Scheduling for single server queues. Variant of the μc-rule.

Zita Fernandes

Abstract

The goal of the work described in this paper concerns the presentation of arguments that bring a new perspective to some existing results of queuing networks. Comparison between numerical data obtained with models that use only buffer length as state variable with models that use also server state show that the models which only use buffer length do not correctly represent the systems under study. Also a new scheduling policy is presented. It is motivated by the belief that the marginal cost of waiting should not be a constant and by the belief that strict priorities are only interesting to customers having higher priority. With this new policy we propose to replace strict priorities, as is the case of the $\mu c-$rule. The variance of the cycle time for all classes achieved with the new policy will be presented, as is numerically computed. The tools used to get the results presented in this work, such as Dynamic Programming and Discrete Event Simulation, are also described.

Keywords: Queuing networks, Scheduling problem, queuing networks modeling, state costs, variance of cycle time.

1 Introduction

Looking at several approaches to model queuing networks to address the scheduling problem, one verifies that the majority were modeled using only the queue length as state variable. Such is the case of [5, 9, 8, 4, 1], where the authors formulate and solve an MDP through Dynamic Programming, and formally establish optimality of the resulting policy. Analyzing those models, we conclude that they do not represent the networks correctly. In this paper, we consider that the state of the servers of the network should also be taken into account.

To validate this hypothesis we modeled a simple queuing network with two alternative state representations: using only the queue lengths; and using the queue lengths and the server state. To verify which of the two models is the correct one, we computed state costs for each and then compared those to the results obtained by simulating the network, through a discrete event simulator. To compute optimal costs and policies, two Markov Decision Problems, MDP, are formulated and solved using Dynamic Programming. To simulate the networks, a software package using object oriented language was developed.

In order to provide an alternative to strict priority policies, we also present a new policy that tries to be more equitable in server access to the several classes of customers entering the network.

A new cost formulation, which is a generalization of the cost function for which the well known μc-rule is optimal, was developed. The μc-rule is such that the class which has the highest score given by $\mu_i c_i$, gets priority over the other classes. The processing rate for class i is given by μ_i and c_i is the cost rate parameter, expressing the amount being payed, while in the system, by each costumer of the same class. For the classic scheduling problem the cost rate is given by a simple linear parameter, but in our new function it is given by a linear and a quadratic term. To evaluate the performance of the policy derived for this new cost function, we computed the variance of the cycle time for the individual classes, comparing it with the same performance measure achieved under the μc-rule.

Finally, we also show that the new policy gets a lower global variance of the total cycle time than the μc-rule.

In the remaining of the paper we present the alternative models in Section 2, formulate the MDP associated with each, and present a sample of the numerical results supporting our claim. Next in Section 3, we present the policy that results from the alternative cost function, providing an intuitive explanation of its structure, match examples of the numerically computed policy with the policy intuitively derived, and show, through numerical examples, the effect of this policy on the variance of the cycle times. We conclude, in Section 4, discussing some consequences of our results and pointing directions for further research.

2 Queuing Networks Modeling

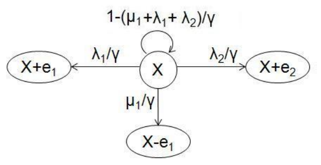
We start by addressing the issue of what is the adequate procedure to model the scheduling problem for single server queues. We formulate the classic scheduling problem, model the network using only the buffer lengths, as the majority of the authors, and model it with the alternative representation, where the server state is also included.

2.1 Problem Formulation

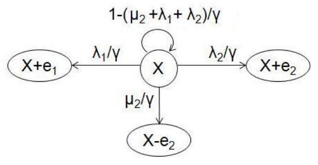
The network used to study the modeling problem is composed by 2 buffers with Poisson arrivals (with rate λ_i for queue i) and one single server with exponential service times (with rate μ_i for class i). Given the fact that we assume linear cost rates for the buffer lengths the μc-rule is the optimal scheduling policy. We also consider that the service is non-preemptive and non idling. Due to the characteristics of the problem, we may build a model of the network as a continuous time Markov chain and then make use of Lippman's uniformization procedure, [12], to obtain the discrete time counterpart.

We represent the queue lengths by $X(t) = [x_1(t)\ x_2(t)]' \in N^2$, with $x_i(t)$ representing the amount of customers of class i, $i = 1, 2$, in the system at time t. For the server state we use $y(t) \in \{0, 1, 2\}$, where $y(t) = 0$ means that, at time t, the server is idle, $y(t) = i$, $i = 1, 2$, the server is processing a customer of class i. The first model will only use $X(t)$ for state representation and the second model will use $[X(t)\ y(t)]$. The first will be designated as the One Variable Model and the second will be the Two Variable Model.

On both models four events are possible: end of service of a customer of class 1, end of service of a customer of class 2, arrival of a customer of class 1, and arrival of a customer of class 2. The resulting MDP's for both models, after the uniformization, are displayed in Figs. 1 and 2



(a) Diagram 1.



(b) Diagram 2.

Figure 1: Uniformized transition diagram using one variable.

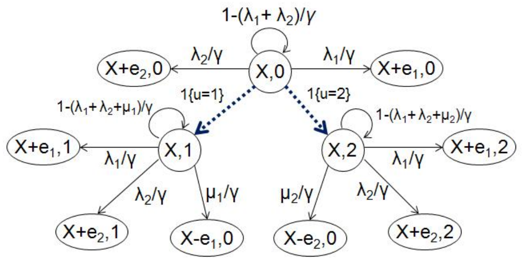


Figure 2: Uniformized transition diagram using two variables.

On both diagrams, γ represents the uniform rate and is such that $\gamma = \mu_1 + \mu_2 + \lambda_1 + \lambda_2$. Still related to the diagrams, it is possible to define two kinds of transitions. Transitions that are controllable and transitions that are uncontrollable. The transitions that are controllable are the conclusion of service and the uncontrollable are the arrival of customers. When a end of service takes place, we can also say that we have a decision point. That is, is it necessary to chose which of the classes to serve. Given we consider the service to be non-preemptive, when the arrival of a priority customer occurs during the service of a non-priority customer, there is no decision point. Also, an arrival generates a decision point only when it occurs when the system is empty of customers.

In any of the two models, a decision point will present three options: remain idle; process a customer of class 1, $u = 1$; and process a customer of class 2, $u = 2$. These last two options, if chose, will enable controllable transitions. Although it is known that the classic scheduling problem produces a non idling policy, for the benefit of completion, we explicitly present the option of staying idle.

The cost criterion used is based on the total expected discounted cost over infinite horizon with discount factor β, as follows

$$V_\pi(X) = \mathrm{E}_\pi \int_0^\infty e^{-\beta t}\left[c_1 x_1(t) + c_2 x_2(t)\right] dt, \quad (1)$$

where the expected value is taken over all possible trajectories of the system, given the fact that the Markov chain is a stochastic process, and some policy π is being used, such that for each transition point some mapping exists between state and decision.

The optimal cost is obtained over all possible policies and is expressed as

$$V(X) = \min_\pi \mathrm{E}_\pi \int_0^\infty e^{-\beta t}\left[c_1 x_1(t) + c_2 x_2(t)\right] dt. \quad (2)$$

We know, [6], that the policy that minimizes the above is the μc-rule. That is, at every decision point where there are customers in both queues, the server will chose to process a customer from the buffer for which the product $\mu_i c_i$ is the highest. Since, μ_i and c_i are assumed to be constant parameters during the control horizon, this means that there will be a class that has always higher priority over the other, except for the trivial case where the above product is the same for both classes.

With all the information above, it is possible to solve the MDP using Dynamic Programming, DP. The Value Iteration algorithm [2, 3] was used to compute the optimal costs according to the specified performance criterion. This was done in order to compare the costs obtained through DP for the different models and not to obtain the optimal policy, because its structure is known upfront.

Looking at the model using only one variable, the DP recursions to apply the value iteration algorithm are given by:

$$\begin{aligned} &V_{k+1}(X) = \tfrac{1}{\beta+\gamma}(c_1x_1 + c_2x_2) + \\ &+\alpha \min\left\{\tilde{V}(X, u|u=0), \tilde{V}(X, u|u=1), \tilde{V}(X, u|u=2)\right\}, \end{aligned} \quad (3)$$

with

$$\begin{aligned} &\tilde{V}(X, u|u=0) = \tfrac{\lambda_1}{\gamma}V_k(x_1+1, x_2) + \tfrac{\lambda_2}{\gamma}V_k(x_1, x_2+1) + \\ &+\left(1 - \tfrac{\lambda_1+\lambda_2}{\gamma}\right)V_k(x_1, x_2), \\ &\tilde{V}(X, u|u=1) = \tfrac{\lambda_1}{\gamma}V_k(x_1+1, x_2) + \tfrac{\lambda_2}{\gamma}V_k(x_1, x_2+1) + \\ &+\tfrac{\mu_1}{\gamma}V_k(x_1-1, x_2) + \left(1 - \tfrac{\lambda_1+\lambda_2+\mu_1}{\gamma}\right)V_k(x_1, x_2), \\ &\tilde{V}(X, u|u=2) = \tfrac{\lambda_1}{\gamma}V_k(x_1+1, x_2) + \tfrac{\lambda_2}{\gamma}V_k(x_1, x_2+1) + \\ &+\tfrac{\mu_2}{\gamma}V_k(x_1, x_2-1) + \left(1 - \tfrac{\lambda_1+\lambda_2+\mu_2}{\gamma}\right)V_k(x_1, x_2). \end{aligned} \quad (4)$$

We omit the details concerning transforming equation 2 into the recursions presented in 3 and 4. The interested reader may find this in [2] for the general case and in [7] for the specific problem being addressed here.

The algorithm stops when the error associated to the iterative process is lower than some ϵ, defined by the user, that is

$$|V_{k+1} - V_k| < \epsilon.$$

Not all the terms of expression 4 are active in all decision points. In some cases, such as states were there are no customers of class 2, only the first two terms correspond to admissible options.

For the model using two variables, the value iteration recursions are given by

$$\begin{aligned} &V_{k+1}(X, 0) = \tfrac{1}{\beta+\gamma}(c_1x_1 + c_2x_2) + \\ &+\alpha \min\left\{\tilde{V}(X, 0, u|u=0), \tilde{V}(X, 0, u|u=1), \tilde{V}(X, 0, u|u=2)\right\}, \end{aligned} \quad (5)$$

with,

$$\begin{aligned} &\tilde{V}(X, 0, u|u=0) = \tfrac{\lambda_1}{\gamma}V_k(x_1+1, x_2, 0) + \tfrac{\lambda_2}{\gamma}V_k(x_1, x_2+1, 0) + \\ &\left(1 - \tfrac{\lambda_1+\lambda_2}{\gamma}\right)V_k(x_1, x_2, 0), \\ &\tilde{V}(X, 0, u|u=1) = \tfrac{\lambda_1}{\gamma}V_k(x_1+1, x_2, 1) + \tfrac{\lambda_2}{\gamma}V_k(x_1, x_2+1, 1) + \\ &+\tfrac{\mu_1}{\gamma}V_k(x_1-1, x_2, 0) + \left(1 - \tfrac{\lambda_1+\lambda_2+\mu_1}{\gamma}\right)V_k(x_1, x_2, 1), \\ &\tilde{V}(X, 0, u|u=2) = \tfrac{\lambda_1}{\gamma}V_k(x_1+1, x_2, 2) + \tfrac{\lambda_2}{\gamma}V_k(x_1, x_2+1, 2) + \\ &+\tfrac{\mu_2}{\gamma}V_k(x_1, x_2-1, 0) + \left(1 - \tfrac{\lambda_1+\lambda_2+\mu_2}{\gamma}\right)V_k(x_1, x_2, 2). \end{aligned} \quad (6)$$

For states where the server is busy, that is, a decision has been made at an earlier transition, the recursions assume the following form

$$\begin{aligned} &V_{k+1}(X, 1) = \tfrac{1}{\beta+\gamma}(c_1x_1 + c_2x_2) + \\ &\alpha\left\{\tfrac{\lambda_1}{\gamma}V_k(x_1+1, x_2, 1) + \tfrac{\lambda_2}{\gamma}V_k(x_1, x_2+1, 1) + \right. \\ &\left. + \tfrac{\mu_1}{\gamma}V_k(x_1-1, x_2, 0) + \left(1 - \tfrac{\lambda_1+\lambda_2+\mu_1}{\gamma}\right)V_k(x_1, x_2, 1)\right\}, \end{aligned} \quad (7)$$

$$\begin{aligned} &V_{k+1}(X, 2) = \tfrac{1}{\beta+\gamma}(c_1x_1 + c_2x_2) + \\ &\alpha\left\{\tfrac{\lambda_1}{\gamma}V_k(x_1+1, x_2, 2) + \tfrac{\lambda_2}{\gamma}V_k(x_1, x_2+1, 2) + \right. \\ &\left. + \tfrac{\mu_2}{\gamma}V_k(x_1-1, x_2, 0) + \left(1 - \tfrac{\lambda_1+\lambda_2+\mu_2}{\gamma}\right)V_k(x_1, x_2, 2)\right\}. \end{aligned} \quad (8)$$

Like in the model using one variable, expression 5 has terms that may not correspond to admissible options for some states.

At this point we are able to see some differences between the two models. Although the events that can occur in the two models are the same, we can verify that different states of the model using two variables are represented by the same state in the model using only one variable. For example, the states $\{3, 2, 1\}$ and $\{3, 2, 2\}$ are represented as the same state $\{3, 2\}$ in the model using one variable. In the model using one variable it is impossible to know if the server is processing a customer from class 1 or class 2. In the model using two variables, that doubt does not exist.

The next step is to compute the costs by simulation. For this purpose, we have developed a package in JAVA that allows the simulation of queuing networks and activity networks, [10, 11]. During the simulation, an output file is created. This output file contains the arrivals times, the end of service times and the number of customers of each class in the system when one of the two previous events occur. Processing this information using Matlab, it is possible to get the costs under the optimal policy.

For each group of parameters we ran 25 simulations, each one with a different seed and each simulation corresponding to 1000 regeneration intervals. One defines a regeneration point as the instant of time where a customer leaves the system empty upon his/her departure.

2.2 Results

The figures 3 and 4 present the results computed using DP and by the simulator. In the case of the simulation results, the 95% interval of confidence has been computed. We computed state costs using the following parameters:

$$\begin{cases} \lambda_1 = 3, \mu_1 = 4, c_1 = 2, \\ \lambda_2 = 0.666666, \mu_2 = 0.1, c_2 = 1 \\ \beta = 0.002. \end{cases}$$

As it can be verified, almost all results obtained with the model of two variables fit in the confidence interval. The few cases where that doesn't happen are not due to an error in the costs computed using DP but by the cost computed using the simulation's output. A way of reducing this error is to increase the number of simulations for each set of parameters. The costs computed using the model of one variable are significantly lower than the other costs. From these results, it is possible to conclude that the model using two variables is the most correct one to model networks of this kind.

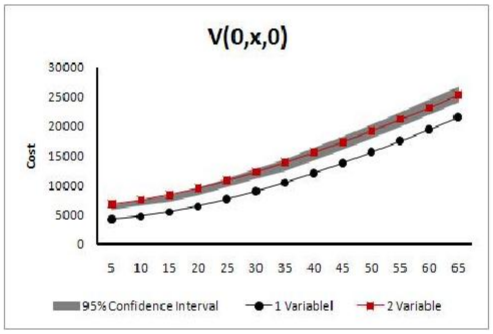


Figure 3: Cost as a function of x_2.

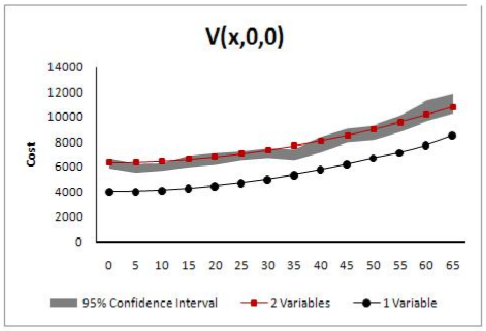


Figure 4: Cost as a function of x_1.

3 The $\mu\Delta$-Rule

In this section we address the problem of parting with strict priorities using, alternatively, smooth priorities. That is, policies where the lower priority classes may be chosen in the presence of higher priority classes as a function of the overall system state.

3.1 Problem Formulation

Taking into account the results of the previous section, this problem is formulated using a model with two variables. The network used in this problem is the same used above and we are still looking at infinite horizon discounted costs.

The difference now concerns with the expression that expresses the cost as a function of the buffer lengths.

For the μc-rule, the costs are given by $c_i x_i$. In this problem the state cost is given by $a_i x_i + b_i x_i^2$, that is, the state cost is given by a linear and a quadratic term. Using this expression in the DP algorithm we will obtain the decision matrices that allow us to identify the new policy. The decision matrices are obtained by choosing the action that minimizes the cost state upon convergence of the value iteration algorithm. Depending on the state, in the respective position of the matrix it will appear a 0 if idleness is the best choice, 1 if processing customers of class 1 is the best choice, or 2 otherwise.

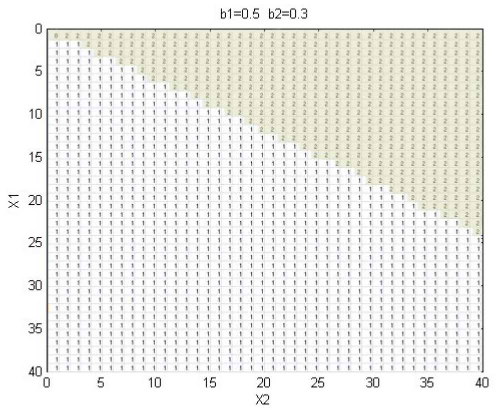
We omit the details of the recursions for this particular problem and move on to the presentation of the results obtained with the Value Iteration algorithm.

3.2 Results

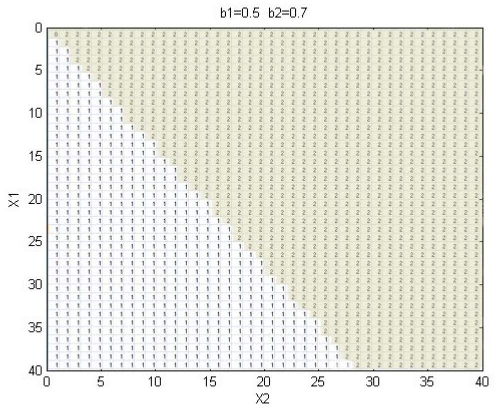
To illustrate the structure of the optimal policy we present a sample of the results, where the linear cost parameters are set to zero and only the quadratic terms are non zero. In particular, we set b_1 to be constant and evaluate how does the policy change with a change in b_2.

$$\begin{cases} \lambda_1 = 0.475, \mu_1 = 1, a_1 = 0, b_1 = 0.5 \\ \lambda_2 = 0.475, \mu_2 = 1, a_2 = 0, b_2 \in \{0.3, 0.7\} \\ \beta = 0.002 \end{cases}$$

In Figure 5 we present the control matrices obtained for decision points as functions of x_1 and x_2. Matrix a) is obtained for $b_2 = 0.3$ and matrix b) corresponds to $b_2 = 0.7$. One should note that if b_2 were to be set to zero, given the fact that b_1 is non zero, the resulting policy would be the μc-rule with priority to class 1. Therefore, when b_2 is non zero but lower than b_1 – Fig. 5 a) –, we observe that there is region of the state space where customers of class 2 get priority over customers of class 1. Moreover, the state space is divided in two regions,where each class has higher priority in each region. Also, there is a straight line separating them. Reporting to matrix b), a similar behavior is observed with the classes switching their roles, since $b_2 > b_1$ for this case. We have observed the switching curve to be a straight line for all other choices of the cost parameters we tried.



(a)



(b)

Figure 5: Sample of decision matrices.

Given the fact that the curve separating the two regions is a straight line, we can conjecture that it should be possible to formally establish the structure of the optimal policy for any set of cost parameters. What we propose next is to derive its structure using intuitive arguments and making an analogy with what happens for the μc-rule.

3.3 Intuitive Formulation of the Policy

Considering an iteration on the process, we have the following for the μc-rule: whenever both queues are non empty at a decision point the choice is made as if looking to what class will reduce more cost at a higher rate if chosen to be processed. That is,

$$\begin{cases} \mu_1 \left(a_1 x_1 - a_1 \left(x_1 - 1\right)\right) = \mu_1 a_1 = \mu_1 C_1 \\ \mu_2 \left(a_2 x_2 - a_2 \left(x_2 - 1\right)\right) = \mu_2 a_2 = \mu_2 C_2 \end{cases}$$

were C_i are the values of the cost reduction due to decreasing the size of queue i by one.

We know the μc-rule to give higher priority according to the product $\mu_i C_i$. That is, the highest priority goes to the class for which the product between the linear cost rate and processing rate is higher.

By analogy with the μc-rule, we now repeat the argument using also the quadratic component for the cost:

$$\begin{cases} \mu_1 \left[a_1 x_1 + b_1 x_1^2\right] - \mu_1 \left[a_1 \left(x_1 - 1\right) + b_1 \left(x_1 - 1\right)^2\right] = \\ = \mu_1 \left[a_1 + 2x_1 b_1 - b_1\right] = \mu_1 \Delta_1 \\ \mu_2 \left[a_2 x_2 + b_2 x_2^2\right] - \mu_2 \left[a_2 (x_2 - 1) + b_2 (x_2 - 1)^2\right] = \\ = \mu_2 \left[a_2 + 2x_2 b_2 - b_2\right] = \mu_2 \Delta_2 \end{cases}$$

In this case, the incurred cost reduction due to decreasing each queue size by one is Δ_i as given above. Therefore, it appears that higher priority should be given to the class for which $\mu_i \Delta_i$ is higher. If this is the case, note that there is no single class which always has higher priority than the other, because Δ_i depends on the queue size x_i.

$$\begin{aligned} &\textit{if } \mu_1 \Delta_1(t) > \mu_2 \Delta_2(t) \\ &y(t) = 1 \\ &\textit{if } \mu_1 \Delta_1(t) < \mu_2 \Delta_2(t) \\ &y(t) = 2 \end{aligned}$$

Assuming our reasoning to be correct, we can derive an expression for the switching curve, which is done by identifying the states for which $\mu_1 \Delta_1 = \mu_2 \Delta_2$.

$$\mu_1 \left[a_1 + 2x_1 b_1 - b_1\right] = \mu_2 \left[a_2 + 2x_2 b_2 - b_2\right]$$

$$\vdots$$

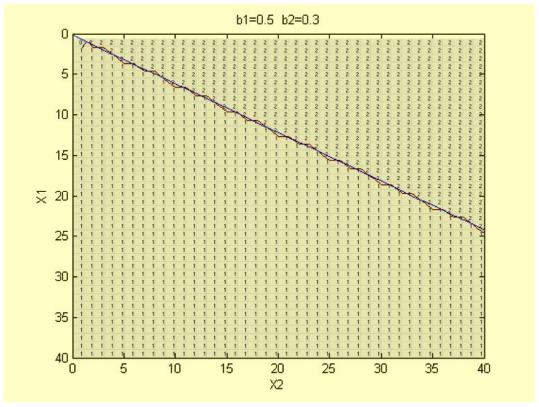
$$x_2 = \frac{(a_1\mu_1 - a_2\mu_2) + (b_2\mu_2 - b_1\mu_1)}{2b_2\mu_2} + \frac{b_1\mu_1}{b_2\mu_2} x_1, \quad (9)$$

The expression above is only valid for $b_i \neq 0$ and corresponds to the threshold that defines when to change priority, and it is indeed a straight line. For $b_1 = 0$ and $b_2 \neq 0$ we get

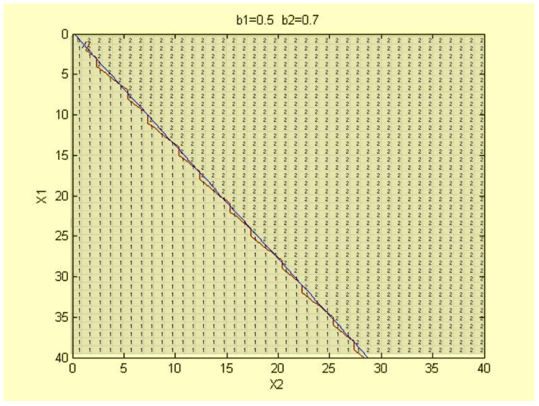
$$x_2 = \frac{(a_1\mu_1 - a_2\mu_2) + b_2\mu_2}{2b_2\mu_2}, \quad (10)$$

and for $b_1 \neq 0$ and $b_2 = 0$ we get

$$x_1 = \frac{(a_2\mu_2 - a_1\mu_1) + b_1\mu_1}{2b_1\mu_1}, \quad (11)$$



(a)



(b)

Figure 6: Decision matrices with switching curve superimposed

meaning that for those cases we either have a straight line with zero slope or infinite slope, which is the same as saying that when only one quadratic term is present, we have a threshold policy. For instance, when $b_1 = 0$, for every $b_2 \neq 0$ there is a value, say k, above which class 2 gets priority over class 1. In this case, we can associate with the threshold the concept of waiting time for service. That is, when the non priority buffer has more than k customers, it means that the first customer of that class has been waiting for service for over k inter-arrival intervals. The purpose is to prevent customers to wait too long for service. The magnitude of b_2 will establish how much is acceptable.

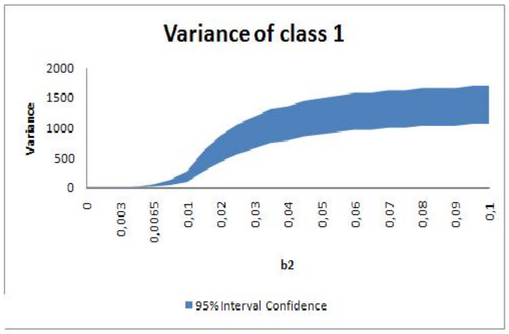
Figure 6 shows that the previous formulation is correct, since the numerical results and the threshold overlap.

3.4 Variance of Cycle Time

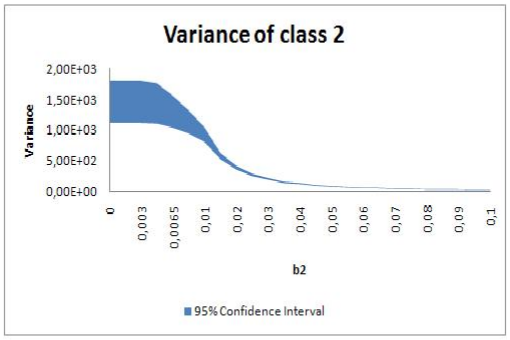
Now we need to address the consequences this new cost function and respective policy have concerning the variance of the cycle time for the two classes. We simulated the network with the following parameters:

$$\begin{cases} \lambda_1 = 0.475, \mu_1 = 1, a_1 = 2, b_1 = 0 \\ \lambda_2 = 0.475, \mu_2 = 1, a_2 = 1, b_2 \in [0, 0.1] \\ \beta = 0.002 \end{cases}$$

For this purpose, we simulated the network with linear and quadratic terms to see the effect of increasing the quadratic term for the non-priority class. Note that when $b_2 = 0$ we have the original classic scheduling problem and, with these parameters, priority should be given to class 1.



(a)



(b)

Figure 7: 95% confidence intervals for cycle time variance of class 1 and 2.

Although it does not make any sense to talk about priority and non priority classes for this new policy, we still make such usage to make our discussion clearer, in terms of comparing the behavior relative to the original μc-rule.

As it is possible to see in figure 7, as we increase the value of b_2 (quadratic term of the non-priority class), the variance of the cycle time for the priority class – class 1, originally – increases and the variance of the cycle time of the non-priority class decreases. These results should not be surprising, given the discussion made so far because, by changing the quadratic term, we can increase the chances for the non-priority class to have access to the server in the presence of the priority class.

Having looked at the variance of the cycle time for each class separately, we can see what happens to the global variance.

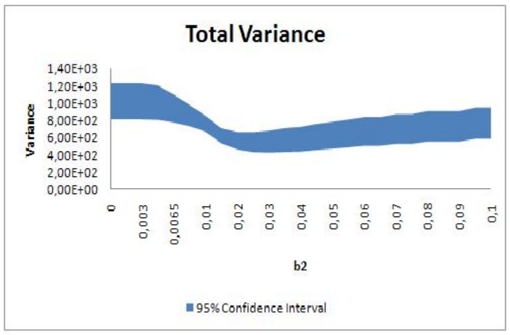


Figure 8: 95% confidence intervals for total cycle time variance.

As can be seen in figure 8 there is a set of parameters that allow the resulting policy to achieve a lower global variance of the cycle time. The value achieved by the μc-rule corresponds to $b_2 = 0$.

Finally it is possible to see if the variance of the cycle time of the two classes is independent. To verify this, we computed the global variance and the sum of the variance of the two classes and confirmed them not to be equal.

From figure 9, we can conclude that the processes are not independent. This result is easy to understand because the classes are competing for the same server. If a non-priority customer is being served, given the service is non-preemptive, the priority customer has to wait for the completion of that service. So, the access of each class to the

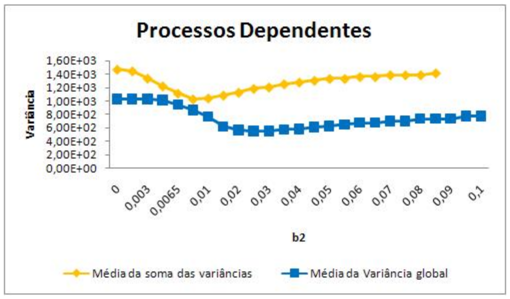


Figure 9: Total cycle time variance and sum of variances for both classes.

server is influenced by the other class.

4. Conclusions

The two main aspects to retain from this work are:

- in queueing networks, the formulation of some problems using models where only the buffer lengths define the state is not correct;
- we formulated a new cost function that produces an alternative to strict priorities policies and decreases the variance of the total cycle time.

Consequences of the first remark are that, although the cost estimates produced by the value iteration algorithm are incorrect for the 1 variable model, the optimal policy is the same in both models. However we conjecture that, for other MDP's constructed for queuing networks, the resulting policies may themselves be incorrect when the state representation does not include the server state.

As to the second remark, we showed that by manipulating the parameters of the new cost function, it is possible to adjust the priorities to both classes in a more balanced way, and we are proposing the concept of smooth priorities as a better alternative to strict priorities.

In terms of performance, we can see that there is a set of parameters that used in the μc-rule and in the new rule gives a lower global variance for the new rule than for the μc-rule, thus improving the departure time predictability for each class of customers.

Future work will have to address the formal demonstration of the $\mu\Delta$-rule optimality.

References

[1] J. S. Baras, A. J. Dorsey, and Makowski. Two competing queues with linear costs: The μc-rule is often optimal. *Advances in Applied Probability*, 17(1):186–209, 1985.

[2] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Volume One*. Athena Scientific, 1995.

[3] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Volume Two*. Athena Scientific, 1995.

[4] C. Buyukkoc, P. Varaiya, and J. Walrand. The μc-rule revisited. *Advances in Applied Probability*, 17(1):237–238, 1985.

[5] A. Cobham. Priority assignment in waiting line problems. *Operations Research*, 2:70–76, 1954.

[6] D. R. Cox and W. L. Smith. *Queues*. Chapman and Hall, 1961.

[7] Z. Fernandes. Sequenciamento em filas de servidor único. Variante da regra μc. Master's thesis, IST, Outubro 2007.

[8] J. M. Harrison. Dynamic scheduling of a multiclass queue: Discount optimality. *Operations Research*, 23(2):270–282, 1975.

[9] J. M. Harrison. A priority queue with discounted linear costs. *Operations Research*, 23(2):260–269, 1975.

[10] J. M. Harrison. Brownian models of open processing networks: Canonical representation of workload. *The Annals of Applied Probability*, 10(1):75–103, 2000.

[11] J. M. Harrison. Stochastic networks and activity analysis. In *Analytic Methods in Applied Probability*, pages 53–76. American Mathematical Society, 2002.

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