[32] Steve C. H. Lu, Deepa Ramaswamy, and P. R. Kumar. Efficient scheduling policies to reduce mean and variance of cycle–time in semiconductor manufacturing plants. *IEEE Transactions on Semiconductor Manufacturing*, 7(3):374–385, 1994.

[33] K. G. Murty. *Linear Programming.* John Wiley and Sons, New York, NY, 1983.

[34] K. G. Murty and S. N. Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39:117–129, 1987.

[35] J. Ou and L. M. Wein. Performance bounds for scheduling queueing networks. *Annals of Applied Probability*, 2:460–480, 1992.

[36] S. S. Panwalker and W. Iskander. A survey of scheduling rules. *Operations Research*, 25(1):45–61, January-February 1977.

[37] L. M. Wein. Scheduling semiconductor wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 1(3):115–130, August 1988.

[20] J. F. C. Kingman. Inequalities in the theory of queues. *Journal of the Royal Statistical Society, Series B*, 32:102–110, 1970.

[21] P. R. Kumar. Re–entrant lines. *Queueing Systems: Theory and Applications: Special Issue on Queueing Networks*, 13(1–3):87–110, May 1993.

[22] P. R. Kumar and S. P. Meyn. Stability of queueing networks and scheduling policies. To appear in *IEEE Tranactions on Automatic Control*, February 1995, 1993.

[23] P. R. Kumar and Sean Meyn. Duality and linear programs for stability and performance analysis of queueing networks and scheduling policies. Technical report, C. S. L., University of Illinois, 1993.

[24] P. R. Kumar and T. I. Seidman. Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. *IEEE Transactions on Automatic Control*, AC-35(3):289–298, March 1990.

[25] S. Kumar and P. R. Kumar. Fluctuation smoothing policies are stable for stochastic re-entrant lines. To appear in *Proceedings of the 33rd IEEE Conference on Decision and Control*, December 1994.

[26] S. Kumar and P. R. Kumar. The last buffer first policy is stable for stochastic re–entrant lines. Technical report, Coordinated Science Laboratory, University of Illinois, Urbana, IL, 1994.

[27] S. Kumar and P. R. Kumar. Performance bounds for queueing networks and scheduling policies. *IEEE Transactions on Automatic Control*, AC-39:1600–1611, August 1994.

[28] S. Lippman. Applying a new device in the optimization of exponential queueing systems. *Operations Research*, 23:687–710, 1975.

[29] S. C.-H. Lu. *Control policies for scheduling of semiconductor manufacturing plants*. PhD thesis, University of Illinois, Urbana-Champaign, IL, 1994.

[30] S. H. Lu and P. R. Kumar. Distributed scheduling based on due dates and buffer priorities. *IEEE Transactions on Automatic Control*, AC-36(12):1406–1416, December 1991.

[31] Steve C. H. Lu, Deepa 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*.

[6] R. L. Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.

[7] J. Dai and G. Weiss. Stability and instability of fluid models for certain re-entrant lines. Preprint, February 1994.

[8] J. G. Dai. On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models. Technical report, Georgia Institute of Technology, 1993. To appear in *Annals of Applied Probability*.

[9] J. G. Dai and Vien Nguyen. On the convergence of multiclass queueing networks in heavy traffic. *The Annals of Applied Probability*, 4(1):26–42, 1994.

[10] J. G. Dai and Y. Wang. Nonexistence of Brownian models for certain multiclass queueing networks. *Queueing Systems: Theory and Applications: Special Issue on Queueing Networks*, 13(1–3):41–46, May 1993.

[11] J. L. Doob. *Stochastic Processes.* John Wiley and Sons, New York, NY, 1953.

[12] D. G. Down and S. P. Meyn. Piecewise linear test functions for stability of queueing networks. In *Proceedings of the IEEE 33th Conference on Decision and Control*, Buena Vista, FL, December 1994. to appear.

[13] J. M. Harrison and V. Nguyen. Some badly behaved closed queueing networks. Technical report, 1994.

[14] J. M. Harrison and L. M. Wein. Scheduling networks of queues: Heavy traffic analysis of a two-station closed network. *Operations Research*, 38(6):1052–1064, 1990.

[15] J. R. Jackson. Networks of waiting lines. *Mathematics of Operations Research*, 5:518–521, 1957.

[16] J. R. Jackson. Jobshop-like queueing systems. *Management Science*, 10:131–142, 1963.

[17] H. Jin, J. Ou, and P. R. Kumar. The throughput of closed queueing networks–uniform bounds, efficient scheduling, heavy traffic behavior, and buffer priority policies. Technical report, University of Illinois, Coordinated Science Laboratory, 1994.

[18] S. Karlin and H. M. Taylor. *A First Course in Stochastic Processes.* Academic Press, New York, NY, 1975.

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

where the downstream shortfall from buffer b_i is the difference between the number of parts downstream from b_i and the mean number downstream. In addition to showing that FSMCT is a stationary policy, this also provides a very appealing interpretation of the policy – as attempting to alleviate downstream shortfalls.

15 Concluding Remarks

We simply end on a broad note. There is a clear need for a theory of scheduling which is (i) tractable for very large systems, (ii) able to handle random events, e.g., machine failures, and (iii) implementable dynamically in real time. There is strong need for a deeper theory on two fronts, (i) a descriptive theory of performance analysis, and (ii) a prescriptive theory of control or scheduling. There are of course many important open problems.

What we believe is that there is a need for researchers to keep an ultimate application in focus, where the phrase "in focus" is interpreted in a broad but nevertheless purposeful sense. We believe that theory and applications are not antagonistic. A knowledge of the application allows for creativity on modeling issues, as well as problem formulations, and allows an easier development of a richer and deeper theory.

Finally, we believe that industry should make a serious effort in publicizing and exposing researchers and potential researchers to problems, models, data, etc.

References

[1] L.-E. Andersson, G. Chang, and T. Elfving. Criteria for copositive matrices and nonnegative Bezier patches. Technical Report LiTH-MAT-R-93-27, Linkoping University and University of Science and Technology, China, August 1993.

[2] 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.*, 22(2):248–260, April 1975.

[3] D. Bertsimas, I. Ch. Paschalidis, and J. N. Tsitsiklis. Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance. *Annals of Applied Probability*, 4:43–75, 1994.

[4] Maury Bramson. Instability of FIFO queueing networks with quick service times. Technical report, Mathematics Department, University of Wisconsin, Madison, WI, 1993.

[5] R. W. Cottle, G. J. Habetler, and C. E. Lemke. On classes of copositive matrices. *Linear Algebra and Its Applications*, 3:295–310, 1970.

Fab 3: Standard Deviation of Cycle-Time	**FIFO**	**FSMCT**	**FSVCT**
Deterministic Release	240.93	139.64	Same policy as FSMCT
Workload Regulation Release	173.31	103.72	83.21

Figure 29: Improvements in variance of cycle time obtained by using WR Release and FSMCT, over Deterministic Release and FIFO.

The improvements are substantial, and, as statistically tested in [32], significant.

14.1 FSMCT is a Stationary Policy

We now show that FSMCT is a *stationary* scheduling policy, in the sense that it depends only on the vector $x(t)$ of queue lengths. Thus, we can potentially use Markovian methods to study its performance.

Clearly, instead of $s_3(\pi)$, one can use any monotone increasing function $f(s_3(\pi))$ to make decisions. Consider the choice

$$f(s_3(\pi)) := \lambda s_3(\pi) - (\text{number of departures from system up to time } t).$$

The first term on the right hand side above is simply $\lambda s_3(\pi) = n - \lambda\zeta_i$. By Little's Theorem, $\lambda\zeta_i =$ mean number of parts downstream from buffer b_i. Thus,

$$\begin{aligned} f(s_3(\pi)) = {} & (n - \text{number of departures from system up to time } t) \\ & - \text{mean number of parts downstream from buffer } b_i. \end{aligned}$$

Note now that parts are always served from the head of a buffer, and so parts never overtake each other. Hence parts depart from the system in the order that they arrive. Since part π is the n-th release into the system, and it is located in b_i,

$$\begin{aligned} & (n - \text{number of departures from system up to time } t) \\ & \qquad = \text{number of parts downstream from } b_i. \end{aligned}$$

Thus,

$$f(s_3(\pi)) = \text{downstream shortfall from } b_i,$$

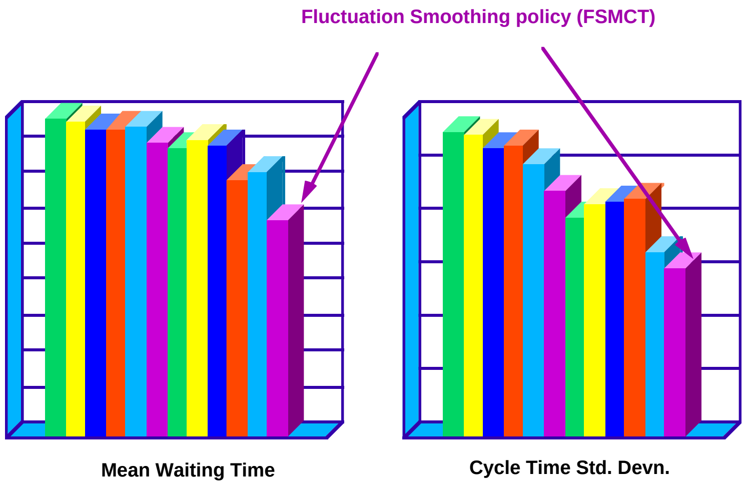
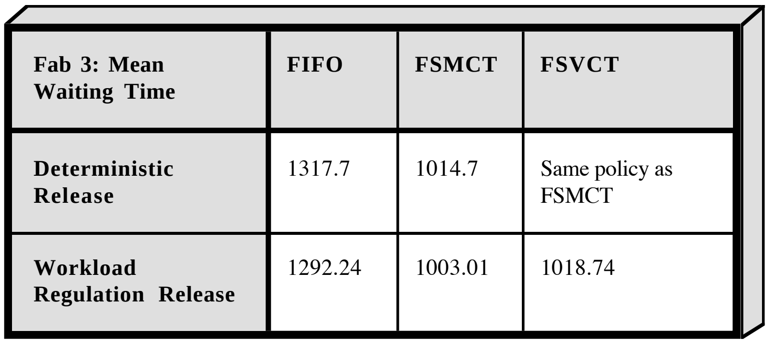


Figure 27: Improvement obtained in mean waiting time by using FSMCT. The FSMCT policy is rightmost. The FSVCT policy is second from the right.

FSMCT yields about a further 18% improvement over FSVCT.

Regarding release policies, the *Workload Regulation Release* (WR) Policy is advocated in [37]. The simulations in [32] affirm the choice of this release policy. While its mean cycle–time under FSMCT is almost the same as deterministic release, its variance is smaller.

Hence we propose the combination of using WR for releases, and FSMCT for scheduling. Figures 28 and 29 show the performance improvements obtained by using either or both of these policies in comparison to Deterministic release and FIFO scheduling.



Fab 3: Mean Waiting Time	**FIFO**	**FSMCT**	**FSVCT**
Deterministic Release	1317.7	1014.7	Same policy as FSMCT
Workload Regulation Release	1292.24	1003.01	1018.74

Figure 28: Improvements in mean waiting time obtained by using WR Release and FSMCT, over Deterministic Release and FIFO.

since the constant term ζ_j in (36) can be ignored in comparisons. This, however, simply corresponds to viewing b_{j-1} as the exit from the system, and trying to reduce the variance of *interarrival times* to b_j.

Thus, under deterministic releases, FSVCT tries to reduce the variance of interarrival times to all buffers b_j – *simultaneously.* Keeping in mind the first law of queueing systems, we see that it reduces the mean cycle-time under deterministic releases.

Now the question arises, how can we exploit this to design a scheduling policy that reduces the mean cycle-time not just under deterministic releases, but also under other releases? The answer is simple! We simply *set* deterministic due-dates, i.e.,

$$\delta(\pi) := \frac{n}{\lambda} \quad \text{if} \quad \begin{array}{l} \pi \text{ is the } n\text{-th part released} \\ \text{into the system, and} \\ \dfrac{1}{\lambda} = \text{ mean interarrival time,} \end{array}$$

and continue to reduce the variance of lateness. Since the due-dates are periodic, when we reduce the variance of lateness, we will end up reducing the variance of interdeparture times. Due to the definition of the ζ_i's, we will thus be *simultaneously reducing the fluctuations in arrivals to all buffers*, and then we should be able to reduce the mean cycle-time.

To summarize, we redefine the slack as

$$s_3(\pi) := \frac{n}{\lambda} - \zeta_i \qquad \begin{array}{l} \text{where } \pi \text{ is the } n\text{-th part released} \\ \text{into the system, and it is} \\ \text{currently located in } b_i. \end{array}$$

The corresponding Least Slack policy is called the *Fluctuation Smoothing Policy for Mean Cycle-Time* (FSMCT).

How well does FSMCT perform? In Figure 27 we show the performance for Poisson releases. Shown are Waiting Time (Policy) and Standard Deviation (Policy), for various policies.

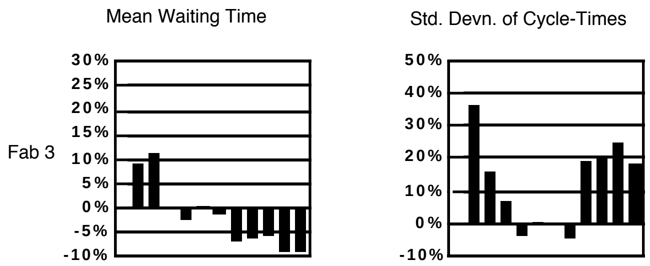


Figure 26: Performance under Poisson releases. Shown are the percentage improvements in the Mean Waiting Time and the Standard Deviation of Cycle-Time, over the baseline FIFO Policy.The FSVCT Policy is shown leftmost.

As observed from the figures, FSVCT provides an improvement in standard deviation of cycle time, over FCFS, of about 40%, for both deterministic and Poisson releases.

Interestingly, while the FSVCT policy was not designed to reduce the mean cycle-time, we note in Figure 25 that it does provide a substantial improvement, about 20%, over FCFS for the *mean waiting time*, when the releases are deterministic. However, there is less of an improvement when the arrivals are Poisson.

To understand why this is so (and to later exploit the understanding and devise a new policy for reducing the mean cycle-time), we recall the *First Law of Queueing Systems – delay is caused by fluctuations.* This is manifested in, for example, the Pollaczek-Khintchine Formula, where one sees that fluctuations in the service time, i.e., the variance of the service time, increase the mean delay in an $M/G/1$ queue. Similarly, Kingman's bounds also show this to be so for the variance in interarrival times.

With this fact – fluctuations cause delays – we can understand why FSVCT reduces the delay under deterministic releases. Clearly the arrivals have no burstiness – they are periodic. Hence, when we reduce the variance of the cycle-times, we have therefore also reduced the *variance of interdeparture times* from the system.

Note now that our definition of ζ_i has the property,

$$\zeta_i = \zeta_{ij} + \zeta_j, \tag{36}$$

where

$$\zeta_{ij} := \text{ mean time to go from } b_i \text{ to } b_j.$$

Thus, when comparing parts in two buffers b_i and b_k at a machine, both of which precede b_j, we see that we could have equivalently defined the slack as

$$s_2(\pi) = \alpha(\pi) - \zeta_{ij},$$

then,

$$e(\pi) - \delta(\pi) = \text{ lateness } = \text{ cycle-time } = e(\pi) - \alpha(\pi).$$

Thus, the earlier policy of reducing the variance of lateness can be used to reduce the variance of cycle-time. Thus we redefine the slack as,

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

and call the corresponding least slack policy the *Fluctuation Smoothing Policy for Variance of Cycle-time* (FSVCT).

Let us see how our FSVCT policy performs. In [31], extensive simulation tests have been conducted on models of a Hewlett Packard Research and Development Fabrication Line Model detailed in [37], and a full scale production line. The scheduling policies compared include all the policies tested in the earlier comprehensive study in [37], as well as others. We present some results for the cases of Poisson releases (arrivals) to the plants, and *deterministic* (i.e., periodic) releases. Other release polcies are also examined in [32]. Define the mean waiting time as the mean cycle time minus the sum of the mean processing times. Let Waiting Time (Policy) denote the average over the simulation runs of the mean waiting time for a particular scheduling policy. The FCFS (aka FIFO) policy is well known. Define the percentage improvement in waiting time of a policy with respect to FCFS as [Waiting Time (FCFS) - Waiting Time (Policy)]/Waiting Time (FCFS). Similarly, the quantity, percentage improvement in standard deviation of cycle time of a policy with respect to FCFS is defined. Below, in Figures 25 and 26, we present the percentage improvements of various policies on a Hewlett Packard Research and Development Fabrication Line Model of [37].

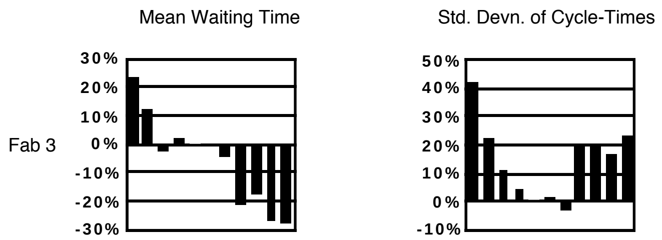


Figure 25: Performance under deterministic releases. Shown are the percentage improvements in the Mean Waiting Time and the Standard Deviation of Cycle-Time, over the baseline FIFO Policy. The FSVCT Policy is shown leftmost.

We will address the issue of designing good policies for three objectives: (i) reducing the variance of lateness, (ii) reducing the standard deviation of cycle-times, and (iii) reducing the mean cycle-time. In this and later sections, our development will not be mathematical. It will be more akin to physics, with emphasis on developing hypothesis, testing, refining hypotheses etc., all for large system models.

We will generically denote a part by π, and adopt the following notation:

$$\begin{aligned} \alpha(\pi) &= \text{release time or arrival time of a part } \pi, \\ e(\pi) &= \text{exit time from system of a part } \pi, \\ \delta(\pi) &= \text{due–date of a part } \pi. \end{aligned}$$

We first address the problem of reducing the *variance of lateness*, $var(e(\pi)-\delta(\pi))$. Let,

$$\zeta_i \quad := \quad \text{estimate of mean remaining time to exit for a part in buffer } b_i.$$

The quantity $\delta(\pi)-t$, where t is the current time, represents the time until due date for a part π. If the part π is located in buffer b_i, then the estimate of the remaining time in the system is ζ_i. Thus, the "slack"

$$s_1(\pi) := \delta(\pi) - t - \zeta_i$$

is a measure of urgency of the part.

Let us therefore consider the *Least Slack* (LS) Policy which gives priority to the part with the smallest slack. Such a policy seems intuitively fair. Quantitatively, what such fairness amounts to is an effort to make all parts equally late (or early). If all parts are equally late or equally early, then the variance of lateness is small. Thus, one has reason to believe that the above policy reduces the variance of lateness.

We note that since the current time t is common to all parts, it can be dropped from the comparison, and one can thus redefine the slack more simply as,

$$s_1(\pi) := \delta(\pi) - \zeta_i. \tag{35}$$

We call this Least Slack policy with slack defined by (35), the *Fluctuation Smoothing Policy for the Variance of Lateness* (FSVL).

Now let us turn to the issue of reducing the *variance of the cycle-time*, $var(e(\pi)-\alpha(\pi))$. Note that if we were to *set* the due-date $\delta(\pi)$ as equal to the arrival time $\alpha(\pi)$, i.e.,

$$\delta(\pi) := \alpha(\pi),$$

and so the Performance LP gives an upper bound. Note now that the effective arrival rate is less than λ. Let us define,

$$w_0(\tau_n) = \begin{cases} 1 \text{ if arrivals are "on"} \\ 0 \text{ if arrivals are "off."} \end{cases}$$

Now, if we run through the arguments of Section 9, we simply find that all the earlier terms of the form $E(\lambda x_j(\tau_n))$ are replaced by $E[\lambda w_o(\tau_n)x_j(\tau_n)]$, which is less than $E(\lambda x_j(\tau_n))$. Thus some of the earlier equality constraints become inequality constraints. Now one can take the limit as $M \nearrow \infty$, obtaining an LP which bounds the original system without any flow control.

Theorem 12: The Monotone LP. *Consider the LP (25-33), replacing however the earlier constraints (26,27,28,30) by the following constraints:*

$$\lambda \sum_{\{i:\sigma(i)=\sigma(1)\}} z_{i1} - \mu_1 z_{11} \geq -\lambda,$$

$$\mu_{j-1} z_{j-1,j} - \mu_j z_{jj} \geq -\lambda \text{ for } 2 \leq j \leq L,$$

$$\lambda \sum_{\{i:\sigma(i)=\sigma(2)\}} z_{i2} - \mu_1 z_{12} + \mu_1 z_{11} - \mu_2 z_{21} \geq 0,$$

$$\lambda \sum_{\{i:\sigma(i)=\sigma(j)\}} z_{ij} - \mu_1 z_{1j} - \mu_j z_{j1} + \mu_{j-1} z_{j-1,1} \geq 0 \text{ for } 3 \leq j \leq L,$$

$$\mu_{i-1} z_{i-1,i+1} - \mu_i z_{i,i+1} + \mu_i z_{ii} - \mu_{i+1} z_{i+1,i} \geq 0 \text{ for } 2 \leq i \leq L-1.$$

Its value for any λ bounds the mean number of parts in the system for all arrival rates $\lambda' \leq \lambda$. □

From the way λ enters the constraints above, it is clear that the feasible set is increasing in λ, and hence so is the value of the LP.

One may note that in the dual this corresponds to setting only certain q_{ij}'s nonnegative.

As earlier, this can be extended to systems incorporating other constraints.

14 Design: Scheduling of Re-Entrant Lines

Let us return to the problem of scheduling re-entrant lines, introduced in Section 2. The following results are drawn from [32].

13.1 Boundedness of Performance LP Implies That All Non–Idling Policies Have a Geometrically Converging First Moment

We will now show that when the Performance LP (25-33) is bounded, every stationary non-idling policy has a geometrically converging exponential moment.

To see this, we use duality and our instability results, as follows.

Since the Performance LP is bounded, its dual, the Drift LP. is feasible. Hence there exists a symmetric matrix Q for which $x^T(\tau_n)Qx(\tau_n)$ has the negative drift (6). We now show that this matrix Q is automatically copositive. Suppose not, then either a non-idling policy, or a small perturbation of it, which is still non-idling, does not have a finite first moment. However that is impossible, since we have already seen that all non-idling policies have performance bounded by M. Thus, Q is copositive. Then however we know that all stationary non-idling policies have a geometrically converging exponential moment.

Theorem 11. *Suppose the Performance LP (25-33) for the class of all non-idling policies is bounded.*

(i) *Then there exists a $Q = Q^T$ satisfying the negative drift condition. Moreover, every such Q is copositive.*

(ii) *All stationary non-idling polices have a geometrically converging exponential moment.*

It is an important open issue whether Theorem 11 can be extended to Performance LP's incorporating additional constraints to model specific policies, such as buffer priority policies, and others considered in Section 10.

13.2 Uniform Bounds, Stability and the Monotone LP

We note that stability is not a monotone property. For example, an originally stable system can become unstable if some service rates are increased, see [4].

Consider the Drift LP with nonegativity constraints $q_{ij} \geq 0$ appended to it. From an examination of (12), it is easy to see that if negative drift holds for some λ, it also holds for all $\lambda' \leq \lambda$. Hence one has established stability not just for a particular value of the arrival rate λ, but also for *all smaller arrival rates.* However, this requires the existence of a nonnegative matrix Q satisfying the Drift LP.

Now we show how the requirement of nonegativity for all coefficients q_{ij} can be weakened. Thereby, not only do we establish stability for a range of λ's, but we also obtain bounds for all smaller arrival rates. This is done by obtaining an LP with certain monotonicity properties.

Consider the following "flow control" device. Whenever the number of parts in the system exceeds M, we discard the arrivals. Clearly the system is stable,

$$-\mu_j r_{ij} + \mu_i r_{i+1,j} - p_j 1_{\{\sigma(i)=\sigma(j)\}} + w_{\sigma(i),j} 1_{\{\sigma(i)\neq\sigma(j)\}} \geq 0 \text{ for } j \geq i+1. \qquad (z_{ij})$$

Above, by r_{ij} for $i > j$, we mean r_{ji}. The dual variables associated with the constraints are shown in parentheses, alongside the constraints.

(ii) *Defining*

$$q_{ij} := -r_{ij},$$

and eliminating the p_i's and $w_{\sigma,i}$'s, we see that the dual of the Performance LP above is the Drift LP*:*

$$\text{Min} \sum_{i=1}^{L} \lambda q_{ii} - \sum_{i=1}^{L-1} \lambda q_{i,i+1}$$

subject to

$$\lambda q_{1j} + \max_{\{i:\sigma(i)=\sigma(j)\}} \mu_i (q_{i+1,j} - q_{ij})$$
$$+ \sum_{\{\sigma:\sigma\neq\sigma(j)\}} \max_{\{i:\sigma(i)=\sigma\}} \mu_i (q_{i+1,j} - q_{ij})^+ \leq -1, \text{ for } 1 \leq j \leq L,$$
$$q_{ij} = q_{ji} \text{ and } q_{L+1,j} = 0 \text{ for } 1 \leq i,j \leq L.$$

Thus we see that whenever the Performance LP (for the class of all non–idling policies) has a finite upper bound, the Drift LP (for the class of all non–idling policies) has a feasible solution; in particular, there exists a $Q = Q^T$ whose associated quadratic form has the negative drift (6). Note however, that we have not yet established that the matrix Q is copositive; that is done in the next section.

Let us now consider briefly the LP for transient performance, given in Theorem 9. There, the earlier equality constraints in (25-33) are replaced by inequality constraints. Thus, the dual of the transient LP features nonnegativity constraints $q_{ij} \geq 0$ for all i, j. Hence, when the LP for transient performance is bounded, there exists a *nonnegative* matrix $Q = Q^T$ whose associated quadratic form has negative drift. Thus we discover a very interesting connection between nonnegative Q's in the drift condition and transient performance.

It is also true that a similar duality as in Theorem 10, also holds for the class of buffer priority policies; the associated Performance LP of Section 10.1 and the Drift LP of Section 7 are similarly duals; see [23] for details. However, in this latter case, we are unable to conclude that when the Performance LP is bounded, the resulting feasible Q in the dual Drift LP is automatically copositive.

13 Consequences of Duality

We can take advantage of duality to considerably sharpen our results.

12 The Duality of Stability and Performance

We have seen, in Sections 6 and 9, two linear programs, one for establishing an upper bound on performance, and another for establishing stability. What is the connection between these two linear programs? In this section we will show that they are simply the duals of each other, in the sense of linear programming. The results in this and the next section are drawn from [23].

Let us begin by considering the class of all non–idling scheduling policies.

Theorem 10: Duality of Performance LP and Drift LP.

(i) *Consider the* Performance LP[5]:

$$\text{Max} \sum_{i=1}^{L} \bar{x}_i$$

subject to

$$\bar{x}_j - \sum_{\{i:\sigma(i)=\sigma(j)\}} z_{ij} = 0 \text{ for } 1 \le j \le L, \qquad (p_j)$$

$$2\lambda \bar{x}_1 + 2\lambda - 2\mu_1 z_{11} = 0 \qquad (r_{11})$$

$$\lambda \bar{x}_2 - \lambda - \mu_1 z_{12} + \mu_1 z_{11} - \mu_2 z_{21} = 0 \qquad (r_{12})$$

$$\lambda \bar{x}_j - \mu_1 z_{1j} - \mu_j z_{j1} + \mu_{j-1} z_{j-1,1} = 0 \text{ for } 3 \le j \le L \qquad (r_{1j})$$

$$2\mu_{j-1} z_{j-1,j} + 2\lambda - 2\mu_j z_{jj} = 0 \text{ for } 2 \le j \le L \qquad (r_{jj})$$

$$\mu_{i-1} z_{i-1,i+1} - \mu_i z_{i,i+1} - \lambda + \mu_i z_{ii} - \mu_{i+1} z_{i+1,i} = 0 \text{ for } 2 \le i \le L-1 \quad (r_{i,i+1})$$

$$\mu_{i-1} z_{i-1,j} - \mu_i z_{ij} + \mu_{j-1} z_{j-1,i} - \mu_j z_{ji} = 0 \text{ for } 2 \le i \le L-2 \text{ and } i+2 \le j \le L \qquad (r_{ij})$$

$$\sum_{\{i:\sigma(i)=\sigma\}} z_{ij} - \bar{x}_j \le 0 \text{ for } 1 \le j \le L, \text{ and } \sigma \ne \sigma(j) \qquad (w_{\sigma j})$$

$$z_{ij} \ge 0.$$

Its dual is the LP,

$$\text{Min} \sum_{i=1}^{L-1} \lambda r_{i,i+1} - \sum_{i=1}^{L} \lambda r_{ii}$$

subject to

$$\lambda r_{1i} + p_i - \sum_{\{\sigma:\sigma\ne\sigma(i)\}} w_{\sigma i} \ge 1 \text{ for } 1 \le i \le L \qquad (\bar{x}_i)$$

$$-\mu_j r_{jj} + \mu_j r_{j,j+1} - p_j \ge 0 \text{ for } 1 \le j \le L \qquad (z_{jj})$$

[5]The Performance LP (25-33) given earlier, is rewritten here using $\bar{x}_j := \sum_{\{i:\sigma(i)=\sigma(j)\}} z_{ij}$.

Theorem 9: Bounds on Transient Performance. *The transient performance* $\frac{1}{N}\sum_{n=0}^{N-1} E[c^T x(\tau_n)]$ *is bounded above by the linear program*[4]*:*

$$\text{Max} \sum_{j=1}^{L} c_j \bar{x}_j$$

subject to:

$$\bar{x}_j - \sum_{\{i:\sigma(i)=\sigma(j)\}} \bar{z}_{ij}(N) = 0 \qquad 1 \le j \le L,$$

$$2\lambda \bar{x}_1 + 2\lambda - 2\mu_1 \bar{z}_{11}(N) \ge -\frac{1}{N} x_1^2(0),$$

$$2\mu_{j-1} z_{j-1,j} + 2\lambda - 2\mu_j \bar{z}_{jj}(N) \ge -\frac{1}{N} x_j^2(0) \text{ for } 2 \le j \le L,$$

$$\lambda \bar{x}_2 - \lambda - \mu_1 \bar{z}_{12}(N) + \mu_1 \bar{z}_{11}(N) - \mu_2 \bar{z}_{21}(N) \ge -\frac{1}{N} x_1(0) x_2(0),$$

$$\lambda \bar{x}_j - \mu_1 \bar{z}_{1j}(N) - \mu_j \bar{z}_{j1}(N) + \mu_{j-1} \bar{z}_{j-1,1}(N) \ge -\frac{1}{N} x_1(0) x_j(0) \text{ for } 3 \le j \le L,$$

$$\mu_{i-1} \bar{z}_{i-1,i+1}(N) - \mu_i \bar{z}_{i,i+1}(N) - \lambda + \mu_i \bar{z}_{ii}(N) - \mu_{i+1} \bar{z}_{i+1,i}(N) \ge -\frac{1}{N} x_i^2(0) \quad \text{for } 2 \le i \le L-1,$$

$$\mu_{i-1} \bar{z}_{i-1,j}(N) - \mu_i \bar{z}_{ij} + \mu_{j-1} \bar{z}_{j-1,i} - \mu_j \bar{z}_{ji}(N) \ge -\frac{1}{N} x_i(0) x_j(0) \quad \text{for } 2 \le i \le L-2 \text{ and } i+2 \le j \le L,$$

$$\sum_{\{i:\sigma(i)=\sigma\}} \bar{z}_{ij}(N) - \bar{x}_j \le 0 \text{ for } 1 \le j \le L \text{ and } \sigma \ne \sigma(j),$$

$$\bar{z}_{ij}(N) \ge 0 \text{ for } 1 \le i, j \le L.$$

The transient performance is bounded below by the same LP, with a "Min" replacing the "Max." □

We note that additional constraints (e.g., buffer priority constraints) can also be appended to the linear program for specific systems, as earlier.

[4] We have used $\bar{x}_j := \sum_{\{i:\sigma(i)=\sigma(j)\}} \bar{z}_{ij}(N)$ to simplify the LP.

bound is always tighter than Kingman's upper bound (see [20]), by at least $\rho/2$. Figure 24 presents some numerical values, for the usual "E_2" case, where $\lambda_1 = \lambda_2$. $\square$

Load Factor ρ	Exact Value	Lower Bound by LP	Upper Bound by LP	King-man's Upper Bound
0.6	1.1901	0.875	1.375	1.675
0.75	2.3229	2.0000	2.5	2.875
0.9	6.8295	6.5	7.0	7.45
0.99	74.333	74.0	74.5	74.995

Figure 24: Mean number in $E_2/M/1$ queue of Example 12.

11 Bounds on Transient Performance

We can also obtain bounds on transient performance, as shown in [23]. Consider the initial condition $x(0)$. Let us define

$$\bar{z}_{ij}(N) := \frac{1}{N}\sum_{n=0}^{N-1} E(w_i(\tau_n)x_j(\tau_n)). \tag{34}$$

Clearly,

$$\begin{aligned}\frac{1}{N}E\left[\sum_{n=0}^{N-1} E[x_i(\tau_{n+1})x_j(\tau_{n+1}) \mid \mathcal{F}_{\tau_n}] - x_i(\tau_n)x_j(\tau_n)\right]& \\ = \frac{E(x_i(\tau_N)x_j(\tau_N)) - x_i(0)x_j(0)}{N}& \\ \geq -\tfrac{1}{N}x_i(0)x_j(0).&\end{aligned}$$

The left hand side above can be computed exactly as in earlier sections, using the transition probabilities given in (4), except that we now write the final result using the averages values (34). This gives us constraints, and thus bounds, on the transient performance.

10.4 General Distributions

Many systems have non-exponential distributions for interarrival and service times. They can be approximated by phase-type distributions.

Example 12: $\mathbf{E_2}$/M/1 Queue

An $E_2/M/1$ queue can be modeled as in Figure 23, where the subnetwork on the left models the interarrival time, and a customer splits into two, one entering Machine 1, and the other going back to b_1.

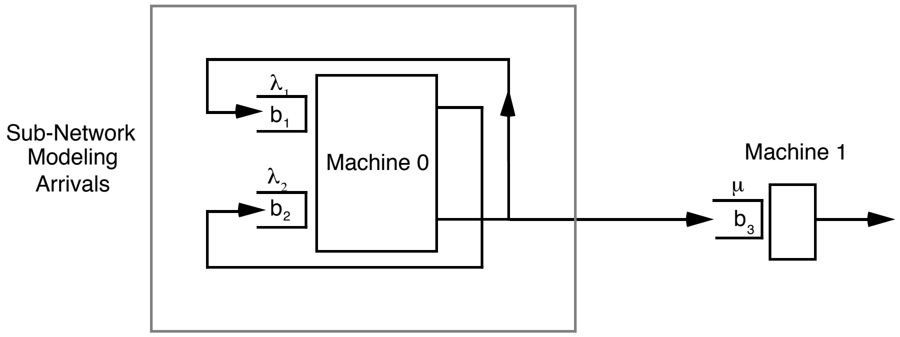


Figure 23: $E_2/M/1$ queue of Example 12.

This subnetwork has one trapped customer.

Since there is only one part in the subnetwork,

$$z_{12} = 0 \text{ and } z_{21} = 0.$$

Since one of the buffers b_1 or b_2 is always busy,

$$z_{13} + z_{23} = z_{33}.$$

In addition, one has all the other usual constraints.

For this example, one can explicitly solve for the bounds $z_{33,\min}$ and $z_{33,\max}$ on the mean number in the system. They are

$$z_{33,\min} = \text{Max}\left\{\rho, \frac{\mu\rho^2}{\lambda_2(1-\rho)}, \frac{\rho}{1-\rho} - \frac{\mu^2\rho^2}{\lambda_1\lambda_2(1-\rho)}\right\},$$

$$z_{33,\max} = \text{Min}\left\{\frac{\rho}{1-\rho}, \frac{\rho}{1-\rho} - \frac{\mu^2\rho^2}{\lambda_1\lambda_2(1-\rho)} + \frac{\lambda_2}{\lambda_1+\lambda_2}\right\}.$$

It can be checked that the upper and lower bounds differ by no more than $\frac{\lambda_2}{\lambda_1+\lambda_2} \geq \frac{1}{2}$ (without loss of generality one can take $\lambda_2 \leq \lambda_1$). Also, the upper

Figure 21: Model of system with machine failures in Example 11.

The lower closed loop contains just one VIP part. In addition to all the usual constraints, one has

$$z_{14} = z_{34} = 0,$$

to model the priority of the VIP part. Also,

$$z_{40} = 0, z_{04} = 0 \text{ and } z_{00} + z_{44} = 1,$$

since there is only one part trapped in the lower loop. Finally, since either b_0 or b_4 must always be busy, we have $z_{02} + z_{42} = z_{22}$, $z_{01} + z_{41} = z_{11} + z_{31} + z_{41}$, and $z_{03} + z_{43} = z_{13} + z_{33} + z_{43}$.

For the values $\mu_i \equiv \mu$, and $\lambda = \mu/4$, with the normalization $\lambda + \sum_{i=0}^{4} \mu_i = 1$, consider the LBFS policy. Figure 22 presents the bounds on the mean number of (real) parts in the system, as the ratio $\frac{\mu_0}{\mu_4} = \frac{MTTR}{MTTF}$ increases.

MTTR/MTTF	Lower Bound on Number in System	Upper Bound on Number in System
0.2	1.5935	2.3482
0.3	1.9233	2.9356
0.4	2.3575	3.5364
0.5	2.9583	4.1825
0.6	3.8506	5.1354
0.99	174.5401	224.3769

Figure 22: Mean number in system vs. MTTR/MTTF, under the LBFS policy, for Example 11.

It is of interest to note that the upper bound has the desirable property that it stays finite throughout the capacity region. □

$$\sum_{\{j:\sigma(j)=\sigma\}} \bar{z}_{ji} \leq \sum_{\{j:\sigma(j)=\sigma(i)\}} \bar{z}_{ji} \text{ for all } \sigma \neq \sigma(\text{i}),$$

$$\bar{z}_{ij} \geq 0.$$

□

Example 10

Consider the system of Figure 19. Let $\mu_1 = 2$, $\mu_2 = 1$, $\mu_3 = 4$ and $\mu_4 = 3$. Under the FBFS buffer priority policy using the ordering $\{b_1, b_2, b_3, b_4\}$, the value of the above LP is $\frac{9}{13}$, while $\lambda^* = \frac{3}{4}$. Hence we are unable to conclude that FBFS asymptotically attains λ^*. (A similar result can also hold for a system where all machines are equally loaded, i.e., a balanced system; see [27] for an example).

□

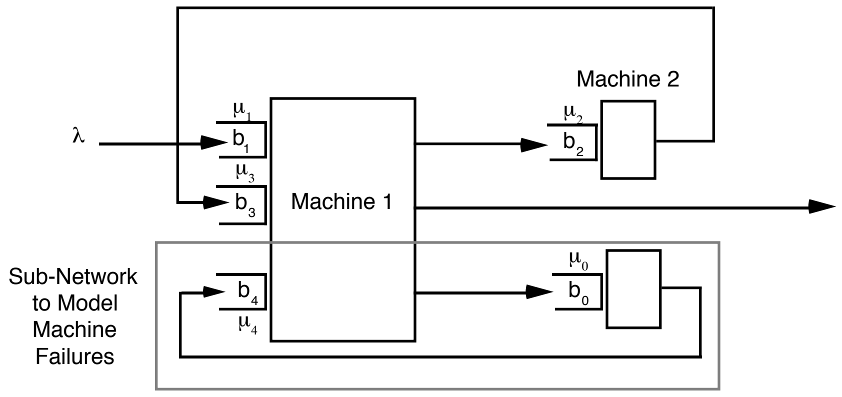
In [13] it has recently been shown that in the closed version of Counterexample 1, the throughput for some policies does not converge to λ^* as $N \nearrow +\infty$. Recently, in [17] several results are established concerning the throughput of closed queueing networks.

10.3 Systems with Machine Failures

If a machine is subject to failure, it can be modeled as being interrupted by a VIP customer.

Example 11: Modeling Machine Failures

Consider the system in Figure 21.



Let us sidestep the issue and determine a sufficient condition for the convergence of the throughput to λ^* as $N \nearrow \infty$.

We illustrate just the basic idea; the details can be found in [27]. For a population of size N, consider one of the more or less typical constraints,

$$2\mu_{j-1} z_{j-1,j} + 2\lambda - 2\mu_j z_{jj} = 0.$$

Normalizing by N, the population size, and denoting $\bar{z}_{ij} := z_{ij}/N$, we obtain,

$$2\mu_{j-1} \bar{z}_{j-1,j} + \frac{2\lambda}{N} - 2\mu_j \bar{z}_{jj} = 0.$$

As $N \nearrow \infty$, the limiting LP has the constraint,

$$2\mu_{j-1} \bar{z}_{j-1,j} - 2\mu_j \bar{z}_{jj} = 0.$$

In this way one obtains the following Theorem.

Theorem 8: Sufficient Condition for a Buffer Priority Policy to Have Maximal Throughput When Population Size Increases. *Consider a closed queueing network with a population of size N, operating under a buffer priority policy using the ordering $\{b_{\theta(1)}, b_{\theta(2)}, \ldots, b_{\theta(L)}\}$. The throughput λ_N converges to λ^* as $N \nearrow +\infty$, if the following linear program has value λ^*:*

$$\text{Min} \quad \lambda$$

subject to

$$\sum_{j=1}^{L} \sum_{\{i:\sigma(i)=\sigma(j)\}} \bar{z}_{ij} = 1,$$

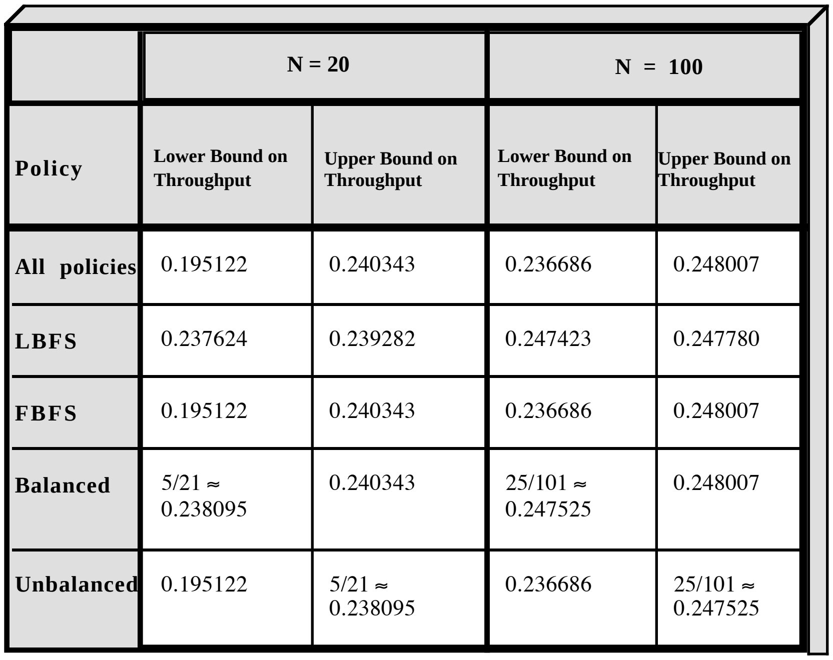
$$\sum_{j=1}^{L} \bar{z}_{ij} = \frac{\lambda}{\mu_i} \text{ for } 1 \le i \le L,$$

$$2\mu_{j-1} \bar{z}_{j-1,j} - 2\mu_j \bar{z}_{jj} = 0 \text{ for } 1 \le j \le L,$$

$$\mu_{i-1} \bar{z}_{i-1,i+1} - \mu_i \bar{z}_{i,i+1} + \mu_i \bar{z}_{ii} - \mu_{i+1} \bar{z}_{i+1,i} = 0 \text{ for } 1 \le i \le L,$$

$$\begin{aligned} \mu_{i-1} \bar{z}_{i-1,j} - \mu_i \bar{z}_{ij} + \mu_{j-1} \bar{z}_{j-1,i} - \mu_j \bar{z}_{ji} &= 0 \\ \text{for } 1 \le i \le L-2 \text{ and } i+2 \le j &\le L \\ \textit{with } j \not\equiv i + 1 (\textit{mod } L&), \end{aligned}$$

$$\bar{z}_{ij} = 0 \text{ for } \sigma(i) = \sigma(j) \text{ and } \theta(j) < \theta(i),$$



	N = 20		N = 100	
Policy	**Lower Bound on Throughput**	**Upper Bound on Throughput**	**Lower Bound on Throughput**	**Upper Bound on Throughput**
All policies	0.195122	0.240343	0.236686	0.248007
LBFS	0.237624	0.239282	0.247423	0.247780
FBFS	0.195122	0.240343	0.236686	0.248007
Balanced	5/21 ≈ 0.238095	0.240343	25/101 ≈ 0.247525	0.248007
Unbalanced	0.195122	5/21 ≈ 0.238095	0.236686	25/101 ≈ 0.247525

Figure 20: Comparison of policies for closed re–entrant line of Example 9.

The Balanced policy has higher throughput than the Unbalanced policy. □

An issue which arises in the study of closed queueing networks is whether, under a particular scheduling policy, the throughput attained converges as $N \nearrow \infty$, to the maximal throughput attainable for that system. For a system with a fixed route, the maximal throughput attainable is the reciprocal of the mean work per part, brought to the bottleneck machine, in one cycle. It is

$$\lambda^* = \min_{\sigma} \frac{1}{\sum_{\{i:\sigma(i)=\sigma\}} \frac{1}{\mu_i}}.$$

(The definition is easily extended when the route is random).

Instead of studying the limit of the bounds of the LP's as $N \nearrow \infty$, one can directly study the bounds on the *limiting LP*. There is an issue of continuity here: Is the limit of the values of a sequence of LP's equal to the value of the limit of the sequence of LP's? This is not always so, and "regularity conditions" under which such equality holds are known in the literature. (See [33] where it is shown that the result holds for perturbations of the RHS's of the constraints, when the set of feasible solutions to the dual is compact).

Example 9: Comparison of Buffer Priority Policies for a Balanced Two Station System

Consider the system shown in Figure 19, where $\mu_1 = 1/3$, $\mu_2 = 2/7$, $\mu_3 = 1$ and $\mu_4 = 2$.

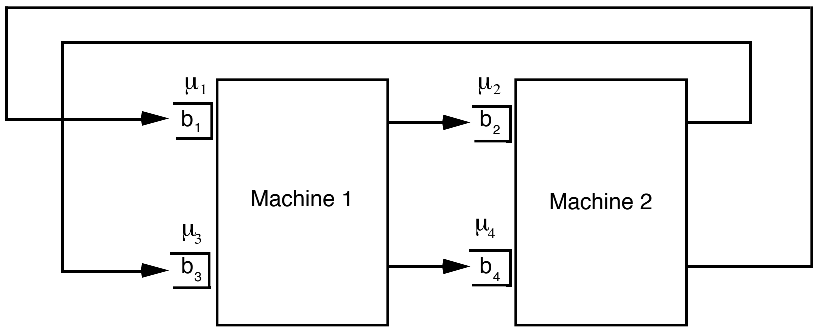


Figure 19: Closed re–entrant line of Examples 9 and 10.

There are four buffer priority policies: LBFS = $\{b_4, b_3, b_2, b_1\}$, FBFS = $\{b_1, b_2, b_3, b_4\}$, Harrison and Wein's Balanced Policy[3] = $\{b_4, b_1, b_3, b_2\}$, and the "Unbalanced" Policy = $\{b_3, b_2, b_1, b_4\}$.

Figure 20 presents upper and lower bounds on throughput for these four buffer priority policies, as well as the class of all non-idling policies, for two population sizes, $N = 20$ and 100.

[3]Based on the analysis of a Brownian Network model, it is conjectured in [14] that a certain method of ordering the buffers has optimal throughput as $N \nearrow \infty$. In [17] it has recently been shown that the throughput does indeed converge to the maximal throughput.

Policy	**Lower Bound on Number in System**	**Upper Bound on Number in System**
LBFS	1.06667	1.60075
FBFS Bounds Without Tandem Constraints	1.6141	2.48964
FBFS Bounds With Tandem Constraints	1.73718	2.48964

Figure 18: Comparison of LBFS and FBFS policies for Example 8.

We see that LBFS is better than FBFS, since its upper bound is lower than the lower bound for FBFS. □

10.2 Closed Queueing Networks

If a queueing network is closed, then the constraint,

$$\sum_{j=1}^{L} \sum_{\{i:\sigma(i)=\sigma(j)\}} z_{ij} = N$$

is satisfied, where N is the population size. Moreover, if one samples the system whenever a particular buffer is being worked on, then one always sees N. Hence

$$\sum_{j=1}^{L} z_{ij} = \frac{\lambda}{\mu_i} N,$$

where

λ := system throughput, i.e., the rate that parts circulate.

The earlier equations (26-31) need to be modified slightly, since there are no exogenous arrivals. Thus one obtains linear programs for upper and lower bounds on the throughput λ.

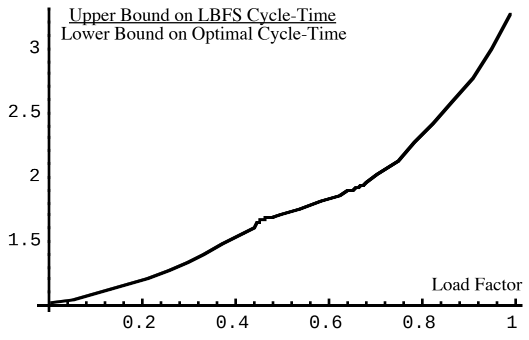


Figure 17: The ratio $\frac{\text{Upper bound on LBFS cycle–time}}{\text{Lower bound on optimal cycle–time}}$ in Example 7.

One notes that LBFS is nearly optimal in light traffic, and never worse than about $3\frac{1}{2}$ times optimal, even in heavy traffic. $\square$

Example 8: Comparison of LBFS and FBFS

Based on a myopic application of Little's Theorem, one believes that, for many systems, LBFS may be better than FBFS. Consider the system shown in Figure 11, with $\mu_1 = \mu_2 = \mu_3/7 = \mu_4/7$, and $\rho_1 = \rho_2 = 0.4$.

Under (the pre-emptive) FBFS, from Burke's Theorem, we know that the departure stream leaving b_1 is Poisson, and moreover its past is independent of the present value of x_1. Thus one has independence of w_1 and x_2. Similarly, one has independence of w_2 and x_1. These give the additional constraints,

$$z_{12} = \frac{\lambda}{\mu_1} z_{22}, \text{ and } z_{21} = \frac{\lambda}{\mu_2} z_{11}.$$

Figure18 provides the lower and upper bounds on FBFS performance, both with and without these "tandem" constraints, and on LBFS performance.

$\square$

10 Additional Linear Constraints for Specific System Models

Queueing networks and scheduling policies often possess much special structure. For example, the system may be a closed network with a fixed population of jobs, machines may fail, the service distributions may not be exponential, a specific buffer priority policy may be in place, etc. In this section we wish to show how such specific structures can be exploited to provide better bounds. We will show that they provide additional linear constraints which can be appended to the linear programs.

The following examples are drawn from [27], to which we refer the further details.

10.1 Buffer Priority Policies

Consider a buffer priority policy using the ordering $\{b_{\theta(1)}, b_{\theta(2)}, \ldots, b_{\theta(L)}\}$. Clearly, if b_i and b_j share the same machine, and b_i has higher priority than b_j, then

$$x_i(\tau_n) \geq 1 \quad \Rightarrow \quad w_j(\tau_n) = 0.$$

Hence

$$x_i(\tau_n) w_j(\tau_n) = 0.$$

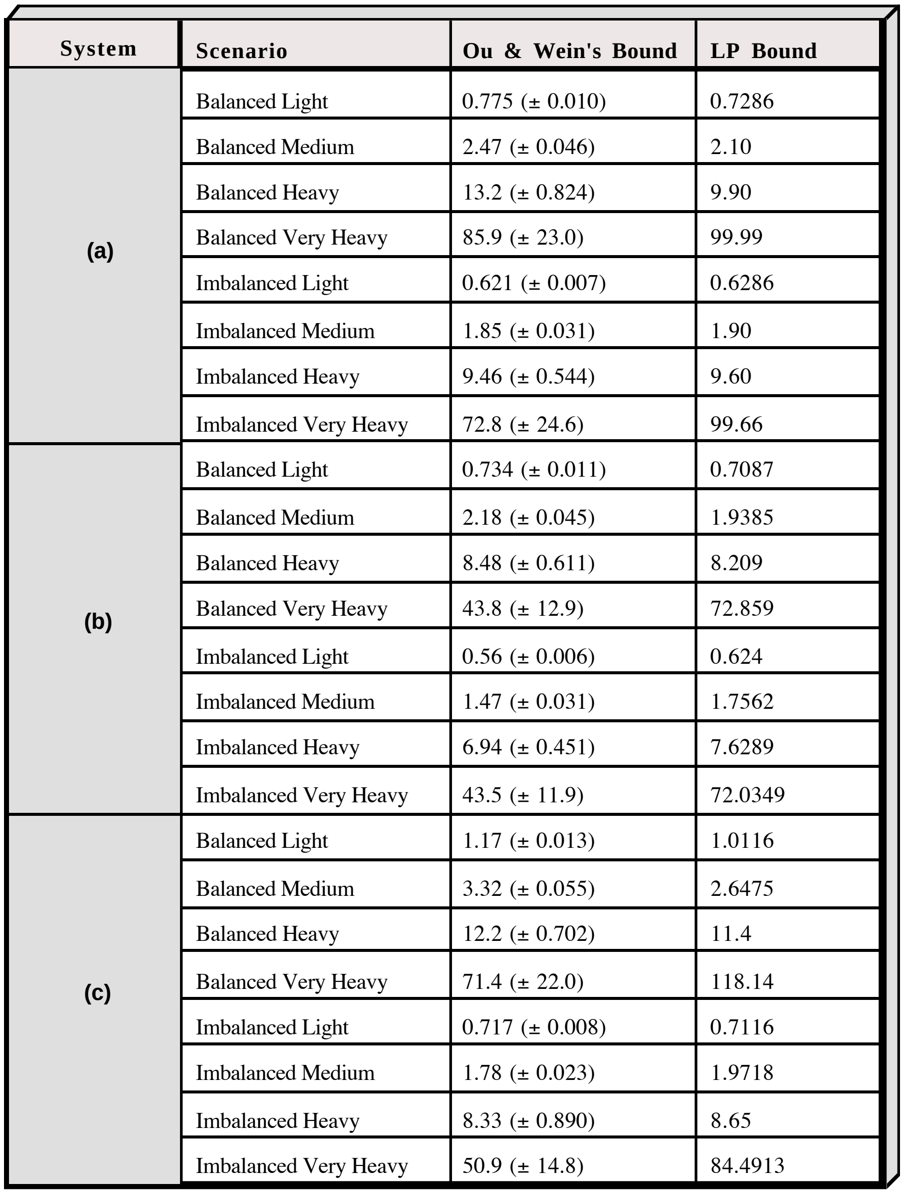
Thus, we obtain the additional *buffer priority constraints*,

$$z_{ij} = 0 \text{ whenever } \sigma(i) = \sigma(j) \text{ and } \theta(i) < \theta(j).$$

We should note that for buffer priority policies, an upper bound as given in Theorem 7, with the buffer priority constraints appended to the LP, does *not* apply unless the policy has a finite first moment. The reason is that the proof that one can dispense with the finite first moment condition for the class of all non–idling policies, used the facts that the class of all non–idling policies contains a stable policy, and that it is closed under composition. Those facts need not hold for restrictive classes of policies.

Example 7: Performance of LBFS

Consider the system shown in Figure11. Let $\mu_i \equiv \mu$. For the LBFS policy, one can determine an upper bound on its performance, and compare it to the lower bound for all non-idling policies. This ratio, which bounds the deviation from optimality, is plotted in Figure 17, as a function of system load.



System	Scenario	Ou & Wein's Bound	LP Bound
(a)	Balanced Light	0.775 (± 0.010)	0.7286
	Balanced Medium	2.47 (± 0.046)	2.10
	Balanced Heavy	13.2 (± 0.824)	9.90
	Balanced Very Heavy	85.9 (± 23.0)	99.99
	Imbalanced Light	0.621 (± 0.007)	0.6286
	Imbalanced Medium	1.85 (± 0.031)	1.90
	Imbalanced Heavy	9.46 (± 0.544)	9.60
	Imbalanced Very Heavy	72.8 (± 24.6)	99.66
(b)	Balanced Light	0.734 (± 0.011)	0.7087
	Balanced Medium	2.18 (± 0.045)	1.9385
	Balanced Heavy	8.48 (± 0.611)	8.209
	Balanced Very Heavy	43.8 (± 12.9)	72.859
	Imbalanced Light	0.56 (± 0.006)	0.624
	Imbalanced Medium	1.47 (± 0.031)	1.7562
	Imbalanced Heavy	6.94 (± 0.451)	7.6289
	Imbalanced Very Heavy	43.5 (± 11.9)	72.0349
(c)	Balanced Light	1.17 (± 0.013)	1.0116
	Balanced Medium	3.32 (± 0.055)	2.6475
	Balanced Heavy	12.2 (± 0.702)	11.4
	Balanced Very Heavy	71.4 (± 22.0)	118.14
	Imbalanced Light	0.717 (± 0.008)	0.7116
	Imbalanced Medium	1.78 (± 0.023)	1.9718
	Imbalanced Heavy	8.33 (± 0.890)	8.65
	Imbalanced Very Heavy	50.9 (± 14.8)	84.4913

Figure 16: Comparison of LP lower bounds with Ou and Wein's bounds, for systems (a), (b), and (c) of Example 6.

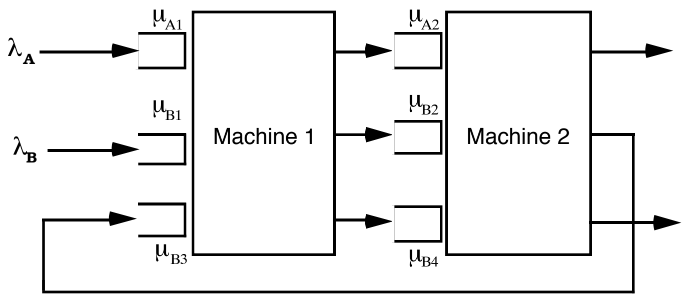


Figure 14: Ou and Wein's system (b) of Example 6.
Balanced: $\mu_{A1} = 1/4$, $\mu_{A2} = 1$, $\mu_{B1} = 1/8$, $\mu_{B2} = 1/6$, $\mu_{B3} = 1/2$, $\mu_{B4} = 1/7$.
Imbalanced: $\mu_{A1} = 1/4$, $\mu_{A2} = 2/3$, $\mu_{B1} = 1/8$, $\mu_{B2} = 1/4$, $\mu_{B3} = 1/2$, $\mu_{B4} = 3/14$.
Light: $\lambda_A = \lambda_B = 3/140$. Medium: $\lambda_A = \lambda_B = 6/140$.
Heavy: $\lambda_A = \lambda_B = 9/140$. Very Heavy: $\lambda_A = \lambda_B = 99/1400$.

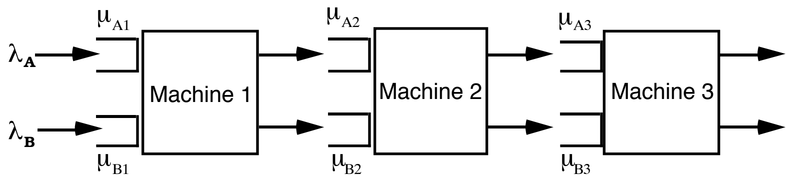


Figure 15: Ou and Wein's system (c) of Example 6.
Balanced: $\mu_{A1} = 1/2$, $\mu_{A2} = 1/4$, $\mu_{A3} = 1/6$, $\mu_{B1} = 1/7$, $\mu_{B2} = 1/5$, $\mu_{B3} = 1/3$,
Imbalanced: $\mu_{A1} = 1/2$, $\mu_{A2} = 1/2$, $\mu_{A3} = 1$, $\mu_{B1} = 1/7$, $\mu_{B2} = 1/4$, $\mu_{B3} = 1/2$
Light: $\lambda_A = \lambda_B = 1/30$. Medium: $\lambda_A = \lambda_B = 2/30$.
Heavy: $\lambda_A = \lambda_B = 0.1$. Very Heavy: $\lambda_A = \lambda_B = 0.11$.

The results are given in Figure 16.

Figure 11: The open re–entrant line of Examples 5, 7, and 8.

We take $\mu_1 = \mu_2 = \mu_3 = \mu_4 =: \mu$. The load factor on either machine is $\rho_1 = \rho_2 = \rho := \frac{2\lambda}{\mu}$. Note that by Little's Theorem, the mean cycle-time of a part is given by $\frac{\text{Mean number in system}}{\lambda}$. Note that $\frac{4}{\mu}$ is the mean total processing time of part. Figure 12 plots the lower bound on the attainable cycle–time multiplier ratio $\frac{\text{Mean Cycle–Time}}{\text{Mean Total Processing Time}}$ as a function of the system load ρ.

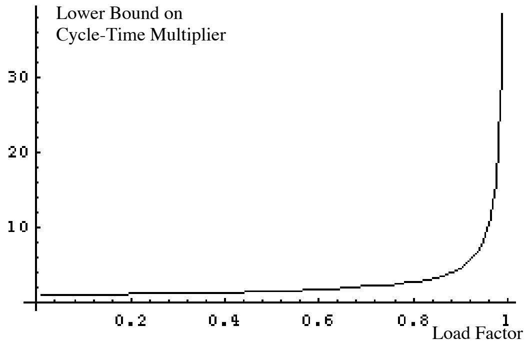


Figure 12: Lower bound on cycle–time multiplier for Example 5.

Example 6: Comparison with Lower Bounds of Ou and Wein

Ou and Wein [35] have proposed a different method for obtaining bounds. They determine a system whose cost is lower than all the policies in the original system. Then they simulate the new system, thus obtaining a confidence interval for a lower bound. In [27], the lower bound is compared with the 95% confidence intervals of [35], for the systems shown in Figures 13, 14, and 15.

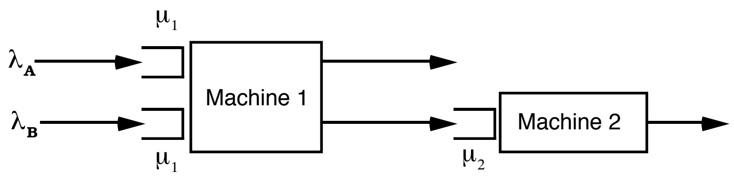


Figure 13: Ou and Wein's system (a) of Example 6.
Balanced: $\mu_1 = 2, \mu_2 = 1$. Imbalanced: $\mu_1 = 2$, $\mu_2 = 1.5$.
Light: $\lambda_A = \lambda_B = 0.3$. Medium: $\lambda_A = \lambda_B = 0.6$.
Heavy: $\lambda_A = \lambda_B = 0.9$. Very Heavy: $\lambda_A = \lambda_B = 0.99$.

with finite moments of all orders. However, the policy is not non-idling. This is easily remedied. Consider the modified policy

$$w_i(\tau_n) = \frac{\frac{1}{\mu_i} 1(x_i(\tau_n) \geq 1)}{\sum_{\{j:\sigma(j)=\sigma(i)\}} \frac{1}{\mu_j} 1(x_j(\tau_n) \geq 1)}$$

(taking $0/0 = 0$). This restricts service attention to non-empty buffers only, and is therefore non-idling. Moreover, a simple sample path argument shows that every part has a smaller delay under this policy, than under the earlier sometimes idling policy. Hence the new non-idling stationary policy is also stable, with finite moments of all orders. Call this policy p.

Now suppose that the value (25) of the above LP is finite, say M. We will now show that every non-idling policy is then stable, and has a mean number of parts bounded by M.

To see this, consider any non-idling policy $\bar{p}$. This policy may not be stable. However consider the composite policy $\bar{p}_N$ which follows policy $\bar{p}$ whenever $|x| \leq N$, but follows the policy p (which has a finite first moment) whenever $|x| > N$.

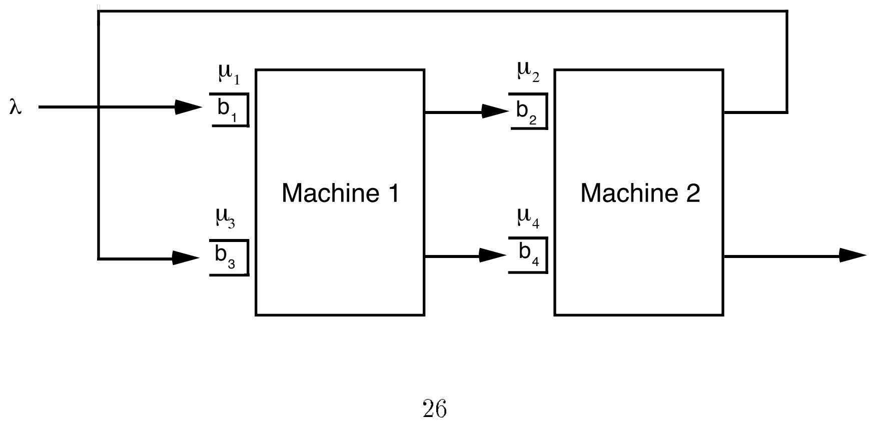
The composite policy $\bar{p}_N$ is stable, with a finite first moment, since whenever the state leaves a compact set, it applies a good policy. Hence its performance is bounded by M, uniformly for all N, since the LP bound applies to all stable policies with a finite first moment. However, as $N \to +\infty$, $\bar{p}_N$ converges to $\bar{p}$ pointwise. Applying weak convergence arguments, we see that the mean number under $\bar{p}$ is also bounded by M.

Thus, all non-idling policies are stable, and in fact have a mean number of parts bounded by M. $\square$

The following examples are drawn from [27].

Example 5: Lower Bound on Cycle-Time Multiplier

Consider the re-entrant line shown in Figure 11.



$$\lambda \sum_{\{i:\sigma(i)=\sigma(j)\}} z_{ij} - \mu_1 z_{1j} - \mu_j z_{j1} + \mu_{j-1} z_{j-1,1} = 0$$
$$\text{for } j = 3, \ldots, L, \tag{29}$$

$$\mu_{i-1} z_{i-1,i+1} - \mu_i z_{i,i+1} - \lambda + \mu_i z_{ii} - \mu_{i+1} z_{i+1,i} = 0$$
$$\text{for } i = 2, \ldots, L-1, \tag{30}$$

$$\mu_{i-1} z_{i-1,j} - \mu_i z_{i,j} + \mu_{j-1} z_{j-1,i} - \mu_j z_{ji} = 0$$
$$\text{for } i = 2, \ldots, L-2, \text{ and } j = i+2, \ldots, L \tag{31}$$

$$\sum_{\{j:\sigma(j)=\sigma\}} z_{ji} \le \sum_{\{j:\sigma(j)=\sigma(i)\}} z_{ji} \text{ for } i = 1, \ldots, L, \text{ and } \sigma \neq \sigma(i) \tag{32}$$

$$z_{ij} \ge 0 \text{ for } i = 1, \ldots, L \text{ and } j = 1, \ldots L. \tag{33}$$

(ii) *The mean number of parts in steady-state is bounded below by the same linear program, with a "min" replacing the "max" in (25).*

Proof:

The above theorem has already been established if the non–idling policy under consideration has a finite first moment, since that property was used to derive the constraints (26-31). The only fact that has not yet been established is that the upper bound also applies to policies which may not have a finite first moment. This is done in [23] as follows.

The key idea is that every non-idling policy can be written as the limit of stable policies.

We will consider a policy which uses time sharing. Thus the $w_i(\tau_n)$'s may not be just 0 or 1, but are allowed to take values in between, subject only to the non–idling restriction (5). However, our theory so far has not really needed this 0 or 1 property.

Consider the stationary scheduling policy,

$$w_i(\tau_n) = \frac{\frac{1}{\mu_i}}{\sum_{\{j:\sigma(j)=\sigma(i)\}} \frac{1}{\mu_j}}.$$

This policy allocates a fixed proposition of a machine's capacity to each buffer, and the capacity provided is enough to meet the buffers needs. Thus, under it, the system behaves like a tandem of $M/M/1$ buffers, and is clearly stable, In addition, the variables $\{z_{ij}\}$ satisfy several other inequality constraints. Consider a station $\sigma \neq \sigma(j)$. It may or may not be working when $x_j(\tau_n) \geq 1$. Hence for $\sigma \neq \sigma(j)$, one only has,

$$x_j(\tau_n) \geq 1 \Rightarrow \sum_{\{i:\sigma(i)=\sigma\}} w_i(\tau_n) \leq 1.$$

Hence

$$\sum_{\{i:\sigma(i)=\sigma\}} w_i(\tau_n) x_j(\tau_n) \leq x_j(\tau_n). \tag{24}$$

Thus, upon taking expectations in (24), and using (22), we obtain the *non-idling inequality constraints* (32), shown in Theorem 7 below.

Also, clearly all the z_{ij}'s are non-negative, yielding (33).

Note finally that due to (22), the mean number in the system can be written in terms of the z_{ij}'s as,

$$E|x(\tau_n)| = \sum_{j=1}^{L} \sum_{\{i:\sigma(i)=\sigma(j)\}} z_{ij}.$$

Thus we see that the mean number is given by a linear expression in the z_{ij}'s, which in turn are constrained by several linear equalities and inequalities. The situation is ripe for linear programming.

Theorem 7: Performance Bounds for All Non–Idling Policies. *Consider any non-idling stationary scheduling policy.*

(i) *In steady–state, the mean number of parts is bounded above by the value of the linear program:*

$$\text{Max} \sum_{j=1}^{L} \sum_{\{i:\sigma(i)=\sigma(j)\}} z_{ij} \tag{25}$$

subject to

$$2\lambda \sum_{\{i:\sigma(i)=\sigma(1)\}} z_{i1} + 2\lambda - 2\mu_1 z_{11} = 0 \tag{26}$$

$$2\mu_{j-1} z_{j-1,j} + 2\lambda - 2\mu_j z_{jj} = 0 \text{ for } j = 2, \ldots, L \tag{27}$$

$$\lambda \sum_{\{j:\sigma(j)=\sigma(2)\}} z_{j2} - \lambda - \mu_1(z_{12} - z_{11}) - \mu_2 z_{21} = 0, \tag{28}$$

For $i = 1$ and $j = 2$,

$$x_1(\tau_{n+1})x_2(\tau_{n+1}) = \begin{cases} (x_1(\tau_n)+1)x_2(\tau_n) & \text{w.p. } \lambda, \\ (x_1(\tau_n)-1)(x_2(\tau_n)+1) & \text{w.p. } w_1(\tau_n)\mu_1, \\ x_1(\tau_n)(x_2(\tau_n)-1) & \text{w.p. } w_2(\tau_n)\mu_2. \end{cases}$$

It is a simple matter to write the other cases down.

We illustrate the consequence of (16), for $i = j = 1$. From (18), we see that (17) evaluates to

$$E[x_1^2(\tau_{n+1}) - x_1^2(\tau_n) \mid \mathcal{F}_{\tau_n}] = \lambda(2x_1(\tau_n)+1) - \mu_1 w_1(\tau_n)(2x_1(\tau_n)-1). \quad (19)$$

Let us define

$$z_{ij} := E[w_i(\tau_n)x_j(\tau_n)]. \quad (20)$$

Taking unconditional expectations in (19), we see that

$$E[x_1^2(\tau_{n+1}) - x_1^2(\tau_n)] = 2\lambda E[x_1(\tau_n)] + \lambda - 2\mu_1 z_{11} + \mu_1 E[w_1(\tau_n)]. \quad (21)$$

Now recall that since our scheduling policy is non-idling, we have the condition (5). Hence, $x_j(\tau_n) = \sum_{\{i:\sigma(i)=\sigma(j)\}} w_i(\tau_n)x_j(\tau_n)$, and so

$$E[x_j(\tau_n)] = \sum_{\{i:\sigma(i)=\sigma(j)\}} z_{ij}. \quad (22)$$

Also, since the system is stable, every buffer b_i is worked on for a proportion of time that is exactly enough to meet the work arriving for b_i. Hence,

$$E(w_i(\tau_n)) = \frac{\lambda}{\mu_i}. \quad (23)$$

Substituting (22) and (23) in (21), we see that

$$E[x_1^2(\tau_{n+1}) - x_1^2(\tau_n)] = 2\lambda \sum_{\{i:\sigma(i)=\sigma(1)\}} z_{i1} + 2\lambda - 2\mu_1 z_{11}.$$

Hence, from (16), for $i = j = 1$, we obtain the equality,

$$2\lambda \sum_{\{i:\sigma(i)=\sigma(1)\}} z_{i1} + 2\lambda - 2\mu_1 z_{11} = 0.$$

Similarly, by considering all the other cases for (i,j), one can obtain $L(L+1)/2$ equality constraints in the L^2 variables $\{z_{ij}\}$. These are shown in equations (26-31), in Theorem 7 below. We note that these constraints are obtained independently in [3] and [27].

since $x(\tau_n)$ takes bounded jumps by (4). From the Optional Sampling Theorem of Doob [11], we can deduce (i) (see [23] for details).

To see (ii) we simply note that one can perturb the stationary policy so that all non-empty buffers get a very small service attention. This makes the whole state space reachable, while preserving the negative drift. Thus the perturbed policy does not have a finite first moment. $\square$

9 Performance Bounds for Queueing Networks and Scheduling Policies

We now examine the issue of quantifying the performance of a system. The results in this section and the next are mainly drawn from [27]. Some sharpenings of the results are from [23].

Suppose now that, under a particular stationary non-idling scheduling policy, the system is stable, i.e., positive recurrent. Suppose moreover that the system has a *finite first moment in steady-state*, i.e.,

$$E|x(\tau_n)| < +\infty, \tag{14}$$

in steady state. Then, it can be shown that,

$$E[x^T(\tau_{n+1})Qx(\tau_{n+1}) - x^T(\tau_n)Qx(\tau_n)] = 0, \tag{15}$$

for *every* matrix Q. This is particularly easy to see if

$$E|x(\tau_n)|^2 < +\infty$$

in steady state, but the result (15) also holds when just (14) holds, as shown in [23].

Relation (15) is equivalent to,

$$E[x_i(\tau_{n+1})x_j(\tau_{n+1}) - x_i(\tau_n)x_j(\tau_n)] = 0 \text{ for all } i, j. \tag{16}$$

Note that this is a set of $\frac{L(L+1)}{2}$ equalities, one for each distinct q_{ij}.

To compute the left hand side of (16), we begin by computing the conditional expectation

$$E[x_i(\tau_{n+1})x_j(\tau_{n+1}) - x_i(\tau_n)x_j(\tau_n) \mid \mathcal{F}_{\tau_n}], \tag{17}$$

using the transition probabilities (4). There are several cases. Consider, for example, $i = j = 1$:

$$x_1^2(\tau_{n+1}) = \begin{cases} (x_1(\tau_n)+1)^2 & \text{w.p. } \lambda, \\ (x_1(\tau_n)-1)^2 & \text{w.p. } w_1(\tau_n)\mu_1, \\ x_1^2(\tau_n) & \text{otherwise.} \end{cases} \tag{18}$$

In fact, in [8], it has recently been established that FBFS is stable for all re-entrant lines (within their capacity region, of course), while in [7] and [26], it has been established that the LBFS policy is also stable for all re-entrant lines. In [25] it is shown that the FSMCT policy of Section 14.1 is also similarly stable.

8 Instability and Non–Robustness of Systems

We have seen that one can (when possible) use linear programming to construct a matrix $Q = Q^T$ that has the negative drift property (6). If the resulting matrix Q is copositive, then we have been able to deduce geometric convergence of an exponential moment, for the stationary policy under consideration. What can we conclude if Q is *not* copositive?

We will show below that if (6) holds for a non-copositive $Q = Q^T$, then either the stationary policy under consideration does *not* have a finite first moment, or there exist arbitrarily small perturbations of the policy which do not have a finite first moment. Thus, at best, the stationary policy is highly non-robust.

Let $\mathcal{C}$ denote the closed communicating class of the Markov chain resulting from the stationary policy. We recall that under stationary non–idling scheduling policies, the Markov chain has only a single closed communicating class which is also aperiodic. The following result is from [23].

Theorem 6: Instability or Non-Robustness When Q is Not Copositive. *Suppose for a stationary policy there exists a non-copositive $Q = Q^T$ such that the negative drift condition (6) holds.*

(i) *If $\inf_{x\in\mathcal{C}} x^TQx = -\infty$, then the stationary policy does not have a finite first moment.*

(ii) *If $\inf_{x\in\mathcal{C}} x^TQx > -\infty$, then either the stationary policy itself, or an arbitrarily small perturbation of it, does not have a finite first moment.*

Proof:

Consider the function $V(x) := \max\{-x^TQx, 0\}$. Since Q is not copositive, $V(x) \not\equiv 0$ in the positive orthant. In fact since one can scale x, $\inf_{x\in R_L^+} x^TQx = -\infty$. Simple calculations (see [23]) show that $V(x)$ has a nonnegative drift outside a compact set; more specifically,

$$E[V(x(\tau_{n+1})) \mid \mathcal{F}_{\tau_n}] \geq V(x(\tau_n)) \text{ for } |x(\tau_n)| \geq \frac{c}{\gamma}.$$

Since $V(x) = O(|x|^2)$, we see that $E[V(x(\tau_{n+1})) \mid \mathcal{F}_{\tau_n}] \geq V(x(\tau_n))$ whenever $V(x(\tau_n)) > m$, for some m. Moreover,

$$V(x(\tau_{n+1})) - V(x(\tau_n)) = O(|x(\tau_n)|),$$

Figure 10: Dai–Wang system of Example 3.

Dai and Wang [10] and Dai and Nguyen [9] have shown that this system does not admit a Brownian network approximation in the heavy traffic limit $\rho_1 = \rho_2 =: \rho \nearrow 1$. Using our LP of Theorem 4.ii, we see that for

$$\rho < 0.528,$$

the LP has value 1, while it is zero for all larger ρ.

In [12,7] a slightly larger value of ρ is determined for which stability holds. In fact, simulations in [29] show that the system appears unstable for even larger values.

7 Stability of Buffer Priority Policies

In many cases, while all non-idling policies may not be stable, specific buffer priority or other policies may be stable. One would therefore like to conduct a more specific test. We illustrate the idea by showing how one can analyze a given buffer priority policy.

Consider a buffer priority policy using the ordering $\{b_{\theta(1)}, b_{\theta(2)}, \ldots, b_{\theta(L)}\}$. Clearly, if $\sigma(i) = \sigma(j)$ and $\theta(j) < \theta(i)$, then buffer b_j is preferred to b_i. Thus, if $x_j(\tau_n) \geq 1$, one has $w_i(\tau_n) = 0$. Hence, in (11), one can drop such $w_i(\tau_n)$'s. This allows us to refine (12), and obtain the following theorem.

Theorem 5: Stability of Buffer Priority Policies.
Suppose $\{b_{\theta(1)}, b_{\theta(2)}, \ldots, b_{\theta(L)}\}$ is a buffer priority policy. Suppose there exists a copositive matrix Q such that,

$$\begin{aligned}\lambda q_{1j} + \max_{\{i:\sigma(i)=\sigma(j) \text{ and } \theta(i)\leq\theta(j)\}} \mu_i(q_{i+1,j} - q_{ij})\\ \sum_{\{\sigma:\sigma\neq\sigma(j)\}} \max_{\{i:\sigma(i)=\sigma\}} \mu_i(q_{i+1,j} - q_{ij})^+ \leq -1,\end{aligned} \tag{13}$$

then the system has geometrically converging exponential moment. □

The test of (13) is conducted, as before, by linear programming.

Example 4

Consider the system shown in Figure 8. We assume that $\mu_1 = \mu_3$, while μ_2 may be different. By computation, one can check that for every $\rho_1 := \frac{2\lambda}{\mu_1} < 1$, and $\rho_2 < 1$, both the FBFS and LBFS buffer priority policies are stable. □

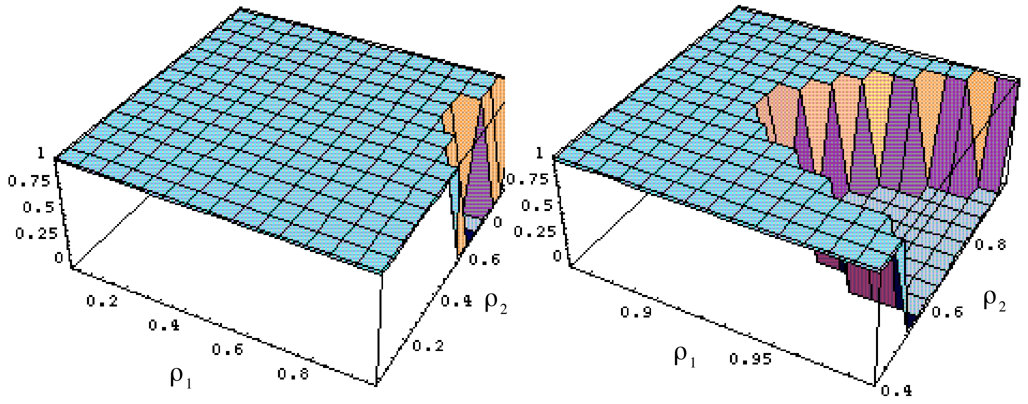


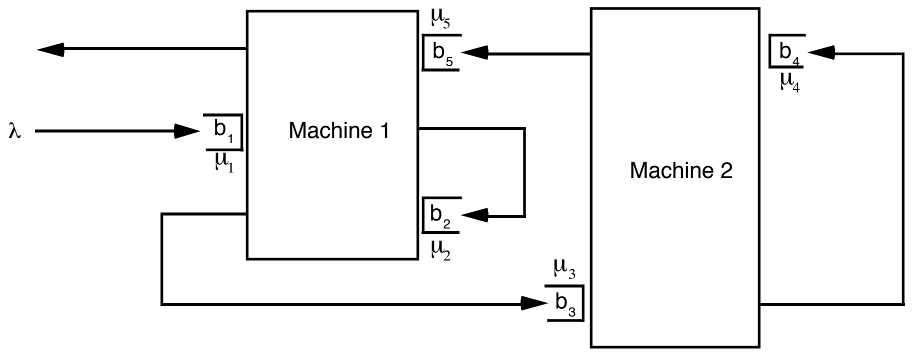
Figure 9: Value of LP for all non–idling, non–interruptive policies in Example 2. The figure on the right is a more detailed view of one corner of the figure on the left.

We see that there is a small region where the value of the LP is zero, and in this region, our test for the stability of all non-idling polices is inconclusive. □

In [12] and [7] it is shown that the system is stable even in the small indeterminate region above.

Example 3

Consider the system shown in Figure10.



Example 1

Consider the system shown in Figure 8.

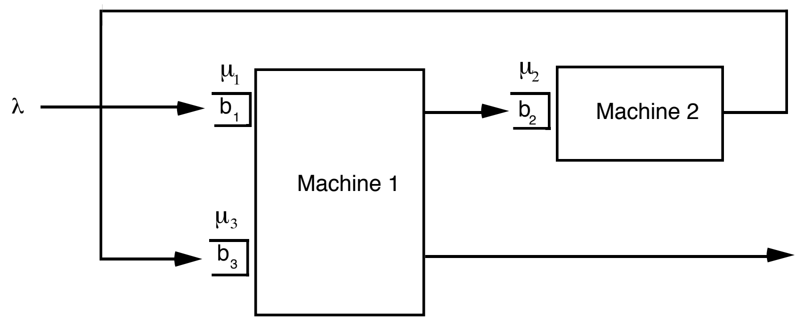


Figure 8: System of Examples 1, 2 and 4.

Suppose that $\mu_1 = \mu_2 = \mu_3 := \mu$. The capacity condition is $\rho := \frac{2\lambda}{\mu} < 1$. It is easy to check, *analytically*, that the LP in Theorem 4.ii has a solution with $\gamma = 1$. Hence, all non-idling policies are stable. $\square$

Example 2

Consider the same system as in Figure 8, except now that while we still consider $\mu_1 = \mu_3$, we let μ_2 be different. Then, the capacity condition is

$$\rho_1 := \frac{2\lambda}{\mu_1} < 1, \text{ and } \rho_2 := \frac{\lambda}{\mu_2} < 1.$$

In Figure 9 we plot the value of the LP in Theorem 4.ii for all (ρ_1, ρ_2) in the capacity region.

may or may not be working. Hence,

$$\sum_{\{\sigma:\sigma\neq\sigma(j)\}} \sum_{\{i:\sigma(i)=\sigma\}} \mu_i(q_{i+1,j} - q_{ij}) w_i(\tau_n) \le \sum_{\{\sigma:\sigma\neq\sigma(j)\}} \max_{\{i:\sigma(i)=\sigma\}} \mu_i(q_{i+1,j} - q_{ij})^+ .$$

Thus we see that (6) holds whenever

$$\lambda q_{1j} + \max_{\{i:\sigma(i)=\sigma(j)\}} \mu_i(q_{i+1,j} - q_{ij}) + \sum_{\{\sigma:\sigma\neq\sigma(j)\}} \max_{\{i:\sigma(i)=\sigma\}} \mu_i(q_{i+1,j} - q_{ij})^+ \le -\gamma \text{ for } j = 1, \dots, L. \quad (12)$$

Note now that if such a $Q = Q^T$ exists, then we can take $\gamma = 1$, since we can always scale Q. Thus, one can search for the largest γ in $0 \le \gamma \le 1$, for which (12) holds for some Q. This largest γ will either be 0 or 1. If it is 1, then we have established the stability of *all* non-idling scheduling policies. If it is 0, then we have obtained no conclusion.

Theorem 4: A Linear Programming Stability Test. *Consider the linear program:*

$$\text{Max } \gamma$$

subject to

$$\begin{gathered} 0 \le \gamma < 1 \\ q_{ij} = q_{ji} \textit{ for all } i, j \\ q_{L+1,j} = 0 \textit{ for all } j \\ \lambda q_{1j} + r_j + \textstyle\sum_{\{\sigma:\sigma\neq\sigma(j)\}} s_{\sigma j} \le -\gamma \textit{ for all } j \\ r_j \ge \mu_i(q_{i+1,j} - q_{ij}) \textit{ for all } i \textit{ with } \sigma(i) = \sigma(j), \textit{ and all } j \\ s_{\sigma j} \ge \mu_i(q_{i+1,j} - q_{ij}) \textit{ for all } i \textit{ with } \sigma(i) = \sigma \neq \sigma(j), \textit{ and all } j, \sigma \\ s_{\sigma j} \ge 0 \textit{ for all } \sigma, j. \end{gathered}$$

i) *Suppose the value of the LP is 1, and suppose Q is an optimal solution. If an optimal Q is copositive, then* all *non-idling scheduling policies are stable-in-the-mean. Also, for all non-idling stationary scheduling policies, the system has a geometrically converging exponential moment.*

ii) *Consider the same linear program with the additional constraints*

$$q_{ij} \ge 0 \textit{ for all } i, j.$$

If the value of this LP is 1, then since every optimal Q is nonnegative, the conclusions of (i) above hold without the need for a separate copositivity check of Q.

iii) *If the value of the first LP is 0, then no conclusion can be drawn.* □

Now we turn to the crucial issue of finding a symmetric matrix Q for which (6) holds. From the probabilities of the various transitions given in (4), it is easy to compute $E[x^T(\tau_{n+1})Qx(\tau_{n+1}) \mid \mathcal{F}_{\tau_n}]$. Simple calculations show that (6) is equivalent to finding a $Q = Q^T$ such that,

$$\begin{aligned}2\lambda e_1^T Qx(\tau_n) + \lambda e_1^T Qe_1 + 2\sum_{i=1}^{L-1} \mu_i w_i(\tau_n)(e_{i+1} - e_i)^T Qx(\tau_n)\\ +\sum_{i=1}^{L-1} \mu_i w_i(\tau_n)(e_{i+1} - e_i)^T Q(e_{i+1} - e_i)\\ -2\mu_L w_L(\tau_n)e_L^T Qx(\tau_n) + \mu_L w_L(\tau_n)e_L^T Qe_L\\ \leq -\gamma|x(\tau_n)| + c \text{ for some } \gamma > 0 \text{ and } c.\end{aligned}$$

Bounding all the terms not involving $x(\tau_n)$ on the left hand side above by a constant, we see that (6) is equivalent to

$$\begin{aligned}2\lambda e_1^T Qx(\tau_n) + 2\sum_{i=1}^{L-1} \mu_i w_i(\tau_n)(e_{i+1} - e_i)^T Qx(\tau_n)\\ -2\mu_L w_L(\tau_n)e_L^T Qx(\tau_n) \leq -\gamma|x(\tau_n)| \text{ for some } \gamma > 0. \qquad (10)\end{aligned}$$

Now we observe that both the left and right hand sides above are linear in $x(\tau_n)$. Hence (10) (and therefore (6)) holds if the coefficient of every component of $x_j(\tau_n)$ is less than $-\gamma$, whenever $x_j(\tau_n) \geq 1$ (note that $x_j(\tau_n)$ is integral), i.e.,

$$\lambda q_{1j} + \sum_{i=1}^{L} \mu_i(q_{i+1,j} - q_{ij})w_i(\tau_n) \leq -\gamma \text{ for } j = 1, \ldots, L, \text{ if } x_j(\tau_n) \geq 1. \qquad (11)$$

(Above we have taken $q_{L+1,j} := 0$). Thus we seek $q_{ij} = q_{ji}$ such that (11) holds.

However, the above inequality involves the *random variables* $w_i(\tau_n)$. We can exploit the non–idling nature of the policies under consideration, quantified in (5), to bound them.

Let us decompose the sum $\sum_{i=1}^{L} \alpha_{ij}$, above, machine by machine, i.e.,

$$\sum_{i=1}^{L} \alpha_{ij} = \sum_{\{i:\sigma(i)=\sigma(j)\}} \alpha_{ij} + \sum_{\{\sigma:\sigma\neq\sigma(j)\}} \sum_{\{i:\sigma(i)=\sigma\}} \alpha_{ij}.$$

Then, since we are only interested in establishing the inequality when $x_j(\tau_n) \geq 1$, from (5) we have,

$$\sum_{\{i:\sigma(i)=\sigma(j)\}} \mu_i(q_{i+1,j} - q_{ij})w_i(\tau_n) \leq \max_{\{i:\sigma(i)=\sigma(j)\}} \mu_i(q_{i+1,j} - q_{ij}),$$

since the $w_i(\tau_n)$'s at Machine $\sigma(j)$ are nonnegative, and add to 1. At the other machines $\sigma \neq \sigma(j)$, the coefficients may add to 1 or 0, since the other machines

Since the average of the mean number of parts in the system is then bounded, we can say that the system is "stable-in-the-mean."

Above, we have not assumed that the system is Markovian. If the scheduling policy chooses $w_i(\tau_n)$ purely as a function of $x(\tau_n)$, then, in the terminology of Markov Decision Processes, we have a *stationary* scheduling policy, and $\{x(\tau_n)\}$ is a time-inhomogeneous Markov chain. Note also that the resulting Markov chain has a single closed communicating class, since the origin can be reached from every state. Moreover, it is aperiodic since the system can remain at the origin for two or more consecutive time periods.

For Markov chains, stability-in-the-mean implies positive recurrence. The reason for this is that if it were not positive recurrent, then

$$\text{Prob } (|x(\tau_n)| \leq c/\gamma) \to 0 \text{ as } n \to \infty,$$

since the set of such states is compact (see [18]), which however contradicts (9).

In fact, more is true. Let us take $\sqrt{x^T(\tau_n)Qx(\tau_n)}$ as a Lyapunov function. It can then be shown that for some $\epsilon > 0$, and compact set K,

$$E\left[\sqrt{x^T(\tau_{n+1})Qx(\tau_{n+1})} \mid \mathcal{F}_{\tau_n}\right] \leq \sqrt{x^T(\tau_n)Qx(\tau_n)} - \epsilon, \text{ whenever } x(\tau_n) \in K^c,$$

and takes bounded jumps. Then, it can be concluded (see [22]) that the Markov chain has a *geometrically converging exponential moment*, i.e., there exist $\epsilon > 0$, $r > 1$, and $c < +\infty$, so that for any function $f(y) \leq e^{\epsilon|y|}$, and any initial condition $x(\tau_0) = x_0$,

$$\sum_{n=0}^{+\infty} r^n \left| E(f(x(\tau_n))) - \sum_y \pi(y)f(y) \right| \leq ce^{\epsilon|x_0|},$$

where π is the steady-state distribution of the Markov chain. Thus, all polynomial moments $|x|^p$, and even an exponential moment $e^{\epsilon|x|}$ exist, and converge geometrically fast to their steady state values.

The central issue that remains is this. How can we find, if it exists, a symmetric matrix Q which satisfies (8), and for which the Lyapunov function x^TQx has the "negative drift" property (6)?

First, let us consider the condition (8). It is clearly true if $x^TQx \geq 0$ for every x in the positive orthant, since queue lengths are nonnegative. Such a symmetric matrix is said to be *copositive*. Copositive matrices have been extensively studied in linear complementarity theory [5]. They are characterized by each principle submatrix, for which the cofactors of the last row are nonnegative, having a nonnegative determinant [5]. However, testing whether a matrix is copositive is NP-complete [34,1]. Note, though, that every symmetric *nonnegative* matrix is copositive. Hence, to avoid a copositivity check, one could confine attention to nonnegative matrices, though that would lead to a less powerful stability test.

$(x_1(\tau_n), \ldots, x_L(\tau_n))$ is described by

$$x(\tau_{n+1}) = \begin{cases} x(\tau_n) + e_1 & \text{w.p. } \lambda, \\ x(\tau_n) - e_i + e_{i+1} & \text{w.p. } w_i(\tau_n)\mu_i \text{ for } 1 \leq i \leq L-1, \\ x(\tau_n) - e_L & \text{w.p. } w_L(\tau_n)\mu_L, \\ x(\tau_n) & \text{otherwise.} \end{cases} \tag{4}$$

Let us now restrict our attention to scheduling policies which are *non-idling.* These are policies where a machine is not allowed to stay idle if any one of its buffers is non-empty. Thus,

$$\sum_{\{i:\sigma(i)=\sigma\}} w_i(\tau_n) = 1 \text{ whenever } \sum_{\{i:\sigma(i)=\sigma\}} x_i(\tau_n) \geq 1. \tag{5}$$

We will now examine whether all non-idling policies yield a stable system, i.e., if the system is stable for all choices of $w_i(\tau_n)$, subject only to the requirement that they depend only on the past, and satisfy (5).

Consider a quadratic functional, $x^T(\tau_n)Qx(\tau_n)$, where

$$Q = Q^T.$$

Let $\mathcal{F}_{\tau_n}$ be the past σ-algebra. Suppose that we can find a Q such that

$$E[x^T(\tau_{n+1})Qx(\tau_{n+1}) \mid \mathcal{F}_{\tau_n}] \leq x(\tau_n)Qx(\tau_n) - \gamma|x(\tau_n)| + c, \tag{6}$$

where $\gamma > 0$, $|x| := \sum x_i$ = number of parts in the system, and c is some constant. (Note that the conditional expectation above exists, since $x(\tau_n)$ can grow only linearly with n). Then, we can take unconditional expectations to obtain,

$$E(x^T(\tau_{n+1})Qx(\tau_{n+1})) \leq E(x^T(\tau_n)Qx(\tau_n)) - \gamma E|x(\tau_n)| + c.$$

By summing, and dividing by N, we obtain

$$\frac{\gamma}{N}\sum_{n=0}^{N-1} E|x(\tau_n)| \leq \frac{x^T(0)Qx(0)}{N} - \frac{E(x^T(\tau_N)Qx(\tau_N))}{N} + c. \tag{7}$$

Suppose now that Q satisfies the condition,

$$x^T(\tau_N)Qx(\tau_N) \geq 0 \text{ for all } N. \tag{8}$$

Then, (7) yields

$$\frac{\gamma}{N}\sum_{n=0}^{N-1} E|x(\tau_n)| \leq \frac{x^T(0)Qx(0)}{N} + c \text{ for all } N. \tag{9}$$

b_i to b_{i+1} after obtaining service at b_i. After completing service at b_L, they leave the system. Parts at buffer b_i require a processing time which is exponentially distributed with mean $\frac{1}{\mu_i}$. As is usual, we assume that the interarrival times and service times are independent.

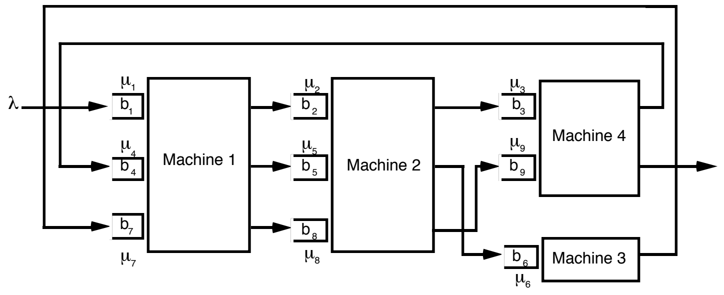


Figure 7: A re–entrant line.

Let $x_i(t)$ denote the number of parts in buffer b_i at time t. Since t is a continuous time variable, this is a continuous time system, which we take to be right continuous with left limits. However we prefer to study a discrete time system, obtained by sampling as follows. If a part in buffer b_i is not being processed by the corresponding machine, we shall suppose that there is a fictitious or virtual part which is being so processed. We sample the system at all arrival times, and all real or virtual service completion times. The rates at which these events occur are λ and μ_i, respectively. We shall now assume (by rescaling time, if necessary) that,

$$\lambda + \sum_{i=1}^{L} \mu_i = 1.$$

Let $\{\tau_n\}$ be the sequence of sampling times, obtained above, with $\tau_0 := 0$. When applied to a continuous time Markov chain, this procedure yields a discrete-time Markov chain whose steady state distribution is the same; it is called "uniformization;" see [28]. (However we have not so far assumed that the scheduling policy is stationary, and so the system may not be a Markov chain).

We shall suppose that the scheduling policy does not change the buffer that it is working on during the time interval $[\tau_n, \tau_{n+1})$. We shall call such scheduling polices as "non-interruptive," and restrict attention to them.

Define the random variable $w_i(\tau_n) := 1$ if the scheduling policy is actually working on a real part in buffer b_i in the time interval $[\tau_n, \tau_{n+1})$, and $:= 0$ otherwise.

If e_i is the i-th coordinate vector, then the discrete time system $x(\tau_n) =$

Theorem 2: Stability of LBFS. *Under the same conditions as in Theorem 1, the LBFS policy is also stable.*

It is worth noting that while the proof of stability of FBFS is simple, the proof of stability of LBFS is quite complex. It relies on a contractive property of the delay experienced by a part, which arrives to find n parts already in the system. In particular, it is shown that the delay of such a part satisfies

$$\text{Delay} \quad \leq \quad (\bar{w} + \epsilon)n + c(\epsilon). \qquad (3)$$

Above $\bar{w} := \max_\sigma \sum_{\{i:b_i \textit{ is served by machine } \sigma\}} \tau_i$ is the maximum work brought by an incoming part to a machine, i.e., a bottleneck machine. Also $c(\epsilon)$ is a constant for every $\epsilon > 0$. The result (3) shows that under LBFS, even re-entrant lines behave like an acyclic "pipeline."

Let us suppose that incoming parts have a due-date stamped on them. If the parts arrive in the order of their due-dates, then the LBFS policy coincides with the well known *Earliest Due Date (EDD)* policy, and so we see that EDD is also stable.

In fact, more is true. For each part, let us define a slack s, by

$$\text{Slack } s \text{ of a part in buffer } b_i \quad := \quad \text{Due date of part } - \zeta_i,$$

where ζ_i is a number associated with buffer i. The *Least Slack* (LS) policy gives priority to a part with the smallest slack. By extending the proof of stability of LBFS, one can establish the stability of all Least Slack policies.

Theorem 3: Stability of All Least Slack policies. *Assume that the arrivals follow the bursty model (2), and the capacity condition $\rho_\sigma < 1$ is satisfied for each station σ. Suppose that due-dates are assigned in such a way that difference between arrival times and due dates is bounded. Then, for* every *choice of the parameters $\{\zeta_i\}$, the corresponding Least Slack policy is stable.*

For all these results, we refer the reader to [30].

6 Stability of Queueing Networks and Scheduling Policies

We will now develop methods for establishing the stability of stochastic queueing networks by simply computing the value of a linear program and checking whether it is 1 or 0. The results in this section and the next are drawn from [22].

For simplicity, only, we focus on re-entrant lines. Consider the system shown in Figure 7. There are L buffers, $b_1, b_2, \ldots, b_L$, and several machines. We shall denote by $\sigma(i)$, the machine serving buffer b_i. We allow $\sigma(i) = \sigma(j)$ even if $i \neq j$. Parts arrive as a Poisson process of rate λ to buffer b_1. They move from the 3α parts in b_3 (recall that b_3 has lower priority than b_2). Since each part in b_3 takes only 0 time units, these 3α parts are immediately sent to b_4. This is the first time that b_4 has any parts, and since b_4 has higher priority over b_1, Machine 1 stops working on b_1, and starts working only on b_4. From $t = 2\alpha^-$ to $t = 4\alpha^-$, Machine 1 works on the 3α parts in b_4. At time $4\alpha^-$, b_4 is empty again, and Machine 1 can turn to b_1. During this time interval, however, 2α new parts have entered b_1. Thus, at time $4\alpha^-$, the new state of the system is $x(4\alpha^-) = (2\alpha, 0, 0, 0)$.

This is a repeat of the situation at time 0^-, except that the number of parts in b_1 has doubled. The cycle of events repeats itself; at time $t = 12\alpha^-$, the state is $x(12\alpha^-) = (4\alpha, 0, 0, 0)$, etc. Thus the number of parts in the system is unbounded, and the system is unstable. (The overall growth rate is linear since parts enter the system linearly in time). □

If τ_i denotes the processing time for parts in b_i, then the *nominal load* on Machine σ is

$$\rho_\sigma := \sum_{\{i: b_i \text{ is served by Machine } \sigma\}} \lambda \tau_i, \tag{1}$$

where λ is the arrival rate. In the above example, $\rho_1 = \rho_2 = 2/3 < 1$, and so the usual capacity condition is satisfied. Yet, as we have seen, the system is unstable as a consequence of the scheduling policy.

It is worth noting that other unstable systems are identified in Kumar and Seidman [24]. In some systems, all clearing policies (also called exhaustive service policies) are unstable. They also show that some systems are unstable even if no part has a re–entrant route. Also, sufficient conditions are provided for stability of some policies.

Motivated by this example, the stability of some buffer priority policies, and the *Least Slack* (LS) policy, is addressed in [30]. The following positive results are established there.

Theorem 1: Stability of FBFS. *Consider the buffer priority policy using the ordering $\{b_1, b_2, \ldots, b_L\}$, where the buffers are ordered in the way they are visited. This is called the* First Buffer First Serve (FBFS) *policy. If $\rho_\sigma < 1$ for each Machine σ, then the system is stable for all arrivals which satisfy the model,*

$$\# \text{ of arrivals in } [s, t] \leq \lambda(t - s) + \delta \text{ for all } 0 \leq s \leq t, \tag{2}$$

for some δ.[2]

[2]Such a model of arrivals is introduced in [6]. The parameter δ allows for some burstiness in arrivals.

examples for different models of systems and other scheduling policies can be found in [24].

Counterexample 1: A system which is unstable under the Longest Processing Time Buffer Priority Policy

Consider the system shown in Figure 6.

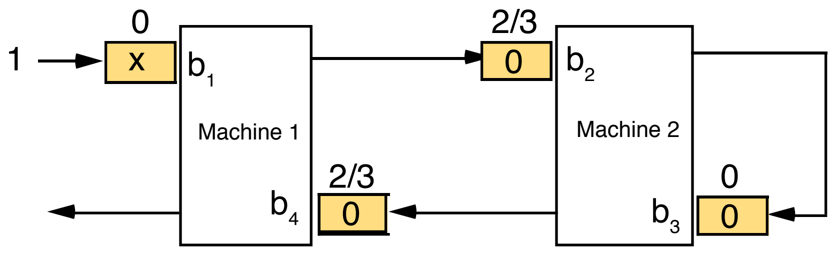


Figure 6: An unstable buffer priority policy.

We shall consider the particular buffer priority policy which uses the ordering $\{b_4, b_2, b_3, b_1\}$ to determine service priority. Thus, at Machine 1, parts in buffer b_1 are processed only when b_4 is empty, and at Machine 2, parts in buffer b_3 are processed only when b_2 is empty. (This policy is different from the earlier LBFS policy, which uses the different ordering $\{b_4, b_3, b_2, b_1\}$).

Let us suppose that parts arrive periodically to the system, with interarrival time of one unit. Parts in b_1 require 0 time units for processing. (This does not however mean that parts can skip b_1 and proceed directly to b_2. They may have to wait for b_4 to become empty, before Machine 1 can process them. There is a difference between 0 and nothing!). Parts in buffers b_2, b_3 and b_4 require 2/3, 0 and 2/3 time units each of processing time, respectively.

Note that the buffer priority policy $\{b_4, b_2, b_3, b_1\}$ corresponds to giving priority to those parts which have the *longest* processing time, and hence is called the *Longest Processing Time (LPT)* buffer priority policy.

We will now show that the system is unstable for some initial conditions. Consider the initial condition $x(0^-) = (\alpha, 0, 0, 0)$ where the i-th component of $x(t)$, denoted $x_i(t)$, represents the number of parts in buffer b_i at time t.

Since b_4 is empty at $t = 0^-$, Machine 1 immediately processes all the α parts in b_1 (since they each take only 0 seconds of processing time), and sends them to b_2. Thus, Machine 2 starts working on b_2. Meanwhile, new parts arrive at b_1, one every time unit, and are immediately sent on to b_2 (since b_4 is still empty). At time $t = 2\alpha^-$, b_2 is empty for the first time. The reason is that a total of $(\alpha + 2\alpha) = 3\alpha$ parts have been sent to b_2 from b_1, and these parts take a total of 2α time units for processing (since each part at b_2 takes 2/3 of a time unit). Thus, at time $t = 2\alpha^-$, Machine 2 is finally able to begin working on

Thus, from a myopic point of view it is better to work on parts in buffer b_6 rather than parts in b_3. Extending this reasoning to the other machines too, it is myopically better to prefer b_5 to b_2, and b_4 to b_1. This suggests ordering the buffers as $\{b_6, b_5, \ldots, b_1\}$, (i.e., in the reverse of the order they are visited), and giving preference according to this ordering. This is called the *Last Buffer First Serve (LBFS) Policy.* It is an example of a *buffer priority policy.*

However, such a myopic policy is not optimal in the long run. In the example above, it may be that Machine 1 is idle since buffers b_1 and b_4 are both empty. In that case, it may be better for Machine 3 to work on the part in b_3, and thereby provide work for Machine 1 to do. Such "starvation avoidance" of machines, especially bottleneck machines, may yield dividends in the long run. While one may qualitatively discern such properties, the determination of an optimal policy, by a methodology such as dynamic programming, which balances both short and long term objectives, is intractable. In the face of such difficulties, several heuristics have been proposed for scheduling; see for example Panwalker and Iskander [36].

5 Possible Instability and Stability under Bursty Arrivals

It is clearly of interest to be able to precisely evaluate the performance of a proposed scheduling policy, such as, for example, the LBFS policy. However, available results in queueing networks are not applicable.

Through the work of Jackson [15,16], Baskett, Chandy, Muntz and Palacios [2], and Kelly [19], explicit solutions have been determined for the steady state distributions of certain queueing networks. Such networks are broadly refereed to as "product form" queueing networks, due to the multiplicative form of their steady state probability distribution. Unfortunately, if the service time distributions for the parts in the buffers at the same machine are different (there are some minor exceptions, unimportant here), or if the scheduling policy gives priority according to the buffers, then the resulting networks generally fall outside the above class.

The types of networks of interest to us possess both these features, and very little is known concerning their behavior. For a deterministic system, we shall say that it is stable, if all its buffer levels are bounded. In this paper if a stochastic system is such that the underlying Markov chain has a steady-state, i.e., it is positive recurrent, we shall say that is is *stable*. It is not even known whether such networks have a steady state. Clearly, it is necessary to first resolve whether a network is stable before one can address other more detailed questions related to steady state performance.

We will now show that even simple networks can be unstable, even when the usual capacity condition, that the nominal load brought to each server is within the server's capacity, is met. This example is drawn from [30]. Other similar

Thus, one wishes to tailor the design of scheduling polices to accommodate the special features of re-entrant lines, and take advantage of whatever structure they possess. (In particular, ideas developed in other environments, e.g., kanbans, should not be blindly used).

4 Beginning the Design of Scheduling Policies: Little's Theorem and the Myopic LBFS Policy

Consider a stable system. Let

λ	:=	average arrival rate of parts to the system,
L	:=	average number of parts in the system,
W	:=	average time spent by parts in the system.

Then, under mild stability assumptions, Little's Theorem says that

$$L = \lambda W.$$

Thus, for a fixed plant throughput λ, one can reduce the mean cycle time W, by just reducing the mean number of parts L in the system.

This provides a convenient starting point for an initial design of a scheduling policy. Consider the re-entrant line shown in Figure 5.

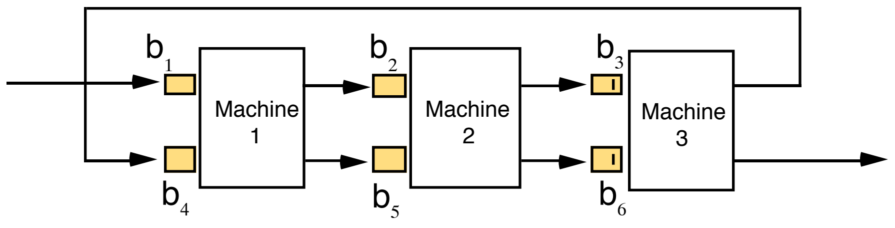


Figure 5: The myopic LBFS policy for a re–entrant line.

It consists of 3 machines and 6 buffers. Parts visit the buffers in the order b_1, $b_2, \ldots, b_6$.

Consider Machine 3. Suppose there are parts in each of the buffers b_3 and b_6, and Machine 3 has to decide which part to process next. If it works on a part in buffer b_6, then it can immediately reduce the number of parts in the system – since parts in b_6 immediately leave the system. However, working on a part in b_3 simply sends it to b_4, without changing the total number of parts in the system.

3 The Traditional Dichotomy in Manufacturing Systems: Flow Shops and Job Shops

Manufacturing systems usually fall into two categories, flow shops and job shops (though, of course, there are other types of plants, e.g., for ship building). Good examples of flow shops are automobile assembly lines; see Figure 3.

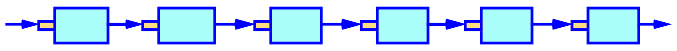


Figure 3: A flow shop.

The key features are that the flow of parts is acyclic (i.e., a part never revisits a machine or worker[1]), and the line is dedicated to a single type of part, which is manufactured in high volume. The second category into which many manufacturing systems fall is job shops. Good examples are metal cutting shops which consist of a number of "resources", e.g., lathes, milling machines, etc. A job shop can accept custom orders, and each order may require a specific sequence of operations, i.e., a route. A caricature is shown in Figure 4,

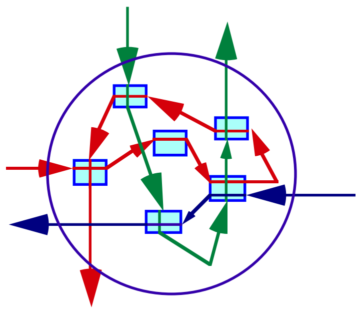


Figure 4: A job shop.

where several "routes" are shown. There may not be much of a pattern in the different routes; they may look "random." Each route is also typically of low volume.

With semiconductor manufacturing plants, we have a new type of system, – a re-entrant line. It combines features of both flow shops and job shops. For example, unlike flow shops, the route is re-entrant; however, unlike job shops, it may be of high volume. Thus, it features the additional complication of competition for machines by parts, not present in flow shops. However, there is more structure to re-entrant lines, than is usually present in job shops.

[1]It may do so for quality control purposes.

of the machines for the performance of a particular processing step at several layers. The resulting manufacturing system is shown in Figure 2. It shows the route of a single product (called process flow or recipe) in an aggregated model of a full scale production line. There are a large number (60) of stages of processing. Many of the machines are revisited several times; for example, Station 1 is visited 14 times by each wafer. This type of a manufacturing system is accordingly called a "re-entrant line" [21]. Wafers usually travel in groups of about 20 wafers, called a *lot*, which we shall simply refer to hereafter as a single "part."

The key feature of a re-entrant line is that several parts at different stages of production may be in competition for the same machine (or set of machines at a station). These parts (like cars at a traffic light) may have to wait for machines. This can lead to large waiting times, an unfortunate characteristic of semiconductor manufacturing plants. For example, while each part may require a total service time from all machines of only about 10 days, a part may nevertheless take 60 days to complete production. The time to complete production, usually referred to as *sojourn time* in queueing networks, is called the *manufacturing lead time* or *cycle-time*. The ratio of mean cycle-time to mean total processing time can be large, between 2 and 10 (it is 6 in the example above); it is called the "cycle-time multiplier."

A key concern in semiconductor manufacturing plants is to reduce the mean cycle-time. It clearly has several economic benefits. For example, when developing new products it allows faster prototyping, and quicker responsiveness to customer needs. This is important in a rapidly changing technological environment where product life cycles are becoming very short, sometimes just a year. By Little's Theorem, a smaller mean cycle-time also implies a smaller mean number of parts in process, called work-in process or WIP. Reducing the WIP reduces the capital tied up, as well as the clutter in the plant. Moreover, when products become obsolescent, so can the WIP. There is also a technological benefit to reducing cycle-times. The smaller the time that a wafer is exposed to contaminants in the plant, the greater is the yield of good wafers, important in an industry where contamination and yield are major concerns.

It is also important to reduce the variance of the cycle-times. A smaller variation in cycle-times allows managers to better predict when products will be completed, and thus meet their due-dates. Thus, a small variance of cycle-times allows for improved coordination of the plant's output to the downstream operations such as assembly, etc.

In Section 14 we shall design new scheduling policies to reduce the mean and variance of cycle-times. We also report briefly from an extensive comparative study of their performance on plant models.

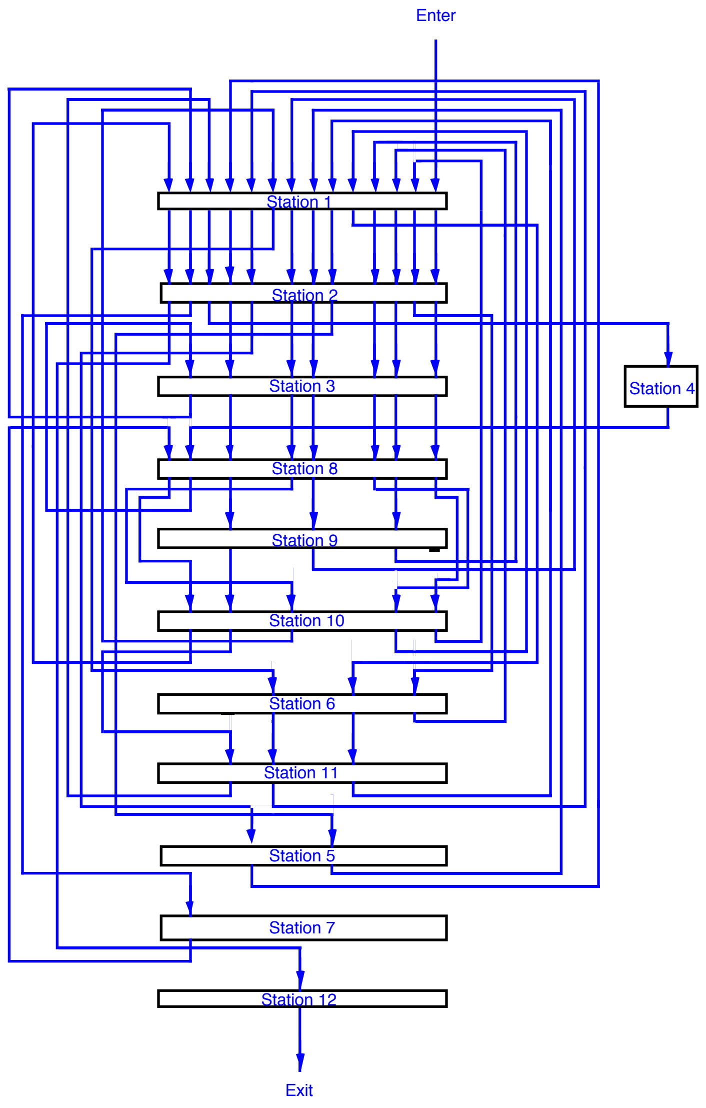


Figure 2: An aggregated model of a semiconductor manufacturing plant.

As shown in Figure 1, the processing of a wafer is layer by layer, and the final wafer can be regarded as a multilayered sandwich. Each layer, in turn, requires many steps of processing (deposition, etching, etc.), and many layers repeat many of the same processing steps.

A machine (or station containing a set of identical machines) can perform one particular processing operation. Thus, wafers return several times to many tion some previous results establishing the stability of certain buffer priority policies. In Section 6, we begin by providing a model of a re-entrant line, on which we focus (for brevity only) in the rest of the paper. Then in Sections 6 and 7, we develop the linear programming approach to establishing stability. In Section 8, we provide some instability/non–robustness tests. In Section 9, we turn to the problem of performance evaluation, and show how one can obtain linear equality and inequality constraints on the performance of a system. In Section 10, we show how one can obtain more linear equalities and inequalities for specific modules which are prevalent in many applications. We also provide several examples showing the power of the approach. In Section 11, we show how one can obtain bounds on transient performance. In Section 12, we establish a duality relationship between the performance and stability problems, and examine some of its consequences in Section 13. In Section 14, we turn to the problem of scheduling semiconductor manufacturing plants. We develop the new class of fluctuation smoothing policies for reducing the variance of lateness and variance of cycle-time, and briefly present the results of a simulation study. Then, we develop a hypothesis for the particular behavior observed in the simulations, and, exploiting this, we design the new fluctuation smoothing policy for reducing the mean cycle-time. We report on its simulation performance, and also present another interpretation of it as alleviating total downstream starvation. Finally, in Section 15, we end on a broad note.

2 Semiconductor Manufacturing

A semiconductor manufacturing plant (also called fabrication line or fab) takes raw wafers (about 8" or diameter) and imprints several layers of chemical patterns on them.

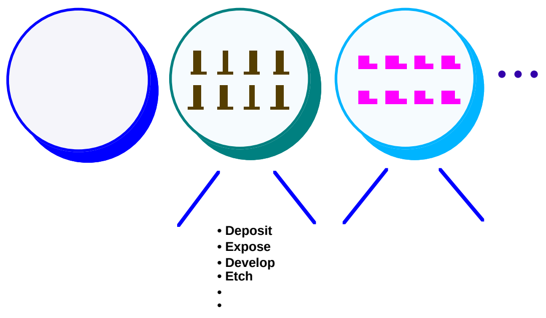


Figure 1: The semiconductor manufacturing process.

properly scheduling the system, i.e., deciding which lot of wafers is served next, one can decrease the mean manufacturing lead time (called mean "cycle-time" in the industry). These systems are very large scale, involving about ten to twenty service stations, and a hundred or more stages of processing (i.e., buffers) for each wafer. One therefore wants to design scheduling policies that are easily implemented on large scale systems, whose decisions can be determined in real-time, and which can cope with random events such as machine failures.

We develop a new set of scheduling policies called "fluctuation smoothing" policies. First, we develop a policy to reduce the variance of lateness of parts. Then, we show how to modify this policy so that it reduces the variance of the cycle-time. Then we report briefly from the results of an extensive comparative simulation study, which shows that our Fluctuation Smoothing Policy for Reducing the Variance of the Cycle-Time (FSVCT), indeed reduces the standard deviation of the cycle-time, by a substantial amount. Surprisingly, as a by product we discover that FSVCT also reduces substantially the *mean* cycle-time, when the arrivals to the plant are periodic (called "deterministic"). In developing a hypothesis for why this is so, we develop yet another Fluctuation Smoothing Policy for Reducing the Mean Cycle-Time (FSMCT), for systems whose arrivals are not periodic. This policy has been tested extensively, and we briefly report from a study showing that it achieves its purpose. The net result is a new scheduling policy that reduces the mean waiting time by about 20%, and the standard deviation of cycle-time by about 50%, in comparison to the well known First Come First Serve Policy (FCFS or FIFO). These results have been subjected to statistical tests.

The main purpose of the FSMCT policy is to smooth the burstiness of all flows into every buffer in the system, simultaneously. That this can be done by an intuitively rational design is surprising.

There is yet another interpretation of FSMCT as a policy that at each time keeps track of the total downstream shortfall of parts from every buffer (short fall := actual number of parts downstream − mean number of parts downstream), and gives priority to the buffer whose shortfall is greatest. This is an appealing interpretation, which also shows that the scheduling policy is "stationary," in the terminology of Markov Decision Processes.

The rest of this paper is organized as follows. In Section 2, we start with a motivating problem: how to schedule semiconductor manufacturing plants. We describe the features of semiconductor manufacturing which lead to a re-entrant line. We provide an example of a model plant. In Section 3, we place re-entrant lines in perspective by showing their relationship vis-a-vis the traditional dichotomy in manufacturing systems: flow shops and job shops. Next, in Section 4, using Little's Theorem as a starting point, we design a myopic scheduling policy – the Last Buffer First Serve (LBFS) policy. Then, in Section 5, we motivate the problem of stability of queueing networks and scheduling policies by providing an example of a re-entrant line which is unstable under the Longest Processing Time (LPT) buffer priority policy. We also briefly men-

1 Introduction

In this paper, we provide a brief survey of some recent results concerning queueing networks. We shall address three sets of issues: stability, performance analysis, and the design of scheduling policies. The results are mainly drawn from [30,27,22,23,32], to which we refer the reader for further details.

After a brief description of a motivating application, we begin by exhibiting a counterexample that shows that even simple queueing networks can be unstable for some scheduling policies. This motivates the problem of how to establish the stability of queueing networks and scheduling policies. We exhibit a new approach based on using linear programming. We show how to construct a linear program (LP) such that every feasible solution is a weighting matrix whose associated quadratic form serves as a stochastic Lyapunov function. Thus, we obtain a simple linear programming procedure for establishing stability. This approach establishes not just the positive recurrence of the underlying Markov chain, but, in fact, a strong form of geometric ergodicity. We mention some extensions of this approach and illustrate its use on several examples.

Next, we turn to the problem of performance evaluation of queueing networks and scheduling policies. Consider a queueing network that has a steady state in which the mean number of customers is finite. (This is the sort of stability issue that is studied in the earlier first part of the paper.) We show that whenever such stability holds, the mean values of certain random variables determining the performance, satisfy a set of linear equality constraints. In addition to satisfying non-negativity constraints, these mean values also satisfy a set of "non-idling" inequality constraints if the scheduling policy under consideration is non-idling. Further, many systems satisfy additional linear equalities or inequalities, e.g., closed systems, networks under buffer priority policies, etc. Given these linear constraints, the performance of a system can be bounded by solving a linear program. Thus we obtain upper and lower bounds on mean queue lengths, by linear programming. We also exhibit the application of this methodology on a variety of examples, showing the power of the approach.

We also show that one can obtain linear programs to bound the transient (as opposed to steady-state) performance of queueing networks and scheduling policies.

Given two separate linear programs, one for establishing stability, and another for studying performance, one may well wonder what the connection is between these two linear programs. We show that these two linear programs are simply the duals of each other. This close connection allows us to develop a variety of the results relating the boundedness of the performance linear program to the existence of a quadratic stochastic Lyapunov function.

The third issue we address is that of designing good scheduling policies for queueing networks. We focus on the important and topical problem of scheduling semiconductor manufacturing plants. Here, wafers at several stages of their life may be in competition for the same set of machines. Thus, potentially, by

SCHEDULING QUEUEING NETWORKS: STABILITY, PERFORMANCE ANALYSIS AND DESIGN*

P. R. Kumar
University of Illinois
Department of Electrical and Computer Engineering
and the Coordinated Science Lab
1308 West Main Street
Urbana, IL 61801, USA

October 10, 1994

Abstract

Queueing networks are a useful class of models in many application domains, e.g., manufacturing systems, communication networks, and computer systems. Control is typically exercised over such systems by the use of scheduling policies.

However, if one ventures outside a certain special class of systems for which the steady state distribution has a product form, very little is known concerning their performance or even stability. In the first half of this survey paper, we present new theoretical developments on stability analysis and performance evaluation for queueing networks and scheduling policies. We show how one may solve problems in stability and performance analysis by solving linear programs.

In the second half, we address the problem of scheduling a class of queueing networks called re-entrant lines, which model semiconductor manufacturing plants. We propose a new class of scheduling policies based on smoothing all the flows in the system. We also report briefly from an extensive simulation study comparing the performance of the suggested scheduling policies with a range of other scheduling policies, for a variety of semiconductor manufacturing plant models.

*The research reported here has been supported in part by the U.S. Army Research Office under Contract No. DAAL-03-91-G-0182, by the National Science Foundation under Grant No. ECS-92-16487, and by the Joint Services Electronics Program under Contract No. N00014-90-J1270.