Enhanced Dynamic Programming Algorithms for Series Line Optimization

Michael H. Veatch* May 2005

Abstract

Dynamic programming value iteration is made more efficient on a five-machine unreliable series line by characterizing the transient and "insensitive" states. Holding costs are minimized subject to a service level constraint in a make-to-stock system with backordering. State space truncations are chosen by checking the recurrent class in previous runs. An approximate model is developed that reduces the number of machine states. Monotone control theory is used to restrict the search for a control switching surface. Numerical optimal policies are compared with the heuristic control point policy and several characteristics of optimal policies are identified.

Index Terms

Make-to-stock, production line, dynamic programming, control point policy.

I. Introduction

Queueing network control problems are so computationally intense that dynamic programming (DP) has mostly been used for systems with only two or three buffers. The curse of dimensionality and typically large truncations lead to very large state spaces. Also, implementing an optimal policy requires look-up from a table, the size of which is again exponential in the number of buffers. Nevertheless, numerical optimization has played a significant role in the study of these networks, providing insights about the structure of optimal policies in demand driven systems [1],[2],[3] and reentrant lines [4]. These insights can complement analytic methods, such as monotone control, stochastic comparison, and asymptotic analysis. Optimization is also valuable

*Department of Mathematics, Gordon College, Wenham, MA 01984, (978) 867-4375, veatch@gordon.edu

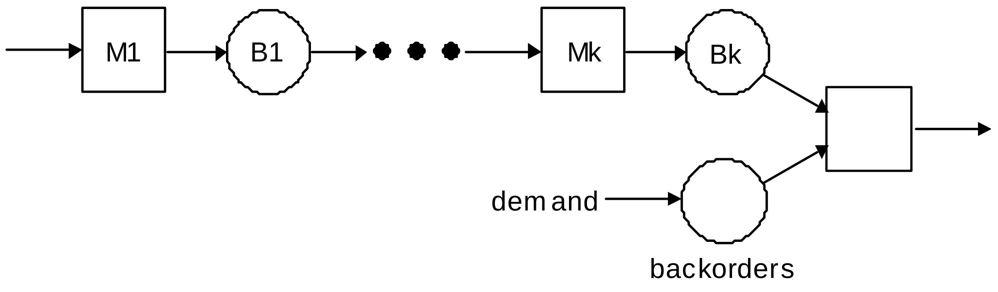

Fig. 1. A make-to-stock series line.

for assessing the performance of heuristics. Although performance bounds are available [5],[6] their tightness is not very satisfying, particularly in balanced heavy traffic.

This paper enhances the DP value iteration algorithm using detailed understanding of a particular network. A series make-to-stock line with five unreliable machines is optimized. The model has 2^5 machine states as well as five buffers. The objective is to minimize long-run average holding cost subject to a service level constraint (see [7] for motivation). Service, failure, repair, and demand interarrival times are exponentially distributed. Several speedups are developed for this network, including a method for choosing state space truncations after one run (as opposed to doing sensitivity analysis on each buffer size), an approximate model that eliminates low probability machine states, and using the monotone form of the control to reduce the number of minimizations. For comparison, using more computing resources but standard value iteration, a six buffer problem has been solved in moderate (but not heavy) traffic [8] and a six buffer problem with input control has been solved using a similar truncation method [9]. Numerical results lead to several observations about the structure of optimal policies. A heuristic called the control point policy [10] is also tested.

II. The make-to-stock line

Consider a production system with k machines in series. Parts are processed by machine 1, enter buffer 1, are processed by machine 2, and so on (Figure 1). The final buffer contains finished goods and is used to satisfy exogenous demand. If it is empty, the demand is backordered. Raw material is always available to machine 1. Denote by $x_i(t)$ the number of parts in buffer i, including any part being processed at the $(i+1)$th machine, at time t. Note that $x_i \in \mathbf{Z}^+$,

$i = 1, \ldots, k-1$ but $x_k \in \mathbf{Z}$, with $x_k < 0$ representing backorders. Machines are subject to operation-dependent failures, that is, they can fail only when processing a part. Write $\alpha_i(t) = 1$ if machine i is operational at time t and 0 if failed. The processing time, operating time until failure, and repair time of machine i are independently exponentially distributed with rates μ_i, p_i, and r_i, respectively. Demand interarrival times are also exponentially distributed with rate λ. Stability of the system requires that $\lambda < \mu_i r_i / (r_i + p_i)$ for all i.

The decision is whether or not each machine processes a part. Processing can be preempted. Let $u_i(t) = 1$ if machine i is processing at time t and 0 if it is not. Admissible controls are nonanticipating and have $u_i(t) \le \alpha_i(t) x_{i-1}(t)$, $i > 1$, and $u_1(t) \le \alpha_1(t)$, i.e., machines cannot process when starved or failed. Holding cost is incurred at the rate $c(x) = \sum_{i=1}^{k-1} c_i x_i + c_k x_k^+$ in state x, with $c_i > 0$ for all i. It will be sufficient to consider only stationary Markov policies, written as $u(\alpha; x) = (u_1(\alpha; x), \ldots, u_k(\alpha; x))$, which are also unichain. Hence, there is a unique stationary distribution and long-run averages do not depend on the initial state. We wish to minimize long-run average holding ("inventory") cost

$$J_I = \limsup_{T \to \infty} \frac{1}{T} E_{\alpha,x} \int_0^T c(x(t))dt \tag{1}$$

subject to the service level constraint $\beta \equiv P\{x_k > 0\} \ge \beta_0$ for some $\beta_0 > 0$. Here $E_{\alpha,x}$ denotes expectation given the initial state $\alpha(0) = \alpha$, $x(0) = x$ and policy u and $P\{\cdot\}$ is with respect to the stationary distribution.

Instead of solving the constrained MDP, introduce the Lagrangian problem as the unconstrained MDP that minimizes $J(b)$, defined as in (1) but with $\bar{c}(x) = c(x) + b 1_{\{x_k \le 0\}}$ replacing $c(x)$. The constant b can be interpreted as the penalty per unit time for stockouts. It is shown in [11] that, for a fixed b, if $J^*(b)$ is optimal for the Lagrangian problem and the associated $\beta^*(b) = \beta_0$, then $J_I^* = J^*(b) - \beta^*(b)b$, i.e., the cost (1) associated with this policy is optimal for the constrained problem. Viewed as a function of b, $J^*(b) - \beta^*(b)b$ is nondecreasing and piece-wise constant. Varying b generates points $(\beta_0,\ J_I^*)$. Between these points, the optimal policy for the constrained problem requires randomization in one state [12, theorem 4.4]. These discontinuities in J_I^* are small, so randomization will be neglected.

The process is uniformized with the potential event rate $\Lambda = \lambda + \sum(\mu_i + p_i)$. Letting e_i denote the ith standard basis vector and $e_0 = 0$, define the DP operator

$$\mathbf{T}h(\alpha;x) = \frac{1}{\Lambda}[\bar{c}(x) + \lambda h(\alpha;x-e_k) + \sum_{i=1}^{k}((1-\alpha_i)\,[r_i h(\alpha+e_i;x) + (\mu_i+p_i-r_i)h(\alpha;x)]$$
$$+ \alpha_i \min\{(\mu_i+p_i)h(\alpha;x),\ \mu_i h(\alpha;x-e_{i-1}+e_i) + p_i h(\alpha-e_i;x)\})] \tag{2}$$

for all states $(\alpha;x)$. Boundaries are enforced by replacing $x - e_{i-1} + e_i$ with x when $x_{i-1} = 0$. The standard value iteration (VI) algorithm is $h_0(\alpha;x) = 0$ and

$$J_{n+1} = \Lambda(\mathbf{T}h_n)(1;e_k);\ \ h_{n+1}(\alpha;x) = (\mathbf{T}h_n)(\alpha;x) - J_{n+1}/\Lambda \tag{3}$$

where $(\alpha;x) = (1;e_k)$ has been chosen as the reference state. It converges, $h_n \to h$, and the optimal policy can be recovered from the differential cost function h.

The service level $\beta^*(b)$ is computed simultaneously with VI. Define the one-step operator $\mathbf{T}_n^{\beta}$ as in (2) but replace $\bar{c}(x)$ with $1_{\{x_k>0\}}$ and use the policy that achieves the min in (2) at each iteration, i.e., use the first (second) term in the min if $u_i = 0$ (1). Along with (3) compute $h_0^{\beta}(\alpha;x) = 0$ and

$$\beta_{n+1} = \Lambda(\mathbf{T}_n^{\beta}h_n^{\beta})(1;e_k);\ \ h_{n+1}^{\beta}(\alpha;x) = (\mathbf{T}_n^{\beta}h_n^{\beta})(\alpha;x) - \beta_{n+1}/\Lambda. \tag{4}$$

Since u_n converges to the optimal policy for the Lagrangian problem, $\beta_n \to \beta^*(b)$.

III. Speed-up strategies

First we describe two general modifications to the VI algorithm.

Flexible VI. Select an ordering of the states and compute h_{n+1} in that order, starting with the reference state. Replace h_n with h_{n+1} in (3) at states where it has already been computed, making use of the updated approximation $h_{n+1}(x)$ sooner. We are not aware of a proof of convergence for this algorithm, but it performs well in numerical tests [13].

Initialization. Since it is necessary to solve the Lagrangian problem repeatedly with different values of b, the final h and h^{β} from one run can be used as h_0 and h_0^{β} for the next run.

Next we describe specific enhancements and approximations for this problem.

State space truncation. The state space is truncated to $x_i \le U_i$, $i = 1, \ldots, k$ and $x_k \ge L$. Searching for the smallest accurate truncation can require many runs. We propose a faster method. For the optimal policy, the recurrent class is bounded above. Define

$$N_i = \max_{\alpha,x}\{x_i : (\alpha,x) \text{ is recurrent}\}. \tag{5}$$

A method to compute N_i from the optimal policy is given in Appendix A. If $N_i \leq U_i - 1$ for all i, then the upper truncation is sufficiently large (increasing N_i will not change the results) for this L. Numerical experience suggests that N_i is fairly insensitive to N_j, $j \neq i$, and nonincreasing in L, suggesting that $U_i = N_i$ be used for future runs with larger L. If it is impractical to find a sufficiently large truncation for some buffers, one must revert to sensitivity analysis to justify these U_i. However, if $N_k = U_k$, we increase U_k, since results are very sensitive to this truncation.

The lower truncation on x_k can never be exact. The accuracy of similar runs can be improved by applying a penalty on the boundary from a run with smaller L. Let

$$\Delta h(\alpha; x_1, \ldots, x_{k-1}) = h(\alpha; x_1, \ldots, x_{k-1}, L-1) - h(\alpha; x_1, \ldots, x_{k-1}, L) \tag{6}$$

where L is the tighter truncation to be used in the current run and h is the final h_n from a previous run. At the lower boundary $x_k = L$, replace $h(\alpha; x - e_k)$ with $h(\alpha; x) + \Delta h(\alpha; x_1, \ldots, x_{k-1})$ in (2). The analogous penalty is used in h^β.

Reduced model of machine failures. If the machines are fairly reliable, then states in which many machines are down have low probability. These states will be omitted by turning off failures when D machines are down. To improve the accuracy of the approximation, failure rates are adjusted. Denoting the number of machines down by $d(t) = k - |\alpha(t)|$ and the adjusted model by a tilda, we find failure rates $\widetilde{p_i}$ such that, in a model with time-dependent failure rates,

$$E[\widetilde{d}] = E[d]. \tag{7}$$

Formulas for $\widetilde{p_i}$ are derived in Appendix B for $D = 1$ and 2 in the homogeneous machine case, where $(\lambda/\mu_i)p_i$ and r_i are constant over i. The reduced model replaces (2) with

$$\mathbf{T}^D h(\alpha; x) = \frac{1}{\Lambda}[\bar{c}(x) + \lambda h(\alpha; x - e_k) + \sum_{i=1}^{k}((1-\alpha_i)\left[r_i h(\alpha + e_i; x) + (\mu_i + p_i - r_i) h(\alpha; x)\right]$$
$$+ \alpha_i(\mu_i + p_i) \min\{h(\alpha; x),\ h(\alpha; x - e_{i-1} + e_i)\})] \tag{8}$$

for $k - |\alpha| = D$ and $\mathbf{T}^D = \mathbf{T}$ for $k - |\alpha| < D$. Similar modifications are made to T_u^β.

Search for switching surface. The optimal policy consists of regions of constant control in the inventory space; their boundaries are called control switching surfaces. This structure of the optimal control can be enforced at every iteration to reduce the number of minimizations. Holding other variables constant and decreasing x_k, the control u_k switches from idle to processing. Once this switching threshold is located, the minimization for machine k need not be performed in states with smaller x_k. In the examples the machine k switching surface is a nondecreasing and nearly constant function of x_i, $i < k$ (see Appendix A), allowing most of the rest of the minimizations to be avoided. This method is not applied to other controls because a significant overhead is incurred in keeping track of the multiple switching surfaces.

The algorithm with these changes is summarized below.

Initialization

1. Choose a limit D on the number of machines down and compute the adjusted failure rates using (17) if $D = 1$ or (16), (18), (12), and (15) if $D = 2$.
2. Choose a truncation $x_i \le U_i$, $i = 1, \dots, k$ and $x_k \ge L$ and an ordering of the states.
3. Choose a stockout penalty b.
4. Set $h_0(\alpha; x) = h_0^\beta(\alpha; x) = 0$ or use the final h_{n+1} and h_{n+1}^β from a previous run. Set $\Delta h(\alpha; x_1, \dots, x_{k-1}) = 0$ or use (6) from a run with smaller L.

Value iteration

Compute h_{n+1} and J_{n+1} from (3), using the ordering of the states and replacing h_n with h_{n+1} as it is computed at each state. Use (2) for the full model or (8) for the reduced model. Compute h_{n+1}^β and J_{n+1}^β from (4), possibly modified for the reduced model. Enforce the truncations and $x_i = 0$ boundaries and include the penalty at $x_k = L$.

$$u_{n+1}(\alpha; x) = \begin{cases} 0 & \text{if the first term in (2) or (8) achieves the min} \\ 1 & \text{otherwise} \end{cases}$$

Output average cost $J_I^* = J_{n+1} - \beta_{n+1} b$ and service level $\beta^* = \beta_{n+1}$.

To check the truncation, compute the upper bounds N_i using Appendix A. If necessary, adjust U_i and L as described above. Once the truncation is adequate, Δh can be set using (6) and L increased somewhat for similar runs. If the service level constraint is not tight, $\beta^* \neq \beta_0$, then adjust b (search using the fact that $\beta^*(b)$ is nondecreasing).

IV. Numerical results

The two- and five-machine examples listed in Table I were optimized. For cases 1 to 3, the control point policy (CPP) is also evaluated. A CPP [10] is defined by

$$u_i(\alpha; x) = \begin{cases} 1 & \text{if not starved, } x_i < C_i \text{, and } \sum_{j=i}^{k} x_j < Z_i \\ 0 & \text{otherwise.} \end{cases}$$

Case	λ	μ	p_i	r_i	c
1	0.64	1, 1	0.05	0.2	1, 1.5
2	0.64	1.5, 1	0.02	0.1	1, 2
3	0.4	1, 1, 1, 1, 1	0.2	0.5	1, 1.4, 1.6, 1.8, 2
4	0.4	2, 1.75, 1.5, 1.25, 1	0.2	0.5	0.125, 0.25, 0.5, 1, 2
5	0.4	2, 1.75, 1.5, 1.25, 1	0.02	0.1	1, 1.4, 1.6, 1.8, 2

TABLE I

PARAMETERS OF THE CASES

	Optimal		Best CPP	
Case	Cost	Service Level	% Suboptimal	C_1, Z_1, Z_2
1	22.37	0.7992	0.4	22, 23, 21
2	17.52	0.7388	1.5	8, 14, 14

TABLE II

POLICY COMPARISON FOR TWO-MACHINE CASES

It uses two production limits at each machine except k; set $C_k = \infty$. The parameter C_i is the capacity of buffer i and Z_i is the *base stock* level of machine i, expressed in terms of the echelon inventory $\sum_{j=i}^{k} x_j$; machine i produces at most Z_i units in advance of demand. It is assumed that $C_i \geq z_i \equiv Z_i - Z_{i+1}$, $i = 1, \ldots, k-1$ and $z_k \equiv Z_k$, so that in the absence of demand the system will move to the *hedging point* $x = z$ and idle.

For the two-machine cases, the best CPP is found by exhaustive search (Table II). For case 1 the CPP uses a large buffer limit and performs well because the optimal policy is nearly base stock form (no buffer limits), consistent with [2],[1]. In case 2 the line is not balanced, so the optimal policy is further from base stock: the CPP has a smaller buffer limit and does not perform quite as well.

For case 3, the best CPP is not known. Instead, CPP1 in Table III was proposed in [14] (their case 1). CPP2 and 3 are modifications of CPP1 suggested by the optimal policy. Their suboptimality is in comparison to optimal policies (not shown) that achieve nearly the same service level. CPP2 shows the importance of holding less inventory at the upstream buffers. CPP3 demonstrates that finite buffer capacities are unimportant: when C_i is increased suboptimality

Policy	Cost (% Suboptimal)	Service Level	C	Hedging Point
Optimal	25.46	0.8914	–	0, 1, 2, 3, 11
CPP1	(18%)	0.8911	7, 7, 7, 7	3, 3, 3, 3, 8
CPP2	(4%)	0.8837	7, 7, 7, 7	1, 1, 2, 3, 10
CPP3	(0.8%)	0.9004	7, 10, 10, 10	1, 1, 2, 3, 10

TABLE III

POLICY COMPARISON FOR CASE 3

decreases.

The optimal policy for case 3 was computed using the truncation $U = (7, 10, 10, 10, 15)$ and $L = 20$, i.e., $-20 \leq x_5 \leq 15$. The upper truncation for x_5 is exact; the maximum of x_5 under the optimal policy is $N_5 = 11$, which is also the hedging point. The other upper truncations are not exact. They were set by testing sensitivity because the method of Section 3 did not yield $N_i < U_i$. The full model contains over 12 million states. Using just the initialization and monotone switching speedups, running 5000 iterations took 7 hours on a 1.5 GHz UNIX workstation and found cost within roughly $\pm 10^{-3}$.

Speedup from each successive enhancement was tested on case 5 using the truncation $U = (3, 3, 3, 3, 12)$ and $L = 20$ (Table IV). The usual stopping criterion is too conservative to be useful in measuring convergence. Instead, the (hundreds of) iterations needed for J_I to be within 0.1% of its final value, computed from a longer run, was measured. For flexible VI, states are ordered increasing in (α, x), i.e., x_5 is the inner loop. The ordering with x_5 decreasing required about 100 more iterations. When h is initialized from a previous run, convergence would be immediate if the parameters were unchanged. The stockout penalty b was changed from 60 to 58, typical of the change made when searching for β_0. Starting with the lower bound penalty, L is reduced to 15, still achieving more accuracy (within 0.2% of a run with $L = 40$) than the $L = 20$ runs. The overall speedup is dramatic. Monotone switching reduced the number of machine 5 minimizations by 97%. However, because of the overhead it requires, run time was essentially unchanged. Slightly fewer iterations were required because a different state ordering is used.

The reduced machine state model is evaluated in Table V. $D = 2$ is fairly accurate. Although

Additional enhancement	Iterations	Time (sec)
None	3300	357
Flexible VI	2900	258
Initialization	1100	93.6
Lower bound penalty	800	72.8
Reduced model ($D = 2$)	800	36.9
Monotone switching	700	32.0

TABLE IV

IMPACT OF ENHANCEMENTS, CASE 5 ON A SMALLER TRUNCATION

	$D = 5$		$D = 2$		$D = 1$	
Case	Cost	β	Cost	β	Cost	β
3	25.46	0.891	25.06	0.896	21.78	0.927
4	11.79	0.875	11.99	0.894	10.84	0.907
5	14.61	0.777	15.97	0.803	16.44	0.788

TABLE V

ACCURACY OF THE REDUCED MACHINE STATE MODEL

cost increases by 9% when it is used for case 5, service level also increases by 0.026, which keeps the result near the $D = 5$ optimal frontier. Without adjusting the failure rate, the $D = 2$ cost is 6% less than the full model for case 3, compared to 1% with the adjustment.

V. EXAMINING NUMERICAL POLICIES

Figure 2 shows the optimal policy for case 2. The machine 1 switching curve has a "slope" less than or equal to -1, with -1 representing a base stock policy, and the machine 2 curve is nondecreasing, consistent with the conjectured transition monotonicity. When machine 2 fails, the machine 1 curve is higher. Generally, failures were observed to turn a machine off but not on. The control for case 1 is more nearly base stock because the line is balanced.

Many two-dimensional cross sections of the switching surfaces were examined for case 3. All showed a nearly base stock form of policy: insensitive to upstream inventory and switching curve slopes of -1 for downstream inventory. Switching curves are also insensitive to failures,

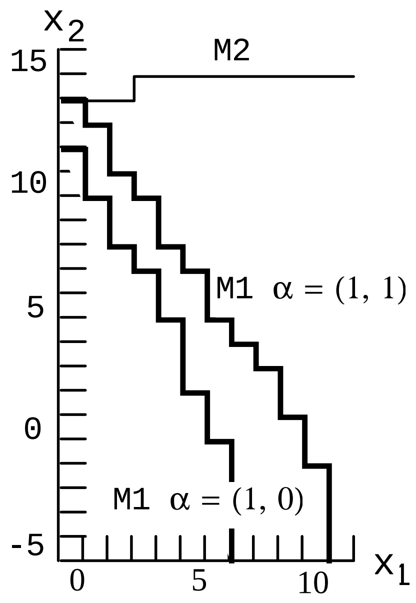

Fig. 2. Switching curves for case 2. Machine 1 (2) idles above the curve M1 (M2).

shifting down by no more than one unit when one or even three machines failed. Numerical studies of similar systems have not detected a dependence on upstream inventory. Some of the switching curves did shift up slightly when downstream inventory increased, suggesting that optimal controls depend on upstream inventory. For example, machine 5 is idle when $x = (0, 0, 0, 1, 10)$ or $(0, 0, 0, 2, 10)$, but not when $x = (0, 0, 0, 3, 10)$, $(0, 1, 2, 1, 10)$ or $(0, 1, 2, 2, 10)$, all of which are recurrent states with $\alpha = 1$.

With faster machines upstream, case 4 is expected to have less of a base stock form. The optimal hedging point is $x = (0, 2, 1, 2, 6)$. The slope of -1 in Figure 3a shows that machine 1 responds to buffer 2 as a base stock policy would. However, the curve is shifted down from the base stock curve, which idles machine 1 when $x_1 + x_2 \geq 13$. In fact, machines 1 and 2 are less sensitive to buffer 5: decreasing x_5 by 10 only shifted the curves up 7 units, not 10. The machine 4 curve in Figure 3b is slightly steeper, gradually deviating from base stock as x_5 decreases.

Figure 3 also shows that the curves shift down slightly when a downstream machine fails. The machine 2 and 5 curves shown are unchanged when any one machine fails. Also, no shift is seen for machine 1 when machine 5 fails. Surprisingly, the shift for machine 1 when machine 2

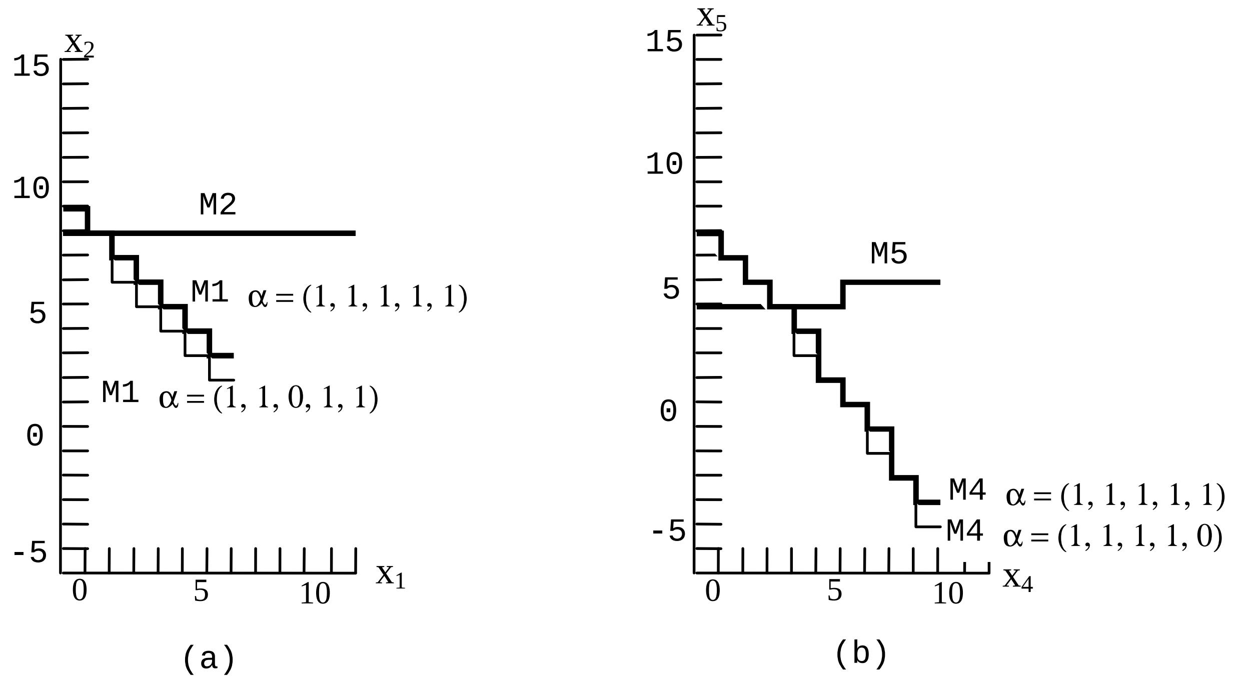

Fig. 3. Switching curves for case 4. (a) Machines 1 and 2, $x_3 = x_4 = 1$, $x_5 = -4$. (b) Machines 4 and 5, $x_1 = 0$, $x_2 = 2$, $x_3 = 1$.

fails is smaller than the shift shown when machine 3 fails. In all the curves examined, the shift when an upstream machine fails is negligible. Case 5, with long repair times, is more sensitive to failures. Some switching curves shifted several units when downstream machines failed.

VI. Conclusions

Carefully adapting the VI algorithm made it possible to compute essentially optimal policies for an unreliable five-machine line. One key was a method to set the truncation in accordance with optimal buffer limits. This method could be applied to any network with monotone control. For a given upper limit on buffers, algorithmic enhancements gave a 91% speedup on a test case. Other possible enhancements include initializing value iteration with the associated fluid value function [4] and, for certain networks, using a power series algorithm for the evaluation step in policy iteration [16]. The numerical findings were

- Control point policies (CPPs) are nearly optimal for balanced lines (with 1% on examples).

- Base stock levels are more relevant than buffer limits in constructing a good policy. For balanced lines a policy with no buffer limits is near optimal.
- A machine's optimal control is fairly insensitive to upstream inventory and machine states.
- Failures of other machines appear to switch a machine's optimal control only from producing to idling.

These findings help justify the use of CPPs and similar policies. They also demonstrate the feasibility of benchmarking heuristic policies against optimal ones for the moderate size networks on which they are usually tested.

APPENDIX A. MONOTONE CONTROL AND BOUNDING THE RECURRENT CLASS

In this appendix a monotone control conjecture is made and used to compute the upper bounds (5) from the optimal policy. A truncation $x_k \geq L$ is assumed to ensure that N_i is finite. Certain submodularity conditions on the value function induce a transition monotone type of policy, namely, a controlled transition can only be turned on, not off, when another type of transition occurs. Transition monotonicity is proven in [15] for a reliable series line, where it implies the following properties.

1) There exist switching functions $s_i(x^{(k)})$ such that machine i is on when $x_k \leq s_i(x^{(k)})$ and off otherwise, where $x^{(k)} = (x_1, \ldots, x_{k-1})$.
2) s_i is nondecreasing in x_j, $j < i$ (upstream inventory) and decreasing in x_j, $j \geq i$ (downstream inventory) with slope no greater than -1, that is, $s_i(x^{(k)}+e_j)-s_i(x^{(k)}) \leq -1$.
3) The switching functions define the boundaries of the recurrent class.

We conjecture that this monotone structure also holds for our model.

Conjecture 1: The transitions $\{e_1, -e_1+e_2, \ldots, -e_{k-1}+e_k, -e_k\}$ are monotone.

Under this condition, the switching function in machine state α, denoted s_i^α, has Properties 1 and 2. A second assumption is needed for Property 3; otherwise, a failure at machine $j \neq i$ could lead to a state beyond s_i^α. Let

$$\overline{u}_i(x) = \max_{\alpha} u_i(\alpha; x) \quad \text{and} \quad \overline{s}_i(x^{(k)}) = \max_{\alpha:\alpha_i=1} s_i^\alpha(x^{(k)}). \tag{9}$$

Conjecture 2: $s_i^1(x^{(k)}) = \overline{s}_i(x^{(k)})$, i.e., the operational state has the largest switching function.

Both conjectures are supported by all the numerical optimal policies we have examined. Let $\mathbf{X}_1$ be the chain for reliable machines using the same policy $u(1; x)$. Under Conjecture 2 the recurrent class of the full chain is the same as that of $\mathbf{X}_1$, i.e., the operational state defines the recurrent class. Also, Properties (2) and (3) apply to $\overline{s}_i$.

Theorem 1: Assume Conjectures 1 and 2. Then

$$N_i = \max_x \{x_i : \overline{u}_i(x + e_{i-1}) = 1 \text{ and } x + e_{i-1} \text{ is accessible from } x$$
$$\text{by some permutation of machine } 1, \ldots, i-1 \text{ transitions}\} + 1.$$

In particular, for the first three buffers,

$$\begin{aligned} N_1 &= \max_x \{x_1 : \overline{u}_1(x) = 1\} + 1 \\ N_2 &= \max_x \{x_2 : \overline{u}_2(x + e_1) = 1 \text{ and } \overline{u}_1(x) = 1\} + 1. \\ N_3 &= \max_x \{x_3 : \overline{u}_3(x + e_2) = 1 \text{ and} \\ &\quad [(\overline{u}_1(x) = 1 \text{ and } \overline{u}_2(x + e_1) = 1) \text{ or } (\overline{u}_2(x) = 1 \text{ and } \overline{u}_1(x - e_1 + e_2) = 1)]\} + 1. \end{aligned} \tag{10}$$

Proof: We prove (10). Only machine 2 service increases x_2, so

$$N_2 = \max_x \{x_2 : x \text{ recurrent and } \overline{u}_2(x) = 1\} + 1. \tag{11}$$

By Property 2, $x = \arg\max$ (11) has $x_i = 0$, $i = 3, \ldots, k-1$ and $x_k = L < 0$ (if $k > 2$). Such states are recurrent if and only if they are accessible from the recurrent state $(1; 0)$. Failures and machine $3, \ldots, k$ service transitions need not be considered, by Conjecture 2 and Property 2, respectively. Property 2 prevents the machine 1 (2) service transitions from crossing the machine 2 (1) switching curve. Hence, $(1; x)$ can be reached from $(1; 0)$ by applying transition $-e_k$ followed by machine 1 and 2 service transitions exactly when the conditions in (10) hold.

■

Note that $x = \arg\max$ (10) lies on the intersection of the machine 1 and 2 switching functions. The general proof establishes that N_i is the maximum of x_i over states x that lie on the switching functions $\overline{s}_1, \ldots, \overline{s}_i$ in the sense that there is some permutation of the machine $1, \ldots, i$ service transitions that lead from x to $x + e_i$ and all these machines are idle at $x + e_i$. Since only machine i increases x_i and, by transition monotonicity, service transitions at other machines cannot turn machine i off, we may assume that the machine i transition occurs last. Thus, all permutations of machines $1, \ldots, i-1$ must be considered.

Appendix B. Adjusted failure rate

Failure rates $\widetilde{p}_i$ are derived for the adjusted model, in which at most D machines are down, that satisfy (7). To simplify the calculations, we make the homogeneity assumptions

$$\gamma = (\lambda/\mu_i)p_i \quad \text{and} \quad r = r_i \quad \text{for all } i. \tag{12}$$

Here γ is the time-average failure rate of each machine. Any stable policy has

$$E[d] = \frac{k\gamma}{r}. \tag{13}$$

Clearly, we must choose D large enough so that $k\gamma/r < D$. Also, under the stationary distribution, the probability that machine i is busy while it is operational is

$$\frac{\lambda/\mu_i}{1 - \lambda p_i/\left(\mu_i r_i\right)} = \frac{\gamma/p_i}{1-\gamma/r} = \frac{\rho}{p_i} \tag{14}$$

where

$$\rho = \gamma/(1-\gamma/r) \tag{15}$$

is the time-average failure rate of a machine while operational. Because failures are operation-dependent, $E[\widetilde{d}]$ depends on the policy. We will approximate $E[\widetilde{d}]$, assuming that failures are time-dependent so that the time-average failure rate $\widetilde{\rho}$ is a failure rate, making $\widetilde{d}$ a birth-death process. We can find the stationary distribution π of this chain in terms of $\widetilde{\rho}$, use it to compute $E[\widetilde{d}]$, then solve (7) and (13) for $\widetilde{\rho}$. Finally, using the fact that (14) is the same for both models,

$$P\{\text{machine } i \text{ busy} \mid \text{machine } i \text{ operational}\} = \rho/p_i = \widetilde{\rho}/\widetilde{p}_i$$

set

$$\widetilde{p}_i = \frac{\widetilde{\rho}}{\rho}p_i. \tag{16}$$

For $D = 1$, $E[\widetilde{d}] = \pi_1 = k\widetilde{\rho}/(r + k\widetilde{\rho})$, $\widetilde{\rho} = \gamma/(1 - k\gamma/r)$, and

$$\widetilde{p}_i = \frac{1-\gamma/r}{1-k\gamma/r}p_i. \tag{17}$$

For $D = 2$, a similar analysis results in

$$\widetilde{\rho} = \frac{k\gamma - r + \sqrt{(r-k\gamma)^2 + 2(2r-k\gamma)\gamma(k-1)}}{(k-1)(2-k\gamma/r)}. \tag{18}$$

Then (16), (12) and (15) are used to obtain $\widetilde{p}_i$.

Acknowledgements

The numerical study was done with W. Owen Webb and Jonathan Senning. I am also grateful to Stanley Gershwin and Francis de Vericourt for their helpful comments.

References

[1] F. Karaesmen and Y. Dallery, “A performance comparison of pull type control mechanisms for multi-stage manufacturing,” *Int. J. Production Economics*, vol. 68, pp. 59–71, 2000.

[2] M. Veatch and L. Wein, “Optimal control of a two-station tandem production/inventory system,” *Oper. Res.*, vol. 42, pp. 337–350, 1994.

[3] M. Veatch and L. Wein, “Scheduling a make-to-stock queue: Index policies and hedging points,” *Oper. Res.*, vol. 44, pp. 634–647, 1996.

[4] R.-R. Chen and S. Meyn, “Value iteration and optimization of multiclass queueing networks,” *Queueing Systems Theory and Appl.*, vol. 32, no. 1-3, pp. 65–97, 1999.

[5] D. Bertsimas, I. Paschaladis, and J. Tsitsiklis, “Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance,” *Ann. Appl. Probab.*, vol. 4, pp. 43–75, 1994.

[6] J. Morrison and P. Kumar, “New linear program performance bounds for queueing networks,” *J. Optim. Theory Appl.*, vol. 100, no. 3, pp. 575–597, 1999.

[7] J. Kimemia and S. Gershwin, “An algorithm for the computer control of production in a flexible manufacturing system,” *IIE Trans.*, vol. 15, pp. 353–362, 1983.

[8] I. Paschaladis, C. Su, and M. Caramanis, “Target-pursuing scheduling and routing policies for multiclass queueing networks,” *IEEE Trans. Automat. Control*, vol. 49, pp. 1709–1722, July 2004.

[9] M. Veatch and J. Senning, “Fluid analysis of an input control problem.” Working paper, Gordon College, Dept. of Math. Available at faculty.gordon.edu/ns/mc/Mike-Veatch/index.cfm, May 2003.

[10] S. Gershwin, “Design and operation of manufacturing systems-the control point policy,” *IIE Trans.*, vol. 32, pp. 891–906, 2000.

[11] L. Sennott, “Computing average optimal constrained policies in stochastic dynamic programming,” *Probab. in the Engr. and Info. Sciences*, vol. 15, pp. 103–133, 2001.

[12] E. Altman, *Constrained Markov Decision Processes*. Boca Raton, FL: Chapman and Hall/CRC Press, 1999.

[13] M. Caramanis, “Continuous time DP problems with controllable markov jump process dynamics.” Unpublished notes, Boston University, Dept. of Manufacturing Engr., May 2003.

[14] F. de Vericourt and S. Gershwin, “Performance evaluation of a make-to-stock production line with a two-parameter-per-machine policy: The control point policy.” To appear, IIE Trans., 2003.

[15] M. Veatch and L. Wein, “Monotone control of queueing networks,” *Queueing Systems Theory and Appl.*, vol. 12, pp. 391–408, 1992.

[16] G. Koole and A. Pot, “Workload minimization in re-entrant lines.” Technical report 2002-11, Dept. of Stochastics, Vrije Universiteit, Amsterdam. To appear, Euro. J. Operational Res., 2005.