Paper Review: Optimal Admission Control in Queues

Mohamed-Amine Azzouz

2024-05-01

Contents

Section 1: Introduction

In Miller's paper, the most general problem of an n-server queuing system is considered, with customers split into m classes, each class representing the rewards $r_1 > \cdots > r_k > \cdots > r_m > 0$ for serving a customer in class $k \in \{1, \ldots, m\}$. Using the assumption that customers of class k arrive at a Poisson rate of λ_k and identically distributed exponential serving times, the paper models the problem as a continuous-time Markov decision process, from which a solution and algorithm is derived for the finite and infinite horizon cases.
For the purpose of this project, the problem will be reduced to 2 queues with respectively M_1 and M_2 servers, with $m = 2$ customer classes, while sticking to a discrete-time case. This approach is explored in Nguyen's paper, which is the one that will be investigated. It will be shown that the optimal policy to maximize the long term reward on the server side is to accept the higher paying customers whenever there are available servers, and accept the lower paying customers whenever the number of busy servers is less than some value, which is to be found and computed.

Section 2: Review of the Poisson Model and Properties of the Threshold Value

The structuring queuing system for this problem consists of:

- M identical servers working in parallel;
- Customers arriving with distribution $X \sim \text{Poi}(\lambda)$, with PDF
$$f_X(n) = \frac{\lambda^n}{n!} e^{-\lambda}$$
- Customers of type 1 arrive with probability p_1, and customers of type 2 arrive with probability $p_2 = 1 - p_1$;
- Zero waiting capacity, so if all servers are busy, the customer is rejected;
- Without loss of generality, the rewards are $r_1 < r_2$;
- Both customer types are known at arrival and require the same service, which is $\exp(\mu)$-distributed, so after the reward is collected (before the service) the customer types look the same.

The main result about the optimal policy, as stated and as will be shown in the paper, is the following:

Proposition 2.1

Trunk reservation is the optimal policy for both the infinite horizon discounted reward and the average reward criteria. More specifically, the policy accepts the higher-paying customer whenever possible, and accepts a lower-paying customer if and only if fewer

than c^* servers are busy at the time of his arrival.

As stated above, let c^* be the threshold number of busy servers to accept a lower-paying customer, and let $l^* := M - c^*$ be the optimal number of servers/trunks to be reserved, it can then be shown with induction that:

Proposition 2.2

l^* is monotonically decreasing in both r_1 and p_1.

This one makes sense, since intuitively one would need to reserve less servers if the reward of the lower paying customer is higher, or if there is a higher proportion of the lower paying customers.

Section 3: Overflow Model Analysis

Next, one must analyze the overflow model, made of two queues. The primary queue is made of:
The structuring queuing system for this problem consists of:

- M_1 identical servers working in parallel;
- Poisson arrival rate of λ_1;
- μ_1 is the rate at which servers from the primary queue dispose of work.

Then, customers who overflow from the primary queue form an arrival stream at the secondary queue:

- M_2 identical servers working in parallel;
- Poisson arrival rate of λ_2;
- μ_2 is the rate at which servers from the primary queue dispose of work.

The service times of all customers are also taken to be exp(1)-distributed. Take overflow customers to be of type 1, and the natural flow input to be of type 2, with rewards r_1 and r_2. The optimal policy will be found for the two cases $r_1 < r_2$ and $r_1 > r_2$.

Dynamic Programming Equations

The overflow system can be modelled as a two-dimensional birth-death process, where one dimension describes the primary queue, and the other models the overflow on the secondary queue. Therefore, the state space is:

$$\mathcal{S} = \{(i,j) : i \in \{0,1\ldots,M_1\},\ j \in \{0,1\ldots,M_2\}\} = \{0,1\ldots,M_1\} \times \{0,1\ldots,M_2\},$$

which depicts the number of servers in each queue.

Similarly as before, both customer types are known at arrival and require the same exponentially distributed service, so after the reward is collected, the customer types still look the same. Now, to come up with the model's Markov Decision Process, one can describe one step time as such. For each queue:

- The customer arrives in queue (contributing to a rate of $\lambda_1 + \lambda_2$?);
- Service is completed in the primary queue (contributing to a rate of $M_1\mu_1 + M_2\mu_2$?);
- Then, a fictitious server is done by the free servers. This all gives us a Poisson rate of $\Lambda = \lambda_1 + \lambda_2 + M_1\mu_1 + M_2\mu_2$.

The policy $\mathcal{R}$ is of the form $\mathcal{R} = \mathcal{R}(\boldsymbol{k}, i, j)$. Since the problem involves an MDP, it can be assumed that the policy is of the form $\mathcal{R} = \{a^k(i,j),\ (i,j) \in \mathcal{S}\}$, where:

$$a^k(i,j) = \begin{cases} 1 & : \text{if customer of type } \boldsymbol{k} \text{ is accepted at state } (i,j), \\ 0 & : \text{ if customer of type } \boldsymbol{k} \text{ is refused at state } (i,j). \end{cases}$$

Given a policy $\mathcal{R}$ the step transition probabilities from state (i,j) to state (i',j') is:

$$p_{(i,j)\to(i',j')}(\mathcal{R}) = \begin{cases} \Lambda^{-1}\mu^1_{(i,j)}(\mathcal{R}) & : \ (i',j') = (i-1,j) \\ \Lambda^{-1}\mu^2_{(i,j)}(\mathcal{R}) & : \ (i',j') = (i,j-1) \\ \Lambda^{-1}\lambda^1_{(i,j)}(\mathcal{R}) & : \ (i',j') = (i+1,j) \\ \Lambda^{-1}\lambda^2_{(i,j)}(\mathcal{R}) & : \ (i',j') = (i,j+1) \\ 1 - \Lambda^{-1}\sum_{k=1}^{2}(\mu^k_{(i,j)}(\mathcal{R}) + \lambda^k_{(i,j)}(\mathcal{R})) = \text{"the rest"} & : \ (i',j') = (i,j) \\ 0 & : \text{ otherwise.} \end{cases}$$

where Λ can be considered to be the unit time and the step transition probabilities are

- Work done by primary queue $\mu^1_{(i,j)} = i\mu_1$;
- Work done by secondary queue $\mu^2_{(i,j)} = j\mu_2$;
- Incoming flow at primary queue $\lambda^1_{(i,j)} = \begin{cases} \lambda_1 & : \ i < M_1 \\ 0 & : \ x < i = M_1 \end{cases}$;
- Incoming flow at secondary queue $\lambda^2_{(i,j)} = \begin{cases} a^2(i,j)\lambda_2 & : \ i < M_1 \\ a^1(i,j)\lambda_1 + a^2(i,j)\lambda_2 & : \ x < i = M_1 \end{cases}$.

Next add a discount factor $\beta \in [0,1]$ over each time step, and define $V_{n,\beta}(i,j)$, which is the maximum expected discounted reward (with discount factor β) that can be earned in n time steps if the system starts with i customers in queue 1, and j customers in queue 2.

For $a \in \{0,1\},\ k \in \{1,2\}$, one can also write $g^k_{n,\beta}(i,j,a) := ar_k + V_{n,\beta}(i,j+a)$. Let $G^k_{n,\beta}(i,j)$ be defined as a comparison function that determines whether to accept or reject a customer of type k at state (i,j) and time step n, where

$$G^1_{n,\beta}(i,j) = \begin{cases} V_{n,\beta}(i+1,j) & : \ i < M_1 \\ \max_{a\in\{0,1\}}[g^1_{n,\beta}(i,j,a)] & : i = M_1,\ j < M_2 \\ V_{n,\beta}(i,j) & : i = M_1,\ j = M_2 \end{cases};$$

$$G^2_{n,\beta}(i,j) = \begin{cases} \max_{a\in\{0,1\}}[g^2_{n,\beta}(i,j,a)] & : \ j < M_2 \\ V_{n,\beta}(i,j) & : \ j = M_2 \end{cases}$$

By the Principle of Optimality, $V_{n,\beta}(i,j+a)$ can be computed recursively as:

$$V_{n+1,\beta}(i,j) = \frac{\beta}{\Lambda}[\underbrace{i\mu_1 V_{n,\beta}(i-1,j) + j\mu_2 V_{n,\beta}(i,j-1)}_{\text{service rate for customer?}} +$$
$$\underbrace{[(M_1-i)\mu_1 + (M_2-j)\mu_2]}_{\text{service rate for free server?}} V_{n,\beta}(i,j) + \underbrace{\lambda_1 G^1_{n,\beta}(i,j) + \lambda_2 G^2_{n,\beta}(i,j)}_{\text{arrival flow?}}]; V_{n,\beta}(i,-1) =$$

$V_{n,\beta}(-1,j) = V_{n,\beta}(i,-1) = V_{0,\beta}(i,j) = 0$.
Since the rewards r_1 and r_2 are positive, $V_{n,\beta}(i,j)$ is monotonically increasing in n and bounded above by $\Lambda^{-1}\sum_{m=0}^{n}\beta^m \max\{r_1,r_2\} \to \frac{\max\{r_1,r_2\}}{\Lambda(1-\beta)}$. Therefore, $V_{n,\beta}(i,j)$'s limit exists, defined by $V_\beta(i,j) := \lim_{n\to\infty} V_{n,\beta}(i,j)$, which can be shown to be the maximum infinite horizon discounted reward.

Aside, the expected average reward starting at (i_0,j_0) is:

$$\phi_{\mathcal{R}(i_0,j_0)} = \sum_{i=1}^{M_1}\sum_{j=1}^{M_2}[\underbrace{\pi_{\mathcal{R}}(i,j)}_{\text{steady-steady distribution of process}} \underbrace{\kappa_{(i,j)}(\mathcal{R})}_{\text{expected reward at }(i,j)}],$$

Some preliminary results shall be listed:

Proposition 3.1: (Derman)

If a policy $\mathcal{R}^*$ is optimal among the class of policies Π for all discounted problems with discount factor β close to 1, then $\mathcal{R}^*$ is also optimal among all policies in the class Π under the average reward criterion.

Definition 3.2

A function $g(i,a)$ is called submodular if $g(i,a') - g(i,a)$ is decreasing in for all $a' >$

a.

Proposition 3.3

If one defines

$$f(i) = \max_a g(i,a) \text{ and } a(i) = \text{argmax}_a g(i,a)$$

Then $a(i)$ decreasing implies that $g(i,a)$ is submodular.

For ease of notation and to follow the paper's notation β will not be written in the subscripts unless it becomes relevant. First, consider the simpler case, where $r_1 < r_2$.

The Optimal Policy when $r_1 < r_2$ When $r_1 < r_2$, the overflow brings an overflow of low paying customers into the secondary queue. Starting with the finite horizon case, let

$$a^k_{n,\beta}(i,j) = \begin{cases} \text{argmax}_a \; g^k_{n,\beta}(i,j,a) & : \; j < M_2 \\ 0 & : \; j = M_2 \end{cases}$$

and then establish some lemmas.

Lemma 3.4

If $V_n(i,j)$ is decreasing concave in j, then so is $G^k_n(i,j)$ for $k = 1, 2$.

Proof Sketch: If $V_n(i,j)$ is decreasing in j, then $G^k_n(i,j)$ is also decreasing in j from 3.1−3.3. For concavity, one uses induction to show that $G^k_n(i,j) - G^k_n(i,j+1)$ is increasing. The argument is simpler when $i < M_1$, when $i = M_1$ the argument is the same for $G^k_n(i,j)$ and $G^k_n(i,j)$, so one can assume $j + 2 < M_2$. Using the submodularity of $g^k_n(i,j,a)$ and decomposing into cases where $a^k_{n,\beta}(i,j+1) = 1$ or 0 proves the result.

Proposition 3.5

For each $i = 0, 1, ..., M$ and $n = 0, 1, ...$, $V_n(i,j)$ is decreasing concave in j; hence, $a^k_n(i,j)$ is decreasing in j for $k = 1, 2$.

Proof Sketch: The proof uses induction and 3.4 to show that $V_n(i,j) - V_n(i,j+1)$ is nonnegative for all i and n, increasing in j. $a^*(i,j)$ is decreasing in j using submodularity. Now, one should capture the intuitive idea the long run benefit of accepting a high paying customer is always higher than not to, whenever doing so is possible.

Proposition 3.6

Let $R = \max(r_1, r_2)$. Then for all $i = 0, 1, ..., M_1$, $j = 0, 1, ..., M_2 - 1$, and $n = 0, 1, ...,$

$$R + V_n(i, j+1) > V_n(i, j).$$

Proof Sketch: Use strong induction over n.
Therefore, the only important decisions that are taken are on whether to accept the low-paying customer (when $k = 1$). It is also clear that whenever such a decision is taken, the primary queue is full $i = M_1$. Furthermore, 3.5 says that $a_n^1(M_1, j)$ is decreasing in j. For finite horizons, this means the optimal action to do is choose

$$c_{n^*} = \min\{j : a_n^1(M_1, j) = 0, \quad j = 0, 1, \ldots, M_2\}$$

to be the maximum number of busy servers to accept a low paying customer.

Proposition 3.7

$V_{n,\beta}(i,j) - V_{n,\beta}(i,j+1)$ is monotone increasing in β and in n, hence $a^*(i,j)$ is monotone decreasing in n and in β for $k = 1, 2$.

Proof Sketch: This uses lots of setup, but it mostly uses induction and monotonocity to decompose $V_{n,\beta}(i,j) - V_{n,\beta}(i,j+1)$ according to the queue and the current state.
The theorem above shows that as the length of the horizon increases, the system becomes more and more selective in accepting lower paying customers, which makes sense since long term considerations may deter the actor from taking the short term gain coming with accepting the lower paying customer. Also, as the discount β increases, the system also becomes more and more selective in accepting lower paying customers, since doing so makes future rewards less attractive. Now, the final result for infinite horizon can be stated:

Proposition 3.8: (Non-discounted case)

Suppose that $r_1 < r_2$. Under the long-run average reward criterion, the optimal policy for accepting or denying service requests is as follows:

1. Type 2 customers are accepted whenever there is an available server.
2. There exists a critical number c^* such that a type 1 customer is accepted if and only if fewer than c^* servers in the secondary queue are busy.

Proposition 3.8: (Discounted case)

Suppose that $r_1 < r_2$. Under the long-run average reward (maximum infinite horizon discounted reward with discount factor β) criterion, the optimal policy for accepting or denying service requests is as follows:

1. Type 2 customers are accepted whenever there is an available server.
2. There exists a critical number c^*_β such that a type 1 customer is accepted if and only if fewer than c^*_β servers in the secondary queue are busy. Furthermore, for all $\beta > 0$, $c^*_\beta \geq c^*$.

Proof Sketch: This mostly comes from the fact that the sequences at play being monotone and bounded preserves properties under limits. Now, it can be said that $a^1(M_1, j) := \lim_{n\to\infty} a^1_n(M_1, j)$ is decreasing in j. For finite horizons, this means the optimal action to do is choose

$$c^*_\beta = \min\{j : a^1_\beta(M_1, j) = 0, \quad j = 0, 1, \ldots, M_2\}$$

to be the maximum number of busy servers to accept a low paying customer. When there is no discount, it is instead:

$$c^* = \min\{j : a^1(M_1, j) := \lim_{\beta\to 1} a^1_\beta(M_1, j) = 0, \quad j = 0, 1, \ldots, M_2\}$$

This kind of policy is called trunk reservation policy.

The Optimal Policy when $r_1 > r_2$

When $r_1 > r_2$, it is useful to generalize the threshold function c^* to

$$c^*_i = \min\{j : a^1(i, j) = 0, \quad j = 0, 1, \ldots, M_2\},$$

whereas the only value considered used to be $c^* = c^*_{M_1}$. But now, since the high paying customers are in the overflow from the first queue, our policy will depend on the state i of that queue.

Lemma 3.9

For each n, $V_{n,\beta}(i+1, j) - V_{n,\beta}(i, j) \geq 0$ and $V_{n,\beta}(i+1, j) - V_{n,\beta}(i, j) \geq V_{n,\beta}(i+1, j+1) - V_{n,\beta}(i, j+1)$ imply:

1. $G^k_n(i, j)$ is increasing in i.
2. $(V_{n,\beta}(i+1, j+1) - V_{n,\beta}(i, j+1)) - (V_{n,\beta}(i+1, j) - V_{n,\beta}(i, j))$ is decreasing in j.

Proof Sketch: Part 1 follows by definition, whereas Part 2's proof splits again into cases like in the proof for 3.4
The main result in finite horizon for $r_1 > r_2$ is then as follows:

Proposition 3.10

For $k = 1, 2$, $G_n(i, j)$ is nonnegative and decreasing in j, thus $V_n(i, j)$ is decreasing in i for each j.

Proof Sketch: Use induction, Lemma 3.9, and submodularity.

Proposition 3.11: (Non-discounted case)

When $r_1 > r_2$, the optimal policy for the average reward criterion is as follows:

1. Type 1 customers are accepted whenever there is an available server.
2. For each state i in the primary queue, there exists a threshold value c_i^* such that when there are i busy servers in the primary queue, a type 2 customer is accepted if and only if fewer than c_i^* servers are busy in the secondary queue.
3. Furthermore, the threshold value c_i^* decreases in i; in other words, the number of trunks reserved increases as the primary queue becomes more congested.

Proposition 3.11: (Discounted case)

When $r_1 > r_2$, the optimal policy for the long-run average reward (maximum infinite horizon discounted reward with discount factor β) criterion is as follows:

1. Type 1 customers are accepted whenever there is an available server.
2. For each state i in the primary queue, there exists a threshold value $c_{i,\beta}^*$ such that when there are i busy servers in the primary queue, a type 2 customer is accepted if and only if fewer than $c_{i,\beta}^*$ servers are busy in the secondary queue.
3. Furthermore, the threshold value $^*_{i,\beta}$ decreases in i; in other words, the number of trunks reserved increases as the primary queue becomes more congested.

Proof Sketch: From Theorem 3.10, Theorem 3.11 is true for every finite horizon problem. Using the same technique as in the proof to Theorem 3.8, the result for this problem easily follows.

Section 4: Computing the Optimal Policy

Now that the optimal policy has been found, the paper investigate different methods to actually compute it. In particular, Policy Iteration and Successive Approximation are used to Poisson and Overflow Models.

Policy Iteration

The paper introduces the policy iteration algorithm, which iteratively refines an initial policy to converge towards the optimal policy. Given an MDP, let g be its average expected reward. For large horizon, the maximum expected reward at time n is about $V_n(i) \approx ng + v_i$, where v_i is an initial condition term. In that case, Bellman's equations becomes:

$$(n+1)g \approx V_{n+1}(j) = \max_a [\kappa_i(a) + \sum_{i,j} p_{ij}(a) V_n(j)]$$

$$\Rightarrow (n+1)g \approx ng + \max_a [\kappa_i(a) + \sum_{i,j} p_{ij}(a) v_j],$$

where $\kappa_i(a)$ is the expected one-period reward in state i with action a. Now, the algorithm below of policy iteration is used to solve for g, the MDP's average expected reward:

Remark 4.1: Policy Iteration Algorithm

1. Select any policy $\mathcal{R} = [a_0, a_1, \ldots, a_N]$.
2. Solve for v_i and g,
$$g + v_i = \kappa_i(a) + \sum_j p_{ij}(a) v_j, \quad v_0 = 0.$$
3. For each i, find a_i' maximizing
$$\kappa_i(a) + \sum_j p_{ij}(a) v_j.$$
4. If $a_i = a_i'$ for all i, then $\mathcal{R}^* = [a_0, a_1, \ldots, a_N]$ is the optimal policy yielding the maximum average return g. Otherwise, replace R by R' and return to step 1.

- Advantage of policy iteration: the number of iterations needed for convergence is relatively small and independent of the state space size;
- Inconvenient of policy iteration: Each iteration requires solving a system of linear equations with dimension proportional to the state space size, which can be computationally problematic when the state space is big.

Successive Approximation

Another method the paper mentions is successive approximation, which instead produces a sequence of approximate policy that convergence to the optimal policy, formulated as such:

Proposition 4.1: White

Let $v_0(i) = 0$ for all i. For $n = 0, 1, 2, \ldots$, define

$$g(n) = \max_a[\kappa_0(a) + \sum_j p_{0j}(a)v_n(j)], \quad v_{n+1}(j) = \max_a\{\kappa_j(a) - g(n) + \sum_j p_{ij}(a)v_n(j)\},$$

$$v_n(0) = 0.$$

Then, $g(n) \to g^*$ where g^* is the maximum average reward.

Proposition 4.2: Odoni

Let

$$g_n^+ = \max\{g(n), \max_i[g(n) + v_{n+1}(i) - v_n(i))]\},$$

$$g_n^- = \min\{g(n), \min_i[g(n) + v_{n+1}(i) - v_n(i))]\},$$

where $g(n)$ and $v_n(i)$ are as defined in Proposition 4.1. For each n, $g_n^- \geq g^* \geq g_n^+$. Furthermore, g_n^+ is monotone increasing, g_n^- is monotone decreasing, and both converge to g^*.

Each step n produces an approximate policy $\mathcal{R}_n$ that induces some $g(n)$, which can be terminated when the error is within some relative tolerance ε. In other words,

$$\text{Fix } \varepsilon > 0. \text{ Then, for any n, terminate when } \frac{g_n^+ - g_n^-}{g_n^+} < \varepsilon.$$

- Advantage of successive approximation: skips the need of solving computationally expensive linear equations;
- Inconvenient of successive approximation: the number of iterations can be large, it's been shown that it increases with the size of the state space.

Solving the Poisson Model

To solve the Poisson model, the paper provides a modified version of the policy iteration model.Define $g(\mathcal{R})$ to be the maximum average reward under policy $\mathcal{R}$. For each policy $\mathcal{R}$, let $\kappa_i(\mathcal{R})$ be the corresponding expected reward in state i and $\pi(\mathcal{R}) = \{\pi_i(\mathcal{R})\}_{i=0,\ldots,M}$ be the steady-state distribution. The average reward, computed in step 2 of the policy iteration algorithm, becomes:

$$g(\mathcal{R}) = \sum_{i=0}^{M} \pi_i(\mathcal{R})K_i(\mathcal{R}).$$

For two-dimensional birth-death processes that the paper has modelled, the other terms are:

$$\begin{aligned} v_0 &= 0, \\ v_1 &= (p_{12}(\mathcal{R}))^{-1} \left[g(\mathcal{R}) - \kappa_0(\mathcal{R}) \right], \\ v_{i+1} &= p_{i,i+1}(\mathcal{R})^{-1} \left[(1 - p_{ii}(\mathcal{R}))\, v_i - p_{i,i-1}(\mathcal{R}) v_{i-1} - 1 - \kappa_i(\mathcal{R}) + g(\mathcal{R}) \right], \end{aligned}$$

where

$$p_i(\mathcal{R}) = \frac{p_i(\mathcal{R})}{\sum_{j=0}^{M} p_j(\mathcal{R})},$$

and

$$p_i(\mathcal{R}) = \frac{\lambda_0(\mathcal{R})\lambda_1(\mathcal{R}) \dots \lambda_{i-1}(\mathcal{R})}{\mu_1(\mathcal{R})\mu_2(\mathcal{R}) \dots \mu_i(\mathcal{R})}.$$

Applying the algorithm with these terms, one ends up with a computation effort complexity proportional to the state space size n. Quick convergence of the method should also be expected, since it is a property of policy iteration. Now, in the case of trunk reservation policies, which was shown to be optimal for admission control, another result makes the process of computing the optimal policy with policy iteration even more efficient:

Proposition 4.3: Theorem 4.3

In solving the Poisson model, if the initial policy is trunk reservation, then one policy iteration step yields another trunk reservation policy.
Remark: Since policy improvement is guaranteed to yield the optimal policy, the foregoing result provides another proof that the optimal policy is indeed trunk reservation.

Solving the Overflow Model

The paper concludes with analyzing overflow models where the overflow is made of low paying customers ($r_1 < r_2$), which one recalls to be a 2D birth-death process. In this case, the last equation for $p_i(\mathcal{R})$ in the previous section cannot be computed since the stead-state distribution of a 2D birth-death process is not known. The state space is also effectively squared (since we have two of them to keep track of) making any possible algorithm solving the problem inefficient. Therefore, the author investigates approximate solutions.

- The first one uses approximations based on interrupted Poisson processes (IPPs), which is a Poisson process that is turned on for an exponentially distributed time, and then turned off for an independent exponentially distributed time.
- The parameters for IPP are chosen to match the three first factorials moments of the two processes.

- The result is an approximate 2D birth-death process with size of $2M_1$ or $2M_2$ instead of M_1M_2, which is good if one of the queue sizes is really big.

The other approach is to just approximate the overflow process by a Poisson process with a parameter λ' corresponding to the average overflow rate, where

$$\lambda' = \lambda_1 \times \text{Blocking probability at the primary queue} = \lambda_1 \frac{\lambda_1^{M_1}/M_1!}{\sum_{k=0}^{M_1} \lambda_1^k/k!}$$

Even though this is a very simple approximation for the overflow, the paper uses numerical evidence to show that the approximation is comparable to the first one in performance and that those two approaches approximate the original model pretty well. In the numerical example, the Poisson approximation is solved with policy iteration. The data shows that the approximations significantly reduce computing time while still yielding pretty much the same optimal policy (modulo a few small mistakes by one trunk to be reserved).

Bibliography

[1] V. Nguyen, "On the Optimality of Trunk Reservation in Overflow Processes," *Probability in the Engineering and Informational Sciences*, vol. 5, no. 3, pp. 369–390, 1991. doi:10.1017/S0269964800002163.

[2] B. L. Miller, "A Queueing Reward System with Several Customer Classes," *Management Science*, vol. 16, no. 3, pp. 234–245, 1969.