observed – a la

physics, and try to take advantage of it – a la engineering.

Let us denote a part by π. Let us suppose that each part π has a *due date* $\delta(\pi)$ stamped on it. Further, let us denote by $\alpha(\pi)$ the time that the part has arrived at the plant, and by $e(\pi)$ the time that it completes production and leaves the plant.

The *lateness* of the part π is defined as

$$\ell(\pi) := e(\pi) - \delta(\pi),$$

i.e., the time by which it misses its due date. Consider the problem of minimizing the *variance of lateness.*

First let us denote by ζ_i the mean time that parts spend in going from buffer b_i in the plant to exit; see Figure 7.

Figure 7: The mean remaining time ζ_i for a part π in buffer b_i.

Thus, ζ_i is the *mean remaining time in the system for a part in buffer* b_i.

Suppose that at time t, part π is located in buffer b_i. Then $\delta(\pi) - t$ is the time left till the due-date of the part. On the other hand, ζ_i is the time that the part is expected to still spend in the plant. Thus the "slack,"

$$s_1(\pi) := \delta(\pi) - t - \zeta_i, \tag{2}$$

measures the relative urgency of the part π. If it is negative, the part is expected to be late, while, if it is positive, the part is expected to be early. In fact $s_1(\pi)$ is just the expected earliness of the part.

Clearly it appears eminently reasonable to give priority to the part with the smallest slack. Such a policy is called a *Least Slack* (LS) scheduling policy. Now let us ponder in what quantitative sense this is reasonable. This policy essentially attempts to make all parts

equally late or equally early. Clearly, if all parts are about equally late, the variance of lateness is small, and similarly if all parts are about equally early, then again the variance of lateness is small. Thus, the policy giving priority to the part with smallest value of the slack (2) actually strives to reduce the *variance of lateness*.

Note now that the current time t in (2) is common to all parts, and hence we can drop it from comparisons. This shows that we can more simply define the slack as,

$$s_1(\pi) := \delta(\pi) - \zeta_i, \tag{3}$$

and give priority among the parts contending for an idle machine to that part whose slack is smallest. We call this the *Fluctuation Smoothing Policy for Variance of Lateness* (FSVL).

Now let us turn to the problem of reducing the *variance of the cycle-time*. Clearly, if the due-date $\delta(\pi)$ had been *set* equal to the *arrival time* $\alpha(\pi)$, then

$$\text{Cycle-time } = e(\pi) - \alpha(\pi) = e(\pi) - \delta(\pi) = \text{ Lateness.}$$

Thus, if we redefine the slack in (3) as,

$$s_2(\pi) := \alpha(\pi) - \zeta_i,$$

then the corresponding Least Slack policy should reduce the variance of the cycle-time. We call it the *Fluctuation Smoothing Policy for the Variance of Cycle-Time* (FSVCT).

How well does this policy perform? We have conducted (see [13] for more details) extensive simulation tests of this policy on some plant models, including models of a Hewlett Packard Research and Development Fabrication Line developed by Wein [14]. This model includes 172 steps of processing (buffers) at 24 stations, some of which contain more than one machine. The machines are subject to random failures and repairs.

Included in the comparative testing were a whole range of scheduling policies (including all the scheduling policies tested in [14], as well as others). To give a reader a feeling for what types of scheduling policies are employed, we now provide a brief description of some of them, along with their rationales. The well known FIFO (aka FCFS) policy gives priority to the part that arrived to the station earliest. A slight modification gives the FIFO+ policy,

which attempts to avoid idleness at the next station by giving priority to a part, if one exists, which is headed for a station having less than 4 parts; if no such part exists, it uses FIFO. The LWNQ policy (Least Work Next Queue) addresses the issue of avoiding idleness more directly by simply giving priority to the part whose next station has the least amount of work per machine (since there may be more than one machine). We have earlier described the myopic LBFS policy, which is also called the SRPT policy since it gives priority to the part that has the smallest expected remaining processing time in the system. The SRPT++ policy is a slight modification which attempts to forestall idleness, by giving priority to a part which is headed for a station having less than 4 parts, selecting among these using SRPT if there is more than one such part; if no such part exists, it uses SRPT. The SRPT+ policy is a slight variant of SRPT++ which uses FIFO when there are no such priority parts. Very slightly different from the SRPT policy is the EDD policy, the difference arising solely from the fact that occasionally a part may overtake another part if they are both being served by different machines at the same station, since the processing times are random. Under EDD the part that arrived earlier is still given priority, even it is in an earlier buffer. Yet other scheduling policies attempt to give priority to the parts going to the bottleneck machines, where a bottleneck is defined as any machine which is either utilized or broken down more than 90% of the time. For example the FGCA policy gives priority to a part, if it is going to visit a bottleneck machine in its next two stops (selecting among these using FIFO); if no such part exists, it uses FIFO. Other policies use special rules at the bottleneck stations. The CYCLIC policy cycles through the buffers at a bottleneck, serving one part in each buffer, and skipping a buffer only if it is empty. The LTNV policy gives priority at a bottleneck to a part that has the longest expected processing time until its next visit to the same station (using FIFO at all non–bottleneck stations).

In Figure 8 we present the averages of 20 simulation runs on one plant model, for each policy, when the arrivals are periodic. Such an arrival sequence is called *Deterministic* release. We present the figures for both the *mean waiting time* and the standard deviation of the cycle–time. (The mean waiting time is the difference between the mean cycle–time

and the mean total processing time).

Figure 8: Performance of FSVCT against other scheduling policies, under Deterministic Release. What we show is the percentage improvement of each scheduling policy with respect to the well known FIFO scheduling policy, which has been chosen as a baseline. The policies shown in order from left to right are: FSVCT, CYCLIC, LTNV, FIFO+, FIFO, LWNQ, FGCA, SRPT++, SRPT+, SRPT, and EDD.

Figure 9: Performance of FSVCT against other scheduling policies, under Poisson Release. Shown are the percentage improvements in the Mean Waiting Time and the Standard Deviation of the Cycle–Time, over the baseline FIFO Policy. The policies shown in order from left to right are: FSVCT, CYCLIC, LTNV, FIFO+, FIFO, LWNQ, FGCA, SRPT++, SRPT+, SRPT, and EDD.

In Figure 9, we present the results for the case of Poisson arrivals to the plant, i.e., when the interarrival times are random and have an exponential probability distribution,

$$\text{Prob (Interarrival Time } > t) = e^{-\lambda t}, \text{ where } \lambda = \text{ arrival rate.}$$

We see that in each case the FSVCT scheduling policy does reduce the standard deviation of cycle-time by about a half – a substantial (and also statistically significant; see [13]) improvement.

We also see that FSVCT provides a sizable (about 20%) and significant reduction in *mean* cycle-time. However, the benefits are less sizable under Poisson releases.

Why is this so? To develop an explanatory hypothesis, we turn to the First Law of Queueing Systems.

6 The First Law of Queueing Systems: Delay is Caused by Fluctuations

Consider the system shown in Figure 10. Parts arrive to a machine, evenly spaced

Figure 10: Burstiness causes delay – waiting time is small when the arrivals are periodic.

(i.e., periodically, or in the earlier terminology, under Deterministic Release), exactly 60 minutes apart. The machine spends exactly 59 minutes on each part. Clearly, in steady state, each part will arrive exactly one minute after the previous part has left. Thus no part ever waits for the machine to become available. The waiting time, or queueing time, is 0.

Now suppose that instead of arriving periodically, parts arrive randomly with a mean interarrival time of 60 minutes. Some parts may therefore arrive close together in time, while others may arrive far apart in time; see Figure 11.

Figure 11: Burstiness causes delay – mean waiting time is large when there are fluctuations in the interarrival times.

Suppose, for specificity, that the interarrival time distribution is exponential with mean 60, i.e., the arrivals form a Poisson process of rate 1/60. Then, even when the machine continues to take exactly 59 minutes per part, the average waiting time is almost 29 *hours*! (This calculation is performed using the Pollaczek–Khintchine formula for an M/G/1 queue; see [1]).

Clearly, the waiting time or delay is increased by fluctuations in the interarrival times of parts.

7 Why the Mean Cycle-Time is Small Under Deterministic Release

Now we are ready to hypothesize an explanation for why the mean cycle-time is small under *Deterministic* Release.

Under Deterministic Release, the interarrival times to the plant are periodic, and so have *no* fluctuations. Also, when the FSVCT scheduling policy is employed, the variance of the cycle–time is small, and there is thus little fluctuation in the times spent in the plant by the parts. Now note that, exit time = arrival time + cycle-time. Hence the exit stream from the plant also has little fluctuations.

We will now show that, in fact, the arrival stream to *every* buffer has little fluctuations, when FSVCT is employed for scheduling.

Let b_j be a given buffer. If b_i precedes b_j, then we can write ζ_i as

$$\zeta_i = \zeta_{ij} + \zeta_j,$$

where ζ_{ij} is the mean time that parts take to go from b_i to b_j.

Consider now two buffers b_i and b_ℓ, both of which are in competition for the same machine. Suppose both buffers precede b_j (i.e., $i, \ell < j$). Consider a part π in b_i, and a part π' in b_ℓ. When comparing the slacks of parts π and π', we can ignore ζ_j, since it is common to b_i and b_ℓ, and write the slacks as,

$$s(\pi) \;=\; \alpha(\pi) - \zeta_{ij}, \text{ and}$$

$$s(\pi') \;=\; \alpha(\pi') - \zeta_{\ell j}.$$

Now we note that this is exactly the way in which the slacks $s_2(\pi)$ and $s_2(\pi')$ would have been defined, if the system had been truncated at b_j, and b_{j-1} was actually the last buffer in the system. Earlier, we saw that the FSVCT scheduling policy succeeds in reducing the fluctuations in the *exit stream*. Hence, for the truncated system, FSVCT reduces the fluctuations in the exit stream from b_{j-1}, and hence the arrival stream to b_j.

Since b_j was arbitrary, we see that the fluctuations in *all internal flows are simultaneously reduced*. By the First Law of Queueing Systems, we expect the mean cycle–time to be small. This is exactly what we observed under Deterministic Release!

This is our hypothesis for why the mean cycle–time was small in Figure 8 when the FSVCT scheduling policy was used under Deterministic Release.

8 How to Reduce the Mean Cycle-Time Under Other Releases

How can we simultaneously reduce the fluctuations in all internal flows in the network, *simultaneously*, when the arrivals to the plant are *not* Deterministic?

The solution we propose is simple. Even if the arrivals are not periodic, we will set *periodic due-dates*, and then reduce the variance of lateness with respect to the periodic due dates. Clearly, the exit stream from the plant will have small fluctuations. Moreover, as we saw in the previous section, every internal flow has small fluctuations, too.

Thus, if π is the n-th part arriving into the system, we set its due-date as,

$$\delta(\pi) := \frac{n}{\lambda}, \text{ where } \frac{1}{\lambda} = \text{ mean interarrival time.}$$

Then we use the Least Slack policy, where we redefine the slack as,

$$s_3(\pi) := \frac{n}{\lambda} - \zeta_i. \tag{4}$$

We call this the *Fluctuation Smoothing Policy for Mean Cycle-Time* (FSMCT).

9 The Magnitude of the Improvements Obtained

How well does FSMCT perform? For Poisson releases, as shown in Figure 12, the newly designed FSMCT scheduling policy achieves a further reduction in the mean waiting time of about 18%, over the FSVCT policy.

Figure 12: Improvement obtained by FSMCT in mean waiting time, for Poisson Releases. For the mean waiting time figure on the left, the scheduling policies shown are, from left to right,EDD, SRPT, SRPT+, SRPT++, FGCA, LWNQ, FIFO, FIFO+, LTNV, CYCLIC, FSVCT, and FSMCT. For the standard deviation of cycle time figure on the right, the scheduling policies shown are, from left to right, FGCA, FIFO+, FIFO, LWNQ, LTNV, CYCLIC, SRPT, SRPT+, SRPT++, EDD, FSVCT, and FSMCT.

We have not addressed so far the issue of choosing a good *release* policy. Based on the idea of replacing a queueing network in its heavy traffic limit (as the arrival rate increases to capacity), by a reflected Brownian motion, the use of a release policy called the Workload Regulation (WR) Release policy has been advocated in [14]. This policy releases parts into the system whenever the work in the system for the bottleneck machines drops below a certain threshold value. The threshold is carefully chosen to maintain a certain long–term average rate of releases.

This *Workload Regulation* Release policy (WR), advocated and tested in [14], was also found to be the best among the release policies tested in [13].

A comparison of WR Release vs. Deterministic Release, and FSMCT scheduling vs. FIFO scheduling, is shown in Figures 13 and 14.

As can be seen, the increase in efficiency obtained by proper scheduling (using FSMCT scheduling with WR releases), can be substantial.

Figure 13: Mean Waiting Times for WR Release and FSMCT Scheduling vs. Deterministic Release and FIFO Scheduling.

Figure 14: Standard Deviation of Cycle–Times for WR Release and FSMCT Scheduling vs. Deterministic Release and FIFO Scheduling.

10 Another Interpretation of FSMCT: Alleviating Total Downstream Shortfall

We now provide another appealing equivalent interpretation of the FSMCT policy, as attempting to alleviate the total downstream shortfall of parts from each buffer.

Note first that instead of using the comparisons of the slacks $s_3(\pi)$ to select parts, one could equivalently use $f(s_3(\pi))$, as long as f is a strictly monotone increasing function. If

$N(t)$ denotes the number of parts that have departed from the system in the time interval $[0, t]$, consider the following function f (which is strictly monotone increasing in the variable $s_3(\pi)$), used to select the part at time t,

$$f(s_3(\pi)) := \lambda s_3(\pi) - N(t).$$

This expression can be simplified as follows. Substituting for $s_3(\pi)$ from (4), we obtain

$$f(s_3(\pi)) = n - N(t) - \lambda\zeta_i.$$

Since n is the number of the part in buffer b_i, the quantity $n - N(t)$ denotes the current number of parts in the buffers *downstream* from b_i at time t, i.e.,

$$n - N(t) = \sum_{j \geq i} x_j(t),$$

where

$$x_j(t) := \text{Number of parts in buffer } b_j \text{ at time } t.$$

Also denote,

$$\bar{x}_j := \textit{Mean} \text{ number of parts in buffer } b_j.$$

Then we note from Little's Law that,

$$\begin{aligned} \lambda\zeta_i &= \text{Mean number of parts downstream from } b_i \\ &= \sum_{j \geq i} \bar{x}_j. \end{aligned}$$

Hence $f(s_3(\pi))$ can be rewritten as,

$$f(s_3(\pi)) = \sum_{j \geq i} x_j(t) - \sum_{j \geq i} \bar{x}_j. \tag{5}$$

Thus, $f(s_3(\pi))$ denotes the *total downstream shortfall* from buffer b_i, where the shortfall is the difference between the actual number of parts downstream and the mean number of parts downstream.

Hence FSMCT can be regarded as a policy which gives priority to that buffer whose total downstream shortfall is currently greatest – an appealing interpretation. We refer the reader

to Li [15] for another approach which attempts to drive each buffer's contents towards its mean value.

Equation (5) is reminiscent of integral compensation in classical control, with the space variable j replacing the usual time variable. Let $e_j(t) := x_j(t) - \bar{x}_j$ be the "error" at buffer b_j, i.e., the deviation from the desired value. Then $\sum_{j \geq i} e_j(t)$ is the integral, in space not time, of the error. The FSMCT policy uses this integral of the error to choose the buffer to serve. The connections are worth pursuing.

11 Concluding Remarks

We have seen that substantial improvements in plant efficiency, about a 20% improvement in mean waiting time, and about a 50% improvement in standard deviation of cycle–time, have been obtained in testing on plant models, by the use of the new scheduling policies. It would be valuable to extend the scheduling policies to incorporate the features of batching, set–ups, multiple process flows, and operator availability.

We have provided only experimental results here. If useful theoretical results were available, they could tell us what the performance of a given policy is, and what the best policy is. The first issue, that of determining the performance of a given policy, reduces broadly to determining the steady–state distribution of a Markov chain. Unfortunately, the queueing networks studied here fall outside the class of "product–form" queueing networks, for which explicit solutions are available. Recent results allow the performance and stability evaluation of even some non–classical queueing networks; see [7, 8, 9, 10, 11], and the survey [12]. The second issue of determining the best policy consists essentially of determining an optimal policy for a stochastic system. Unfortunately, the method for solving such problems, dynamic programming (see [2]) is computationally intractable even for very small systems. One recent method of some interest consists of approximating the detailed stochastic system by a reflected vector Brownian motion, see [16]. It is sometimes easier to develop scheduling policies for the resulting Brownian approximation; see [17].

However, as we have seen in this paper, even in the absence of a completely satisfactory

theory, it is still possible to exploit the central tenets of what we know from theory. Here we have described two "laws" of queueing theory, and shown how one can exploit such general principles to design scheduling policies which appear attractive. In this approach, one takes recourse to experimentation (in our case, on virtual factory models stored in computers), to test hypotheses about system behavior, refine them, and exploit them. We believe that this may well be the way that many large systems can be better controlled.

Ending on a broad note, we note that there is much economic incentive to develop scheduling methods which are designed for large systems that are prone to random disruptions, and which can be implemented dynamically in real-time. Clearly, there is a great need for both a deeper descriptive theory (performance evaluation), and a prescriptive theory – control.

References

[1] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. New York, NY: Wiley–Interscience, 1975.

[2] P. R. Kumar and P. P. Varaiya, *Stochastic Systems: Estimation, Identification and Adaptive Control*. Englewood Cliffs, NJ: Prentice-Hall, 1986.

[3] J. R. Jackson, "Networks of waiting lines," *Mathematics of Operations Research*, vol. 5, pp. 518–521, 1957.

[4] J. R. Jackson, "Jobshop-like queueing systems," *Management Science*, vol. 10, pp. 131–142, 1963.

[5] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios, "Open, closed and mixed networks of queues with different classes of customers," *J. Assoc. Comput. Mach.*, vol. 22, pp. 248–260, April 1975.

[6] F. P. Kelly, *Reversibility and Stochastic Networks*. New York, NY: John Wiley and Sons, 1979.

[7] D. Bertsimas and I. Ch. Paschalidis and J. N. Tsitsiklis, "Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance." Laboratory for Information and Decision Systems and Operations Research Center, M. I. T., December 1992.

[8] P. R. Kumar, "Re–entrant lines," *Queueing Systems: Theory and Applications: Special Issue on Queueing Networks*, vol. 13, pp. 87–110, May 1993.

[9] S. Kumar and P. R. Kumar, "Performance bounds for queueing networks and scheduling policies," technical report, Coordinated Science Laboratory, University of Illinois, Urbana, IL, 1992. To appear in *IEEE Transactions on Automatic Control*, August 1994.

[10] P. R. Kumar and S. P. Meyn, "Stability of queueing networks and scheduling policies," tech. rep., C. S. L., University of Illinois, 1993.

[11] P. R. Kumar and S. Meyn, "Duality and linear programs for stability and performance analysis of queueing networks and scheduling policies," tech. rep., C. S. L., University of Illinois, 1993.

[12] P. R. Kumar, "Scheduling queueing networks: Stability, performance analysis and design," technical report, Coordinated Science Laboratory, University of Illinois, Urbana, IL, 1994. To appear in *Stochastic Networks*, IMA Volumes in Mathematics and its Applications, Frank Kelly and Ruth Williams, Editors, Springer-Verlag.

[13] Steve C. H. Lu, D. Ramaswamy, and P. R. Kumar, "Efficient scheduling policies to reduce mean and variance of cycle–time in semiconductor manufacturing plants," technical report, University of Illinois, Urbana, IL, 1992. To appear in *IEEE Transactions on Semiconductor Manufacturing, 1994*.

[14] L. M. Wein, "Scheduling semiconductor wafer fabrication," *IEEE Transactions on Semiconductor Manufacturing*, vol. 1, pp. 115–130, August 1988.

[15] S. Li, "An introduction to basic principles of production," in *Semiconductor Manufacturing Technology Workshop*, (Hsinchu, Taiwan, R.O.C.), pp. 3–20, March 1993.

[16] J. M. Harrison, "Brownian models of heterogeneous customer populations," in *Stochastic Differential Systems, Stochastic Control Theory and Applications, IMA Volumes in Mathematics and its Applications* (W. Fleming and P. L. Lions, eds.), ch. 10, pp. 147–186, Springer-Verlag, 1988.

[17] J. M. Harrison and L. M. Wein, "Scheduling networks of queues: Heavy traffic analysis of a two-station closed network," *Operations Research*, vol. 38, no. 6, pp. 1052–1064, 1990.