Optimal service policies under learning effects

Geoffrey S. Ryder*, Kevin G. Ross, and John T. Musacchio

Technology and Information Management Department
University of California, Santa Cruz
1156 High Street, Santa Cruz, California, USA 95064
Fax: 650-528-4034 E-mail: gryder@soe.ucsc.edu
E-mail: kross@soe.ucsc.edu
E-mail: johnm@soe.ucsc.edu
*Corresponding author

Abstract:
For high-value workforces in service organizations such as call centres, scheduling rules rely increasingly on queueing system models to achieve optimal performance. Most of these models assume a homogeneous population of servers, or at least a static service capacity per service agent. In this work we examine the challenge posed by dynamically fluctuating service capacity, where servers may increase their own service efficiency through experience; they may also decrease it through absence.

We analyse the special case of a single agent selecting between two different job classes, and examine which of five service allocation policies performs best in the presence of learning and forgetting effects. We find that a type of specialisation minimises the steady state queue size; cross-training boosts system capacity the most; and no simple policy matches a dynamic optimal cost policy under all conditions.

Keywords: operations models in services, service science, service engineering, optimal allocation policies, capacity management, queueing theory, learning, forgetting, Markov decision process, dynamic programming

Reference to this paper should be made as follows: Ryder, G., Ross, K., and Musacchio, J. (2008) 'Optimal service policies under learning effects', *Int. J. of Services and Operations Management*, Vol. 4, No. 6, pp.zzz–aaa.

Biographical Notes: Geoffrey S. Ryder is currently a Ph.D. candidate in Technology and Information Management at the University of California, Santa Cruz. He has a BSEE and BA (Asian Studies) from the University of Illinois at Urbana-Champaign, and a MS (Computer Engineering) from UCSC. His research interests involve the application of statistical methods, decision theory, and optimisation theory to study problems in the growing fields of service engineering, service management, and service delivery networks.

Kevin G. Ross is an Assistant Professor of Technology and Information Management at the University of California, Santa Cruz. His

Copyright © 2008 Inderscience Enterprises Ltd.

expertise is in Service Network Engineering and Management, applying operations research techniques from scheduling, queueing theory and optimisation to problems in human resource planning, call centres, autonomic computing networks, flexible manufacturing and service enterprise management. His contributions include developing classes of throughput maximizing policies for general multi-class service networks, demonstrating a robust framework for analysis and design, and enabling the control of quality of service, load balancing and scalability. He gained his Ph.D. from Stanford University in Management Science and Engineering, in 2004.

John T. Musacchio is an Assistant Professor of Information Systems and Technology Management at the University of California, Santa Cruz. His expertise and research interests include stochastic modelling and control of queuing networks with applications in communications networks, manufacturing and service systems, network pricing, and game theory. His contributions include fairness results for control policies in queueing networks, and equilibrium results for pricing policies in wireless networks in the presence of asymmetric information. He gained his Ph.D. from the University of California Berkeley in Electrical Engineering and Computer Science in 2005.

1 Introduction

1.1 The Effect of Work Allocation on Skill Development

A customer service manager considers how to assign tasks to one of her subordinates. The assignment must take into account the customers' needs and the agent's skill at handling them. She assigns the agent to process two classes of incoming jobs, as in Figure 1. Customers who cannot immediately enter service join the queue with other customers who share the same type of problem to be solved.

The managers of a service facility might often use a queueing model to assign jobs to agents, and these models typically assume that an agent's mean service time is static. In reality, the agent's skill fluctuates due to his experiences over time. For the purpose of this paper, we equate his mean service rate with his skill level and investigate the impact of such experience-based fluctuations on the agent's performance over a time span of weeks to months.

As a concrete example, suppose company *ABC* sells a new nanotechnology chemical sensor to customers in two different industries. The company sells a bundled product containing the sensor hardware, cabling, software driver, and a data acquisition program. A service engineer (the *agent*) at the company's customer support centre handles incoming calls regarding this product.

Calls of type x come from equipment suppliers to medical diagnostic labs, which use the sensor to characterise patient fluid and tissue samples. Calls of type y come from biotechnology factories, which use the product as part of a control loop within the fragile manufacturing process for genetically engineered cancer drugs.

If the agent takes calls of type x exclusively, he will become very familiar with the medical users' situation: how often the sensor must be replaced for different uses, how the interface program can be used and customised, and what operating

settings give the most accurate results. Over time, his knowledge of these matters accumulates and he becomes a medical customer support specialist.

However, type y callers face a much different situation, with higher volumes of different material flowing by their sensors, higher interference from surrounding equipment, and so on. Experience relevant to type x jobs is not transferable; separate experience with type y jobs is indispensable in order to be proficient at helping those customers.

Depending on the allocation of time and experience, the agent's ability to advise customers on these matters will fluctuate in both speed and quality. Here, we assume a constant unchanging quality, and focus on the cost impact of the improvement in the speed of answering.

New service tools, new organizational arrangements, and learning effects may all drive changes in his mean service rate. Here we assume all changes are caused by the latter. There is a considerable literature describing the phenomenon of learning curves; among those are recent empirical results describing the effects of learning and forgetting on the job. They make clear that learning and forgetting effects seen in human performance data may play a key role over the short and medium terms; since the job categories themselves keep changing in modern dynamic service environments, learning effects may play important roles over long term time horizons as well.

We develop a simple stochastic model that accommodates such on-the-job learning and forgetting results, and in this paper we present conclusions drawn from our experiments with it that suggest how to assign agents to tasks in ways that improve system performance. A good assignment policy must resolve the following question: with respect to the backlog in the queues, the agent's skill levels, and potential experience-based changes in his skills, from which queue should the agent select his next job?

2 Background and Motivation

2.1 Operations Research Studies of Learning Effects

Researchers and practitioners have recently highlighted the potential of applying a more fine-grained approach to stochastic learning models. Aksin et al. (2007), Bodreau et al. (2003) and Dietrich and Harrison (2006) examine areas of overlap between operations research (OR) and behavioural science, and call for more investigation of the effects of learning and forgetting. The phenomenon of improved performance through learning is observed in many settings, in both individuals and in organizations (Badiru (1992); Mazzola and McCardle (1997); Reis (1991); Schilling et al. (2003)). Shafer et al. (2001) derived a learning curve model from empirical observations of manufacturing workers that includes both learning and forgetting. Thompson (2007) and Zamiska et al. (2007) contribute new insights into how forgetting mechanisms do or do not manifest themselves, and support the idea that an agent's proficiency in a task drops when that task is neglected.

Several recent papers are notable for their inclusion of agent learning in the analysis of systems. Eitzen et al. (2004) and Sayin and Karabati (2007) consider worker skill development and maintenance as important factors when optimising

staffing schedule assignments. Misra et al. (2004) explored the impact of experience effects on the structure of a sales force; Tucker et al. (2007) studied team learning in hospital intensive care units. Whitt (2006) develops analytical means to describe the distribution of performance over a population of agents as their experience increases.

The approach of Gans and Zhou (2002) is similar to ours in that they use a Markov decision process model to explore the effects of learning in a service organization. In their model, employees have a probability of learning and increasing their skill at each decision epoch, and at the same time a probability of leaving the firm due to turnover. As future research they recommended exploring ways to model stochastic learning effects more precisely, as they may significantly affect the firm's performance. In our model each epoch represents a single customer departure (a single job completed) in a queueing model, so we examine the issue at the level of the routing decision. We do not consider turnover.

Harrison (1975), Hasija et. al (2005), Iravani et al. (2007), Wallace and Whitt (2005), Yuen (2006) and Pinker and Shumsky (2000) evaluate the tradeoffs involved in cross-training workers. In our paper the single service agent is servicing two queues, each with a different job type, requiring two distinct learning curves; in that respect we are also observing the effects of cross-training.

2.2 Design Details from Related Models

We would like to know the costs that result from the agent's choice of which job class to serve next. We investigate that choice by formulating the problem as a Markov decision process which can be solved using a dynamic programming solver (Bertsekas (1995), George and Harrison (2001), Mazzola and McCardle (1997)). The solver computes each state's cost, which is the sum of the holding costs of both queues for that state plus its cost-to-go to other states, and the agent's best choice may change from one state to the next.

The customers in one queue are of a different type than customers in the other, so they can be thought of as two separate job classes with different service rates. The system consists of two parallel Markovian FCFS queues with exponentially distributed service and interarrival times. In addition to learning while serving, an agent has a chance of forgetting skills learned for one queue during time periods spent serving the other queue. This behaviour is a simplified version of the *recency* effect discussed in Nembhard and Osothsilp (2001), and Shafer et al. (2001). Following Hampshire et al. (2006), the extended Kendall notation $M/M_t/1/K$ can be used to describe queues with changing service times such as these. Caro and Gallien (2007) use multiple coupled dynamic programs that include approximation methods for switching costs and other hard-to-model phenomena; that may be a path to expanding upon the results we report here in future work.

If our model did not incorporate learning and forgetting, the optimal policy would be given by the well-known μc rule (Baras et al. (1985); Harrison (1975); Koole (1997); Mandelbaum and Stolyar (2004); Ryokov and Lembert (1967)), which says it is optimal to serve the queue for which the product of the service rate μ and the queue occupancy cost c is a maximum. We find that the μc rule is in fact not the best rule to follow for this problem when the parameters corresponding to learning curve effects are significant.

3 Model Formulation and Analysis

3.1 Adding Learning Effects to the Model

Gans and Zhou (2002)'s call centre learning study considers three levels of agent performance: baseline, a 40% improved service rate, and an 80% improved service rate. Following this we assign the vector of improving service rates for μ_x and μ_y to be $m \cdot [1, 1.4, 1.8]$, where the base service rate is $m \approx 0.25$ jobs per minute.

Shafer et al. (2001) provides a wealth of data, in particular the average learning and forgetting rates for a group of workers conducting a detailed assembly line operation. The testing station workers incurred forgetting effects during interruptions in their work assignments, although they did not switch between two different tasks, as we assume here. In order to incur forgetting losses in our Markov chain model with three learning curve states, forgetting transitions incur a performance penalty by pushing the agent back to less proficient states. We then analyse performance using metrics based on the long-run steady state distribution of the chain.

Their average worker might achieve his highest learning curve state after completing 1,000 jobs, while others learned much faster. Our learning curve has three states, and we allow the mean jump rate from a slower service rate to a faster one to range from $(\frac{1}{500} \cdot m)$ up to $(\frac{1}{2} \cdot m)$. So with regard to the data from the product testing study, we are focusing on workers with above average learning rates. We use various values for the forgetting rate that are within the range suggested by the study, with our default forgetting rate being one-third of the learning rate.

3.2 Policy Definitions and Time Series Examples

We analyse five policies that could be used to direct the agent's work.

1. Serve the longest queue first, or the *LQF* policy, Figure 2.

2. Serve the job class that the agent is most skilled (fastest) at handling, or *SMax*, Figure 3. This could be considered a policy of specialisation in Pinker and Shumsky (2000). Note that such a policy is preferred in many settings. In policies for managing priority queues, this is the *shortest processing time* rule, or *SPT* (Gross and Harris (1998); Schrage and Miller (1966); Tezcan (2006)).

3. Serve the job class that the agent is slowest at handling, or *SMin*, Figure 4. This could be considered a policy of cross-training.

4. Serve the job class specified by the μc rule, or *MuC*, Figure 5.

5. Serve the job class specified by the dynamically optimal cost service rule, or *Opt*, Figure 6. *Opt* is the only one of the five for which policy iteration is used to alter the policy choice at each state according to the step-by-step value iteration results from the dynamic program.

3.3 Dynamic Programming for Optimisation

The dynamic programming solver computes the optimal policy to follow for each state in a discrete Markov chain. As a Markov decision process (MDP), a decision maker acts to minimize the expected cost of the present value at time zero of the stream of costs incurred. Assumptions made during the design of the solver include:

- Advancing in skill does not depend on how long the agent has been in a state, but only if the last job served was relevant to this learning curve. Therefore learning and forgetting are memoryless state transitions, which together with the queueing processes form a four-dimensional Markov chain.
- The agent attends one of the two queues at all times, with no breaks or server vacations.
- There is a finite number of states and an infinite planning horizon. Discount factor β (where $0 < \beta < 1$) discounts the future cost of visiting other states.
- i denotes one of the states, where each i is an index number for a uniquely valued tuple $\{q_x, q_y, \ell_x, \ell_y\}$. Here q_x is the number of customers in the first queue, q_y is the number of customers in the second queue, ℓ_x is the learning state (specifying a service rate μ_x) of the server for type x jobs, and ℓ_y is the learning state of the server for type y jobs.
- The number of valid transitions and their relative probabilities are determined by the policy u adopted for that state, $u \in \{u_x, u_y, u_{tie}\}$. Under u_{tie}, ties in the DP solver are resolved by including transition probabilities for serving both queues and renormalizing.
- The static cost is the sum of the number of customers waiting in the two queues at each state. The holding cost of the backlogs is assumed to be \$1 per waiting job per unit time. Customers to not balk or renege. Arrivals to full queues are turned away, and a one-time penalty $M = \$100$ is charged to the system to represent the impact of dropping a customer. The number of waiting positions in the two buffers is $K_x = K_y = 9$.
- $J(i, u)$ is Bellman's equation, the cost functional equation for state i and policy u. The dynamic program iterates until $|(J^k(i, u) - J^{k-1}(i, u))| < \epsilon$, for every state i and policy u, and reports the final value vector $\vec{J}^*$ for each of the five policies.

The process of uniformisation is used to convert the continuous time queueing problem to discrete time. Five exponential processes govern state transitions, and the components of the uniformisation constant ν are the largest values of those processes found among all of the model states. Let a star next to a parameter denote its largest value anywhere in the model. The five parameters are: λ_x, the arrival rate to x; λ_y, the arrival rate to y; μ, the departure rate from one of the queues (not both); ϕ, the learning rate; and ψ, the forgetting rate. Then the fastest exponential race transition out of any state is $\nu = \lambda_x^* + \lambda_y^* + \mu^* + \phi^* + \psi^*$. In the experiments the arrival rates for both queues are fixed and equal to each other, so $\lambda_x = \lambda_y = \lambda_x^* = \lambda_y^*$. For each individual state, we have $\nu_i(u) = \lambda_x + \lambda_y + \mu_i(u) + \phi_i(u) + \psi_i(u)$.

3.4 Structure of the Transition Matrix p(u)

Consider Figure 7, which illustrates the policy "serve x" u_x in the model state $J^k(q_x, q_y, \ell_x, \ell_y, u_x)$. The diagram shows three of the waiting positions, plus empty, for four queue states on each side, and it shows three learning states for each job type. Learning, service, and an arrival are possible for queue x, while forgetting and an arrival are possible for queue y. Self-transitions at each queue are also allowed due to uniformisation. The activities of the two queues are considered independent. Bellman's equation then becomes

$$J^k(q_x, q_y, \ell_x, \ell_y, u) = \frac{1}{\beta+\nu}\left[q_x(i) + q_y(i) + (\nu - \nu_i(u)) \cdot J^{k-1}(q_x, q_y, \ell_x, \ell_y, u^*)\right] + \frac{1}{\beta+\nu}\left[v_i(u) \cdot \left(\sum_j p_{ij}(u) J(j)\right)\right].$$

Here $q_x(i)$ and $q_y(i)$ are the fixed queue holding costs in state i. To minimize the right-hand side of this equation we seek a policy in each state such that the cost of the policy-dependent terms is a minimum. In the cost-to-go term, the transition probabilities p_{ij} take the form $\left(\frac{\lambda_x}{\lambda_x+\lambda_y+\mu_x+\phi_x+\psi_y}\right)$. This particular p_{ij} value is for an arrival of type x, for a policy "serve x"; the other p_{ij}s are similar but with the appropriate value in the numerator.

3.5 Steady State Measures

The transition matrix p_{ij} is constructed in such a way that it is an irreducible, ergodic Markov chain with all states positive recurrent, and so it has a stationary distribution $\vec{\pi}$ defining the steady-state probability of being in each state. The dynamic programming solver finds this distribution.

Based on $\vec{\pi}$, the mean learning state value $\ell_{\vec{\pi}}$, mean queue backlog $(q_{x\vec{\pi}} + q_{y\vec{\pi}})$, and total cost for all states in the system $C_{\vec{\pi}}$ are computed. $\sum_{i=1}^{N} \pi(i) = 1$, so $\vec{\pi}$ serves as a means of weighting the values of $\ell(i)$, $q(i)$, and $C(i)$ according to the steady state likelihood of the system being in state i.

Recall that the fixed cost portion of each state's cost is the sum of the two queue backlogs; $(q_{x\vec{\pi}} + q_{y\vec{\pi}}) = \sum_{i=1}^{N} \pi(i) \cdot [q_x(i) + q_y(i)]$. $C_{\vec{\pi}}$ includes this fixed cost, and also includes the cost-to-go to neighbours at each state, with penalties if applicable: $C_{\vec{\pi}} = \sum_{i=1}^{N} \pi(i) \cdot J^*(i)$. So the performance metric $(q_{x\vec{\pi}} + q_{y\vec{\pi}})$ is a part of the metric $C_{\vec{\pi}}$, and typically $C_{\vec{\pi}}$ will be roughly twice as large.

Note that in the cases we consider here, the learning, forgetting, and service mechanisms affect each job class equally. Thus the steady state metrics that describe experimental outcomes in the next section are symmetric in job classes x and y. In the following sections the terms *long run, steady state, $\vec{\pi}$-weighted,* and *mean* are equivalent.

Each experiment is considered an exact computation for a hypothetical agent who has those characteristics. Truncation error from limiting the number of DP solver iterations is approximately 1e-9 for the value function J in each state, and roundoff error incurred computing $\vec{\pi}$ is near 1e-16 for each state.

Table 1 Parameter settings for all simulation runs, which define the fixed reward structure.

Parameter	*Value*
Number of service agents	1
Number of job (queue) types	2
Number of queue buffer states $K = K_x = K_y$	9
Number of learning curve states	3
Discount factor β	0.1
Total # of dynamic programming states N	1800
Minimum service rate m	0.25 departures per minute
Improving service rates μ_x, μ_y	$[m \;\; 1.4m \;\; 1.8m]$

Table 2 Learning and forgetting parameters that were varied in simulations.

Learning Parameter	*Symbol or Abbreviation*	*Sample Value* ($m = 0.25$ jobs per minute)
Arrival rate	λ_x, λ_y	$0.4 \cdot m$ to $0.75 \cdot m$, or 50% to 100% capacity
Moderate learning rate	ϕ_x, ϕ_y	$\frac{1}{500} \cdot m$
High learning rate	ϕ_x, ϕ_y	$\frac{1}{32} \cdot m$
Very high learning rate	ϕ_x, ϕ_y	$\frac{1}{2} \cdot m$
Default forgetting rate	ψ_x, ψ_y	$(\phi/3) \cdot m$

4 Experimental Results

4.1 Experimental Parameters

Simulation inputs are a specific, unchanging dynamic programming reward structure, Table 1, combined with specific values of learning parameters that are allowed to vary, Table 2. The experiments we present have three learning curve states and ten queue states (including empty) for two job classes.

As noted in Section 3, the learning rate ϕ varies from $(\frac{1}{500} \cdot m)$ up to $(\frac{1}{2} \cdot m)$ in these experiments. The agent therefore steps up the learning curve roughly at rate $\frac{\phi}{m}$, or from once every 500 to once every two jobs completed. Unless stated otherwise, the forgetting rate is $\psi = \frac{\phi}{3}$. The value $\phi = \frac{1}{500} \cdot m$ is referred to as a *moderate* learning rate case; $\phi = \frac{1}{32}$ is *high* learning; and $\phi = \frac{1}{2}$ is *very high* learning.

Server utilisation is computed as the fraction $\frac{\lambda_x + \lambda_y}{E[\mu]}$. Here the expected service rate $E[\mu]$ is the service rate at the mean learning curve state for this trial, as weighted by the stationary distribution $\vec{\pi}$.

4.2 The Character of the Five Policies

Table 3 gives metrics for evaluating the performance of the five policies. The first row under each utilisation value gives the mean queue length in steady state,

Table 3 Comparison of policies by different $\vec{\pi}$-weighted measures, which are defined in Section 4.1. Data is for a *moderate* learning rate; $\psi = \phi/3$; and the system faces arrival rates into both buffers that give 75%, 90%, and 100% server utilisation. The best two policies by each metric are shown in bold. Trends seen here are accentuated at *high* and *very high* learning rates.

75% utilisation						
	Units	*LQF*	*SMax*	*SMin*	*MuC*	*Opt*
$q_{x\vec{\pi}} + q_{y\vec{\pi}}$	jobs	2.83	**2.44**	2.67	2.73	**2.48**
$\mu_{x\vec{\pi}}$, $\mu_{y\vec{\pi}}$	jobs/min	1.62	1.62	1.62	1.62	1.62
$(o_{x\vec{\pi}} + o_{y\vec{\pi}}) * 1000$	jobs/min	**1.30**	2.52	3.93	**1.51**	2.25
$C_{o\vec{\pi}}$	cost/min	**0.76**	1.16	1.76	**0.82**	1.07
$C_{\vec{\pi}}$	cost/min	5.59	**5.19**	5.96	5.43	**5.19**
90% utilisation						
	Units	*LQF*	*SMax*	*SMin*	*MuC*	*Opt*
$q_{x\vec{\pi}} + q_{y\vec{\pi}}$	jobs	4.75	**3.84**	4.22	4.54	**3.96**
$\mu_{\vec{\pi}}$, $\mu_{y\vec{\pi}}$	jobs/min	**1.64**	1.63	**1.65**	**1.64**	1.63
$(o_{x\vec{\pi}} + o_{y\vec{\pi}}) * 1000$	jobs/min	**5.85**	9.06	11.87	**6.48**	8.05
$C_{o\vec{\pi}}$	cost/min	**2.32**	3.19	3.95	**2.43**	2.88
$C_{\vec{\pi}}$	cost/min	9.57	**8.76**	9.74	9.26	**8.73**
100% utilisation						
	Units	*LQF*	*SMax*	*SMin*	*MuC*	*Opt*
$q_{x\vec{\pi}} + q_{y\vec{\pi}}$	jobs	6.63	**5.23**	5.58	6.33	**5.26**
$\mu_{\vec{\pi}}$, $\mu_{y\vec{\pi}}$	jobs/min	**1.65**	1.64	**1.66**	1.64	1.64
$(o_{x\vec{\pi}} + o_{y\vec{\pi}}) * 1000$	jobs/min	**12.79**	17.83	21.13	**14.09**	17.51
$C_{o\vec{\pi}}$	cost/min	**4.89**	5.42	6.50	**4.91**	5.34
$C_{\vec{\pi}}$	cost/min	12.97	**11.48**	12.51	12.47	**11.45**

$q_{x\vec{\pi}} + q_{y\vec{\pi}}$. Performance on this measure usually (but not always) matches the total simulation cost weighted by the stationary distribution, $C_{\vec{\pi}} = \vec{\pi} \cdot \vec{J}^*$, shown on the fifth row. The second row gives the long run service rate $\mu_{\vec{\pi}}$; this is the long run system capacity. Because of the symmetry in the characteristics of job classes x and y, we have $\mu_{x\vec{\pi}} = \mu_{y\vec{\pi}}$. The third and fourth rows measure the impact of buffer overflow on the system. Define $o_{\vec{\pi}} = \lambda \vec{\pi}_{\text{over}}$, where $\vec{\pi}_{\text{over}}$ contains the values of the stationary distribution for states with one or both buffers full. $C_{o\vec{\pi}}$ measures the contribution of those full buffer states towards the total cost $C_{\vec{\pi}}$. Figure 8 shows plots of the $C_{\vec{\pi}}$ values from Table 3.

Each of the five policies does well on at least one of the metrics. *LQF* is the best at minimizing the costs due to buffer overflow. *SMax* is best at minimizing the mean queue backlogs. *SMin* is best at maximizing the system's long run capacity. *MuC* is similar to *LQF*, but trades off some of the ability to minimize overflow for a lower total cost, $C_{\vec{\pi}}$, than *LQF*. Finally, *Opt* uses the advantage of policy iteration to achieve the lowest total cost among all of them.

In Table 3, with the learning rate set to *moderate*, the characteristics of the five policies start to become significant, giving performance differences of between 0.5% to 10% on the various metrics. This is roughly at the level of the average learner case from Rossiter (1987). At higher learning rates these policy differences are even more pronounced.

4.3 Learning and Forgetting Effects

The fundamental dynamic driving the results is that increased learning decreases the holding costs in the queue by achieving higher average service rates. Figure 9 illustrates this by showing a plot of the long run cost $C_{\vec{\pi}}$ against the arrival rate into the system, using the optimal cost policy *Opt*. Note that if forgetting is absent, learning pushes the service rates to their maxima for every policy in steady state. That trivial result does not occur when forgetting effects are present, and Figure 9 shows how increasing the forgetting rate ψ increases the cost.

While forgetting can occur in any state, learning-based service rate improvement is not allowed when the system is empty. Higher arrival and utilisation rates thus benefit the agent by giving him a chance to practice his skills and ascend the experience curve.

4.4 Overflow Penalty Effects

When a customer arrives at a full buffer, she is turned away and denied service. This event triggers a cost penalty of $M = \$100$. The penalty's effect increases if λ_x and λ_y are large in proportion to the other rates. For instance, if queue x is full, the penalty becomes $\left(\frac{\lambda_x \cdot M}{\lambda_x + \lambda_y + \mu_x + \phi_x + \psi_y}\right)$. Its impact on $C_{\vec{\pi}}$ is further adjusted by how often the state is visited and the mean length of each visit (from $\vec{\pi}$, ν and $\nu(i)$).

Figure 10 illustrates the effect of the buffer overflow penalty over the five policies. Only *Opt* is able to react to the increased cost in full buffer states and reduce its cost by choosing a different queue. This is reflected in Table 3 where $o_{\vec{\pi}}$, $C_{o\vec{\pi}}$, and $C_{\vec{\pi}}$ are consistently greater for *SMax* as compared to *Opt*. This is noticeable at

Table 4 Values of the stationary distribution $\vec{\pi}$ for policies *SMax*, *SMin*, and *MuC*. They are organized according to the probability mass found at each learning curve state. Subscript *1* indicates the lowest service rate, and subscript *3* the highest. The optimal policy follows the *SMax* pattern below closely; *LQF* follows the *MuC* pattern closely. The system utilization was set to 90% of capacity, ϕ was *very high*, and ϕ was set equal to ψ. When $\phi = \frac{1}{3}\psi$, similar distributions are obtained, but with a skew toward the northeast corners because the forgetting rate is smaller.

SMax				**SMin**				**MuC**			
y_3	**0.24**	0.08	0.05	y_3	0.05	0.09	**0.14**	y_3	0.17	0.09	0.07
y_2	0.09	0.06	0.08	y_2	0.11	**0.16**	0.09	y_2	0.11	0.08	0.09
y_1	0.09	0.09	**0.24**	y_1	**0.20**	0.11	0.05	y_1	0.11	0.11	0.17
	x_1	x_2	x_3		x_1	x_2	x_3		x_1	x_2	x_3

higher values of ϕ as well. For instance, at a *high* learning rate and 75% utilisation, *SMax* incurred a 1.5% higher cost $C_{\vec{\pi}}$ than *Opt*.

4.5 Approximating the Optimal Policy

The dynamic optimal cost policy *Opt* is found at each state by a combination of value and policy iteration in the DP solver. In other words, the form of *Opt* at state i is influenced by the cost of i's neighbours. In the process of seeking the optimal cost-to-go for each state, *Opt* may sacrifice its performance to a small degree on other metrics, such as the steady state overflow cost and the maximum system capacity (Table 3). For all the learning parameter ranges we study, the (static) policy *SMax* turns out to be a very good approximation to *Opt* for the purpose of minimizing the total cost. However, *SMax* does not respond well to cost changes at boundaries (Figure 10). Section 4.7 describes cases where *MuC* is closer to *Opt* by the metric $C_{\vec{\pi}}$.

4.6 Specialisation Versus Cross-Training

Figure 11 and Table 4 give more insight into the difference between *SMax* and *SMin*, or the difference between specialising and cross-training.

On the left side of Figure 11, *SMin* gives up to a 5% improvement over *SMax* in long run system capacity $\mu_{\vec{\pi}}$ as the learning rate is varied, for a system at 100% utilisation. The forgetting rate is held constant at $\psi = \frac{1}{12} \cdot m$, or one-sixth times the *very high* learning rate. While *SMin* was the best policy for boosting $\mu_{\vec{\pi}}$ over the long run, *LQF* never differed by more than 3% from *SMin* by this metric; *LQF* itself can be considered a good cross-training policy. In fact *LQF* may be preferred in practice since it does not require an estimate of the agent's skill level.

The right side of Figure 11 shows the system steady state backlog, $(q_{x\vec{\pi}} + q_{y\vec{\pi}})$, a metric for which policy *SMax* performs better. *SMin* generates a 25% to 40% worse value for the backlog, a metric for which the trends are usually aligned with the total system cost $C_{\vec{\pi}}$. Here ψ is held at the same value as on the left.

SMax generates a centrifugal trend in service rate improvement that discourages a balanced skill set, while *SMin* promotes balanced skills. These trends are evident in the steady state distributions $\vec{\pi}$ for those policies, which are described in Table 4.

Table 5 The best simple policy by the total cost metric $C_{\vec{\pi}}$ may not agree with that given by the sum of the backlogs $(q_{x\vec{\pi}} + q_{y\vec{\pi}})$ when the server is absent for long stretches of time while away serving the other job class.

K_{th}	Lowest $(q_{x\vec{\pi}} + q_{y\vec{\pi}})$	Lowest $C_{\vec{\pi}}$	Second Lowest $C_{\vec{\pi}}$
4	*SMax*	*SMax*	*MuC*
5	*SMax*	*MuC*	*SMax*
6	*SMax*	*MuC*	*SMax*
7	*SMax*	*MuC*	*LQF*

In all our experiments *SMax* performed better than the other policies as a means of minimizing $(q_{x\vec{\pi}} + q_{y\vec{\pi}})$.

4.7 Performance Penalties from Service Level Agreements

Up to now we have used a simple holding cost for the queue that is linearly increasing in the size of the backlog, and in fact is equal to the size of the backlog in dollars per minute. But real organizations such as call centres may be contractually obligated to provide a certain service level, with financial penalties for nonperformance (Cleveland and Mayben (2000); Gans et al. (2003)). These penalties will be assessed by observing if customer waiting times are too long.

For this example consider queue backlogs to be proxies for customer waiting times. Then a cost penalty is assessed whenever the queue length goes beyond a threshold. In Table 5, our two buffers are each segmented into low cost states (from zero to threshold K_{th}) and high cost states (from $K_{th} + 1$ to K). An additional fixed cost penalty $M = \$25$ per minute is assessed for being in each of the high cost states. The learning rate is set to *high*, and utilisation is 75%. In all cases policy *Opt* has the lowest cost $C_{\vec{\pi}}$, and we would like to know which of the simpler policies best approximates this cost. With these parameters and no thresholds, we found that this was *SMax*.

When the threshold K_{th} is low, and most of the queue states accrue penalty costs, then all the policies suffer about equally. When the threshold is four fifths or more of the buffer size, *MuC* and *LQF* have lower costs than *SMax*. Yet note that *SMax* still gives the lowest mean steady state queue length $q_{\vec{\pi}}$.

The problem for *SMax* is that the agent leaves one queue alone for long periods, and the unattended job class slips into penalty-accruing states often during his absences. The more equitable policies *MuC* and *LQF* reduce these absences – visible in the time series of Figure 2 and Figure 3 – and so perform better than *SMax* here. This example demonstrates that the best steady state cost $C_{\vec{\pi}}$ may not agree with the lowest mean steady state backlog $(q_{x\vec{\pi}} + q_{y\vec{\pi}})$; $C_{\vec{\pi}}$ is a better measure of the impact of server absences when those absences are costly.

The act of reducing the buffer sizes will produce a similar result to the imposition of performance penalties described above; *MuC* and *LQF* become more attractive. On the other hand, preliminary data indicate that *SMax* retains its superiority in runs with larger buffer sizes.

5 Conclusions and Future Work

In summary, we have examined a model of one agent serving two parallel queues, each queue holding a separate and unrelated type of customer. We seek the best policy to follow, or in other words which of the two should be served, for each combination of queue states. These queue states impute holding costs for the service system in proportion to the sum of the queue occupancies and penalties for missed jobs. An additional factor in this model is that the agent may increase his service rate through learning-by-doing, and decrease his service rate through forgetting when ignoring a job class. The choice of which to serve thus affects future performance, and makes the policy choice more complicated.

Policies *SMax* and *MuC* were the best approximations to the optimal cost policy in terms of their steady state cost in different situations. *SMin* maximized the steady state capacity of the system, and LQF minimized the costs due to buffer overflow. We emphasized learning rates that were average to high by comparison with prior empirical data. But with the increasing availability of online information and knowledge management tools, high learning rates may become the norm in service industries in the future. The effect of forgetting may be ameliorated by the ease of accessing persistent electronic records, making a cross-training strategy with policy *SMin* more viable.

For the sake of brevity we only include results here for job classes with symmetric learning curves. Issues of asymmetric learning curves and class priorities will lead to different outcomes, due to the inherent nonlinear behaviours in queueing systems. Switching costs in real-world queueing systems are another factor that is complicated to model. There also exist analytical results in applications of queueing theory that could be adapted to investigate the properties of our model in future work (see Fischer and Meier-Hellstern (1992); Grassman (2003); Meier-Hellstern (1987); Nunez-Queija (1998); Rossiter (1987), and Iravani et al. (2007)). Finally, we would like to scale up to larger systems, with more agents and job classes. Shafer et al. (2001) consider the impact of worker heterogeneity, which we can study better in systems with multiple agents. All of these topics merit further investigation in this context.

Acknowledgements

The authors would like to thank both the anonymous reviewers and our editor, Professor Angappa Gunasekaran, for their many helpful suggestions.

References

Aksin, Z., Armony, M., and Mehrotra, V. (2007) 'The modern call center: a multi-disciplinary perspective on operations management research', working paper available at http://www.stern.nyu.edu/om/faculty/armony/research/CallCenterSurvey.pdf

Badiru, A. (1992) 'Computational survey of univariate and multivariate learning curve models', *IEEE Transactions on Engineering Management*, Vol. 39, No. 2, pp.176–187.

Baras, J.S., Dorsey, A.J., and Makowski, A.M. (1985) 'Two competing queues with linear costs and geometric service requirements: the mu-c rule is often optimal', *Advances in Applied Probability*, Vol. 17, No. 1, pp.186–209.

Bertsekas, D.P. (1995) *Dynamic Programming and Optimal Control*, Vol. 2, Athena Scientific, Belmont, Massachusetts, USA, pp.242–256.

Bodreau, J., Hopp, W., McClain, J., and Thomas, L. (2003) 'On the interface between operations and human resource management', *Manufacturing and Service Operations Management*, Vol. 5, No. 3, pp.179–202.

Caro, F., and Gallien, J. (2007) 'Dynamic assortment with demand learning for seasonal consumer goods', *Management Science*, Vol. 53, No. 2, pp.276–292.

Cleveland, B., and Mayben, J. (2000) *Call Center Management on Fast Forward: Succeeding in Today's Dynamic Inbound Environment*, Call Center Press, pp.33–40.

Dietrich, B., and Harrison, T. (2006) 'Serving the services: the emerging science of service management opens opportunities for operations research and management science', *OR/MS Today*, June, 2006, http://www.lionhrtpub.com/orms/orms-6-06/frservice.html

Eitzen, G., Panton, D., and Mills, G. (2004) 'Multi-skilled workforce optimisation', *Annals of Operations Research*, Vol. 127, No. 1, pp.359–372.

Fischer, W., and Meier-Hellstern, K. (1992) 'The Markov-modulated Poisson process (MMPP) cookbook', *Performance Evaluation*, Vol. 18, No. 2, pp.149–171.

Gans, N., Koole, G., and Mandelbaum, A. (2003) 'Telephone call centers: tutorial, review, and research prospects', *Manufacturing and Service Operations Management*, Vol. 5, No. 2, pp.79–141.

Gans, N., and Zhou, Y. (2002) 'Managing learning and turnover in employee staffing', *Operations Research*, Vol. 50, No. 6, pp.991–1006.

George, J. and Harrison, J. (2001) 'Dynamic control of a queue with adjustable service rate', *Operations Research*, Vol. 49, No. 5, pp.720–731.

Grassman, W. (2003) 'The use of eigenvalues for finding equilibrium probabilities of certain Markovian two-dimensional queueing problems', *INFORMS Journal on Computing*, Vol. 15, No. 4, pp.412–421.

Gross, D., and Harris, C. (1998) *Fundamentals of Queueing Theory*, third edition, Wiley-Interscience, New York, page 149.

Hampshire, R., Harchol-Balter, M., and Massey, W. (2006) 'Fluid and diffusion limits for transient sojourn times of processor sharing queues with time varying rates', *QUESTA Special Issue on Queueing Models for Fair Resource Sharing*, Vol. 53, No. 1-2, pp.19–30.

Harrison, J. (1975) 'A priority queue with discounted linear costs', *Operations Research*, Vol. 23, No. 2, pp.270–282.

Hasija, S., Pinker, E.J., and Shumsky, R.A. (2005) 'Staffing and routing in a two-tier call center', *Int. J. Operational Research*, Vol. 1, No. 1/2, pp.8-29.

Iravani, S., Kolfal, B., and Oyen, M. (2007) 'Call-center labor cross-training: it's a small world after all', *Management Science*, Vol. 53, No. 7, pp.1102–1112.

Koole, G. (1997) 'Assigning a single server to inhomogeneous queues with switching costs', *Theoretical Computer Science*, Vol. 182, No. 1, pp.377–332.

Mandelbaum, A., and Stolyar, A. (2004) 'Scheduling flexible servers with convex delay costs: heavy traffic optimality of the generalized c-mu rule', *Operations Research*, Vol. 52, No. 6, pp.836–855.

Mazzola, J. and McCardle, K. (1997) 'The stochastic learning curve: optimal production in the presence of learning-curve uncertainty', *Operations Research*,Vol. 45, No. 3, pp.440–450.

Meier-Hellstern, K. (1987) 'A fitting algorithm for Markov-modulated Poisson processes having two arrival rates',*European Journal of Operational Research*, Vol. 29, No. 3, pp.370–377.

Misra, S., Pinker, E.J., and Shumsky, R.A. (2004) 'Salesforce design with experience-based learning', *IIE Transactions*, Vol. 36, No. 10, pp. 941–952.

Nembhard, D.A., and Osothsilp, N. (2001) 'An empirical comparison of forgetting models', *IEEE Transactions on Engineering Management*, Vol. 48, No. 3, pp. 283–291.

Nunez-Queija, R. (1998) 'A queueing model with varying service rate for ABR', *Proceedings of the Tenth International Conference on Computer Performance Evaluation (Tools)*, pp.93–104.

Pinker, E.J., and Shumsky, R.A. (2000) 'The efficiency-quality trade-off of cross-trained workers', *Manufacturing and Service Operations Management*, Vol. 2, No. 1, pp.32–48.

Reis, A. (1991) 'Learning curve in food services', *Journal of the Operational Research Society*, Vol. 42, No. 8, pp.623–629.

Ryokov, V., and Lembert, E. (1967) 'Optimal dynamic priorities in single-line queueing systems',*Engineering Cybernetics*, Vol. 5, No. 1, pp.21–30.

Rossiter, M. (1987) 'A switched Poisson model for data traffic', *Australian Telecommunication Research*, Vol. 21, No. 1, pp53–57.

Sayin, S., and Karabati, S. (2007) 'Assigning cross-trained workers to departments: a two-stage optimization model to maximize utility and skill improvement', *European Journal of Operational Research*, Vol. 176, No. 3, pp.1643–1658.

Schilling, M., Vidal, P., Ployhart, R., and Marangoni, A. (2003) 'Learning by doing something else: variation, relatedness, and the learning curve', *Management Science*, Vol. 49, No. 1, pp.39–56.

Schrage, L. and Miller, L. (1966) 'The queue M/G/1 with the shortest remaining processing time discipline', *Operations Research*, Vol. 14, No. 4, pp.670–684.

Shafer, S., Nembhard, D., and Uzumeri, M. (2001) 'The effects of worker learning, forgetting, and heterogeneity on assembly line productivity', *Management Science*, Vol. 47, No. 12, pp.1639–1653.

Tezcan, T. (2006) *State Space Collapse in Many-Server Diffusion Limits of Parallel Server Systems and Applications*, Ph.D. Thesis, Georgia Institute of Technology, August, 2006, pp.55–59.

Tirtiroglu, E. (2005) 'An entropy measure of operating performance uncertainty in queues: Markovian examples', *Int. J. Operational Research*, Vol. 1, Nos. 1/2, pp.204–212.

Thompson, P. (2007) 'How much did the Liberty shipbuilders forget?', *Management Science*, Vol. 53, No. 6, pp.908–918.

Tucker, A., Nembhard, I., and Edmondson, A. (2007) 'Implementing new practices: an empirical study of organizational learning in hospital intensive care units', *Management Science*, Vol. 53, No. 6, pp.894–907.

Wallace, R., and Whitt, W. (2005) 'A staffing algorithm for call centers with skill-based routing', *Manufacturing and Service Operations Management*, Vol. 7, No. 4, pp.276–294.

Whitt, W. (2006) 'The impact of increased employee retention on performance in a customer contact center', *Manufacturing and Service Operations Management*, Vol. 8, No. 3, pp.235–252.

Yuen, G. (2006) 'Operations systems with discretionary task completion', *Manufacturing and Service Operations Management*, Vol. 8, No. 1, pp.98–117.

Zamiska, J., Jaber, M., and Kher, H. (2007) 'Worker deployment in dual resource constrained systems with a task type factor', *European Journal of Operational Research*, Vol. 177, No. 3, pp.1507–1519.

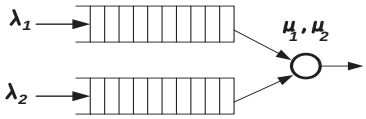

Figure 1 The optimal service policy is sought for the one server, two queue case in the presence of stochastic learning (increasing service rate) and forgetting (decreasing service rate).

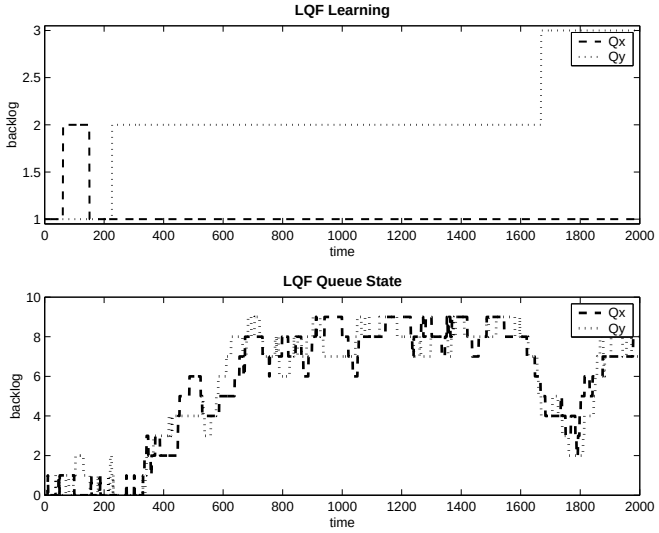

Figure 2 Time series sample for the stationary policy *LQF*, or "serve the Longest Queue First." Note that the *LQF* rule does not respond to changes in the service rate.

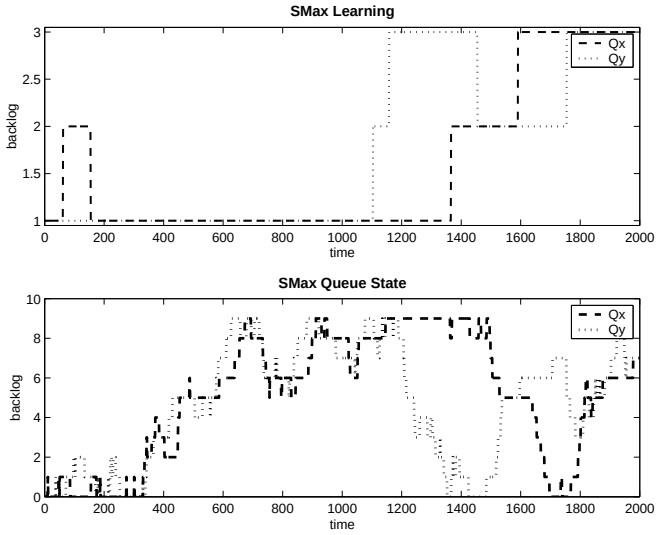

Figure 3 Time series sample for the stationary policy *SMax*, or "serve the job class for which the agent is fastest." Here long stretches of time can occur where the other job class is not served.

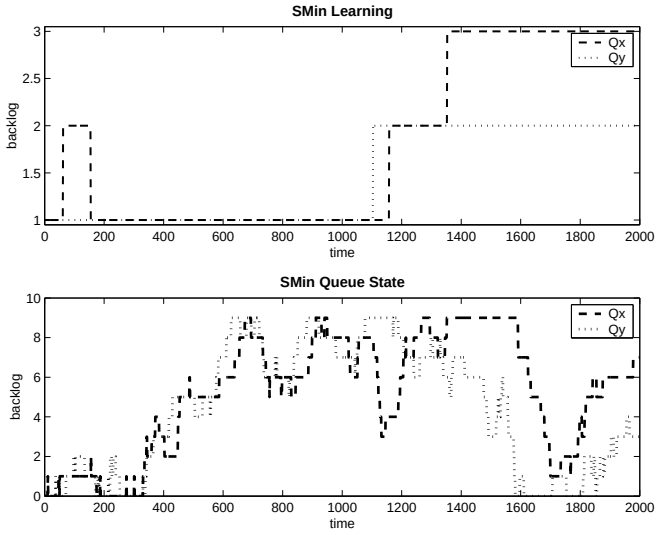

Figure 4 Time series sample for the stationary policy *SMin*, or "serve the job class for which the agent is slowest." It is the reverse of *SMax*. This policy accrues a higher cost than others, but maximises the average capacity of the system.

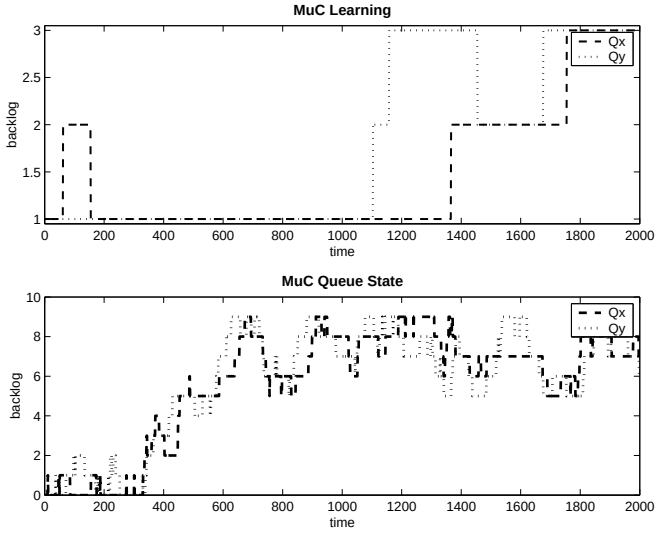

Figure 5 Time series sample for the stationary policy *MuC*, or the μc rule: serve the job class for which the product of the queue length and service rate is a maximum. It is similar to but more flexible than *LQF* in permitting some jobs to accumulate in one queue while the other is served.

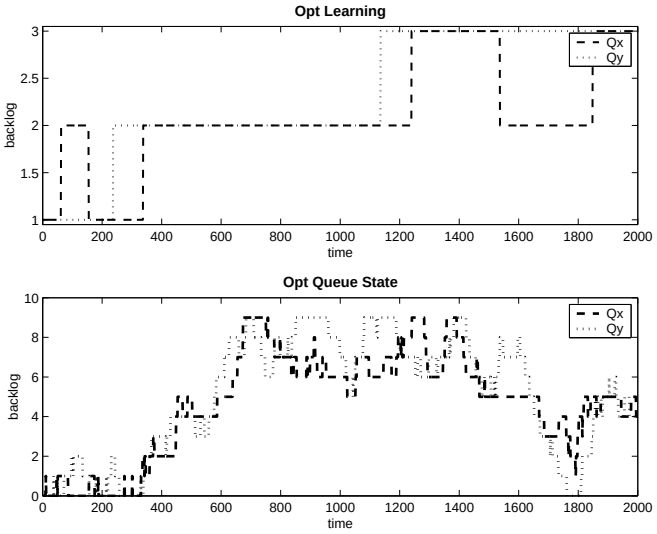

Figure 6 Representative time series sample for the optimal policy, as computed by the dynamic program using a combination of value and policy iteration.

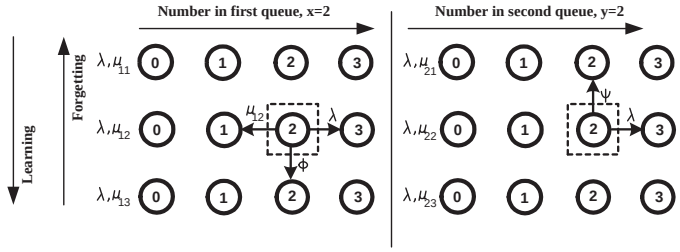

Figure 7 Joint model state $J(q_x, q_y, \ell_x, \ell_y, u_x)$, with $q_x = 2$, $q_y = 2$, $\ell_x = 2$, and $\ell_y = 2$. The probabilistic transitions under the policy "serve queue one", u_x, are shown. Learning, service, and an arrival are possible for queue x, while forgetting and an arrival are possible for queue y. Self-transitions at each queue are also possible. The activities of the two queues are considered to be independent.

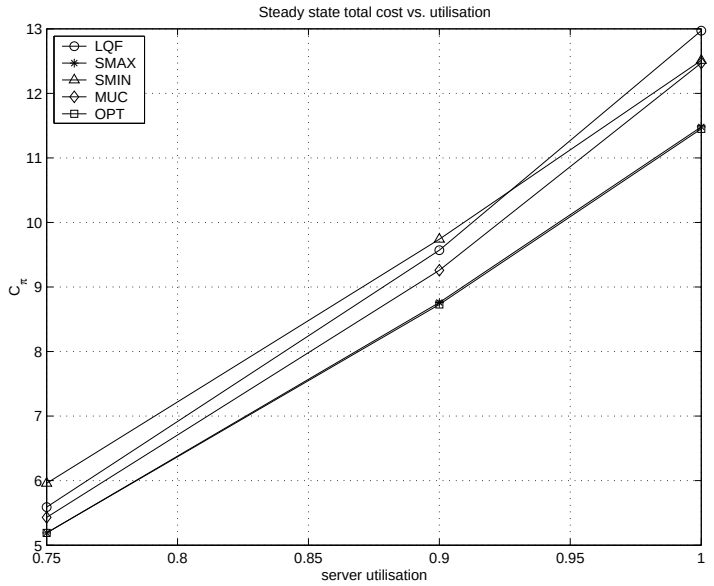

Figure 8 Plot of the steady state total cost $C_{\vec{\pi}}$ values from Table 3 for all five policies. See Section 3.5 for the definition of $C_{\vec{\pi}}$.

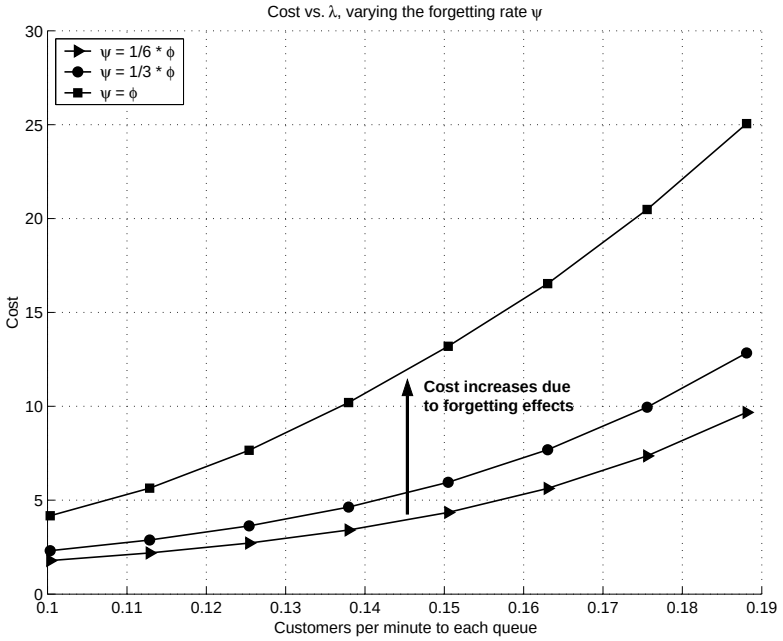

Figure 9 Plot of the combined steady state cost $C_{\vec{\pi}}$ vs. the arrival rate into the system, using the optimal dynamic policy *Opt*, while varying the forgetting rate ψ. Increased forgetting increases the holding costs in the queue through lower average service rates. The learning rate for all cases is set to *high*.

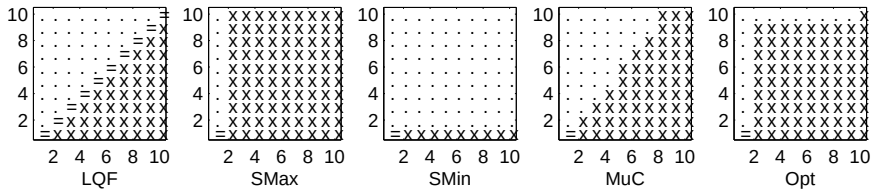

Figure 10 Side-by-side comparison of policies by observing queue service rules for the same subset of states. Each of the five squares represents the state of queue x on the horizontal and queue y on the vertical axis. The policy in question serves x if an x appears, y if a dot appears, and serves either if an equals sign appears. All five plots show the same unbalanced learning curve state: $\mu_x = 1.8m$, $\mu_y = 1.4m$, and x has a service rate advantage. Here *Opt* behaves like *SMax*, except at the border where *Opt* reacts to the overflow penalty. For balanced learning curve states with $\mu_x = \mu_y$ (not shown), all policies look like *LQF*.

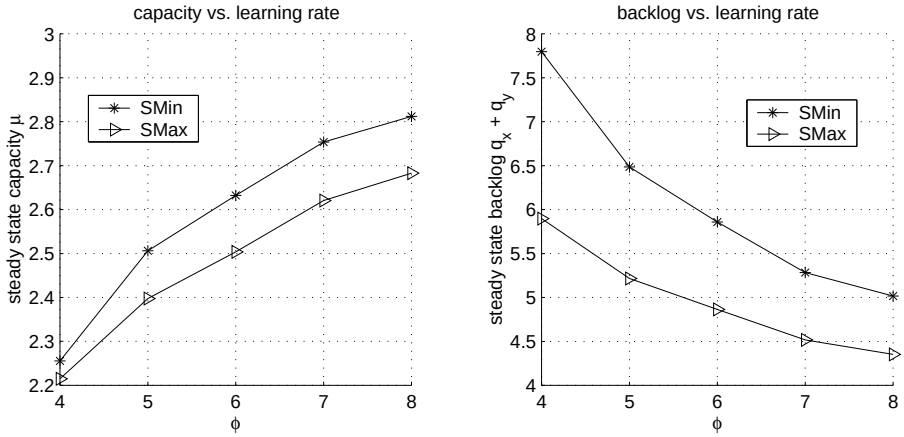

Figure 11 Left: plot of long run system capacity $\mu_{\vec{\pi}}$ (note that $\mu_{x\vec{\pi}} = \mu_{y\vec{\pi}}$) at 100% utilisation, showing a metric for which *SMin* is superior to *SMax*. Right: plot of long run system backlog ($q_{x\vec{\pi}} + q_{y\vec{\pi}}$) at 100% utilisation, showing a metric for which *SMax* is superior to *SMin*. Learning is swept from *high* to *very high* rates. Forgetting is held constant at $\psi = \phi_{vh}/6$, where ϕ_{vh} is the *very high* learning rate.