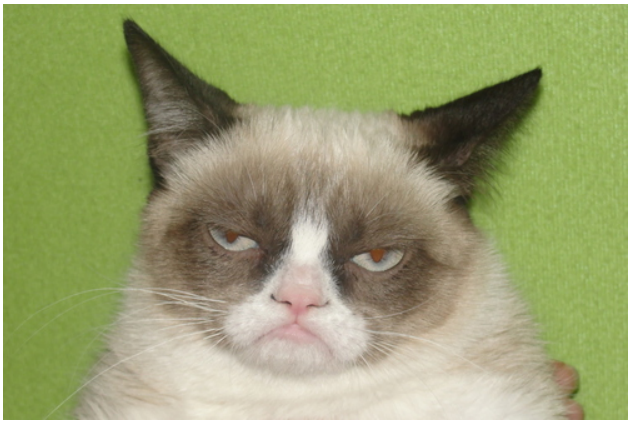
When the cat goes: the paper is ready.



Understanding the effect of selfish behaviour in a series of two queues.

Jason Young, Vincent Knight

January 2, 2014

Abstract

Hierarchical queues in HC; PoA; Simulation model; Heuristics developed to obtain optimal policies;

1 Introduction

Queues arise in a variety of settings: road traffic, data transfer and healthcare are just a few examples. A large quantity of literature has investigated strategic decision making (for example: whether or not to join a queue) with regards to queues stemming from []. When leaving one queue; customers (which we will refer to as players) often join a second queue and this is what is referred to as a hierarchical queueing system (as shown in Figure 1).

There is a wide range of literature evidencing the negative effect of selfish actions in queueing systems []. The following is a general conclusion of a majority of the literature:

> Selfish users make busier systems.

- Naor;
- VK+PH;
- RS;
- Bell Stidham;
- Hassani book;
- Rouhgharden;

The effect of selfish behaviour when compared to optimal behaviour can be quantified as the Price of Anarchy (PoA). This is defined as the ratio of the selfish ($\tilde{C}$) and optimal (C^*) costs:

$$\text{PoA} = \frac{\tilde{C}}{C^*}$$

The contribution of this paper is to consider two stations (to avoid confusion we will refer to a queue and service station as a station) in series allowing for two consecutive decisions: whether or not to join the station. Players who join the first station upon completion of service will potentially drop out from the entire system. A potential application of this is a healthcare system where patients must choose to see their general practitioner before obtaining care from a specialist/emergency centre.

In Section 2, the model will be described. A novel aspect of this work is that the cost of a given policy is evaluated through the use of purpose built Agent Based Simulation (ABS). In Section 3 various heuristic approaches to obtaining optimal policies are considered. These range from random search algorithms to a more sophisticated approach that depends on the analytical formula for an $M/M/c$ queue [**?**]. Before concluding in Section 5 a variety of numerical experiments will be shown in Section 4 demonstrating the effect of selfish decision making in queues in series.

2 Model

Let $k\{1, 2\}$ be the index for each station in the system. The players arrive into the system at a rate Λ. The players know the rate of service μ_k at each station, the number of servers at each station c_k and also the number of players currently in each station. Upon observing a station, players must select one of two potential strategies: 'join' or 'skip'. The strategy picked at station k is denoted by $d_k \in \{0, 1\}$:

$$d_k = \begin{cases} 1, & \text{if player joins queue } k \\ 0, & \text{if player skips queue } k \end{cases} \tag{1}$$

The cost associated with joining system k is denoted by J_k and corresponds to the actual time spent in system k. The cost associated with skipping station k is denoted by β_k. The cumulative cost of going through the entire system is given by $C = C(d) = C_1(d) + C_2(d)$ where:

$$C_k = \begin{cases} \beta_k & \text{if } d_k = 0 \\ J_k & \text{if } d_k = 1 \end{cases} \tag{2}$$

The expected value of J_k when there are m players present upon arrival in system k is given by:

$$E[J_k] = \frac{m+1}{\mu_k c_k} \tag{3}$$

A final parameter of our model is the dropout probability p. This is the probability with which a player who joins the first station will exit the entire system (implying $C_2 = 0$).

All of this is summarised in Figure 1.

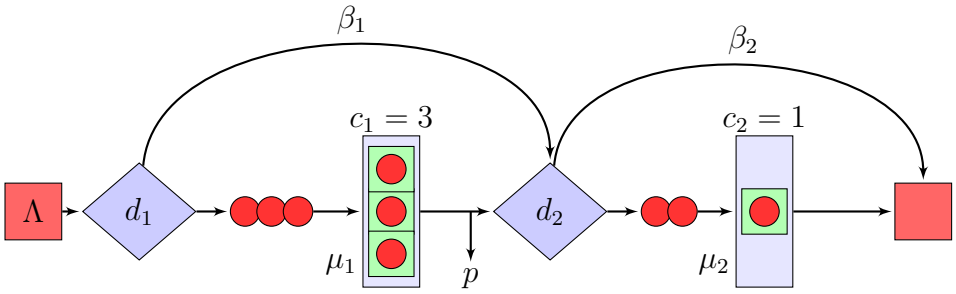


Figure 1: Diagrammatic representation of the model.

If a general population of players is considered a state dependent policy $\tau = (n_1, n_2)$ can be defined that all players follow. In general this policy will be of the form:

$$d_k = \begin{cases} 1, & \text{if } m_k \leq n_k \\ 0, & \text{otherwise} \end{cases}$$

Where m_k is the number of players present at station k upon having to make a decision d_k. In other words a policy τ is a set of two threshold policies, one for each station, simply denoting a threshold at which players skip. As $d = d(\tau)$ we have $C = C(\tau)$.

DISCUSS SOLUTION SPACE FOR τ: Ω.

The optimal cost is thus defined as:

$$C^* = \min_{\tau \in \Omega} E[C(\tau)] \tag{4}$$

Note that the restriction of the optimisation problem to only consider stationary policies is potentially an incorrect one. Indeed, a non homogeneous policy could in fact be optimal however **FIND REFERENCE AND PAPER THAT SAYS THAT AN OPTIMAL POLICY WOULD BE HOMOGENOUS**.

The particular policy that gives C^* will be denoted as τ^*. Another policy that will be considered in this paper is the selfish policy $\tilde{\tau}$. The selfish policy is the policy under which individual players reduce their expected cost.

To obtain the selfish policy a further assumption is made: players are pessimistic. It is assumed that players ignore the dropout probability p at decision d_1 so that we have (recall equation (1)):

$$\tilde{d}_k = \begin{cases} 1, & \text{if } E[J_k] \leq \beta_k \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

This in turn gives (recalling (3)):

$$\tilde{\tau} = (\lfloor \beta_1 \mu_1 c_1 - 1 \rfloor, \lfloor \beta_2 \mu_2 c_2 - 1 \rfloor) \tag{6}$$

To evaluate the $C(\tau)$ for given τ a bespoke simulation model has been developed (written in Python [**?**]). The model itself is a combination of Discrete Event [**?**] and Agent Based [**?**] Simulation (taking advantage of the Object Orientated framework available in Python). The flow of the simulation model is shown in Figure 2.

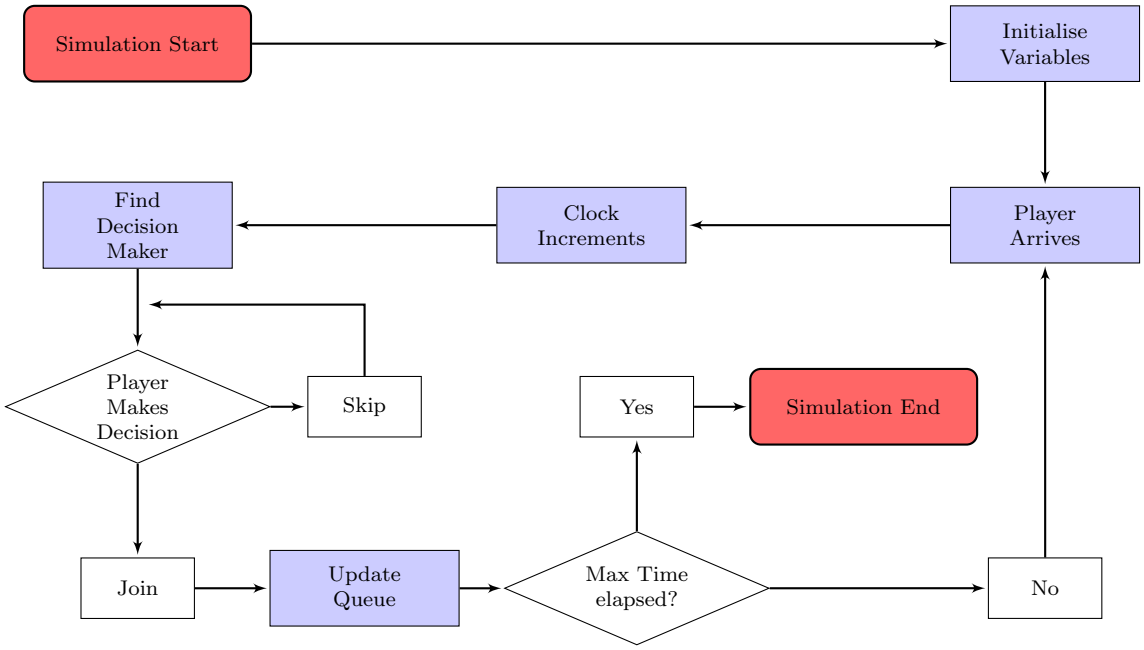
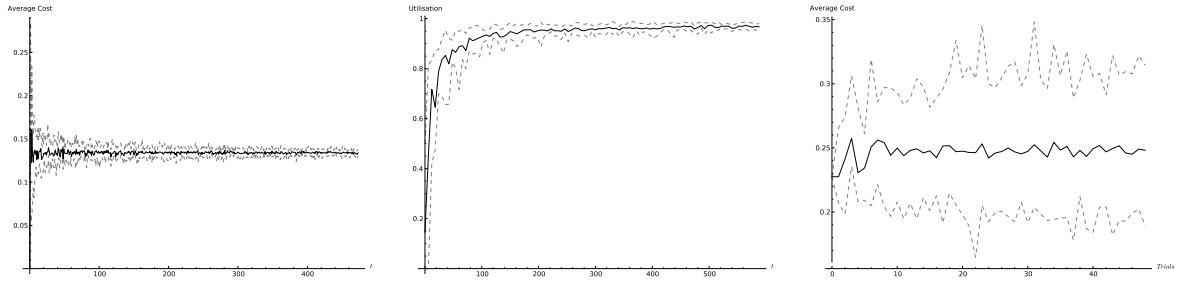


Figure 2: Flow chart describing the simulation model

The input to the model (apart from the system parameters described in Section 2) is a policy τ and the required output is $C(\tau)$. An investigation of run time, warm up time, and number of trials was carried out to ensure a reliable value of $C(\tau)$ is output [**?**]. Graphical methods were used for system parameters discussed in Section 4 and are shown in Figure 3.

A run time of 250 time units was shown to reduce the variation in $C(\tau)$ (Figure 3a) whilst a warm up period of 50 time units reduced gave a stable utilisation of servers (Figure 3b).



a: Run Time b: Warm up time c: Number of trials

Figure 3: Investigating run time, warm up time and number of trials required to obtain reliable values of $C(\tau)$

Finally experiments of 16 trials each were chosen despite the lack of variation shown in Figure 3c. This relatively large number of trials did not prove to be computationally expensive as they were all run in parallel using **??**.

As discussed, the simulation model is used as an objective function for a given τ. In the next section, various heuristic approaches to obtaining τ^* (recall (4): this corresponds to the optimal policy) will be discussed.

3 Heuristic Optimal Policies

Using the simulation model as an objective function a variety of search based heuristics have been tested []. All of these use $\tilde{\tau}$ as an initial solution and make minor perturbations accepting improvements within a specific region:

- Fixed search range: randomly evaluate a fixed range around the initial policy;
- Variable search range: increase the range of the search range;
- Random descent: randomly search the entire solution space;
- Iterative random descent: JASON CHECK THESE AND ALSO INCLUDE A SENTENCE FOR THIS ONE. I DONT LIKE THE NAMES. ARE THEY THE CORRECT NAMES?

3.1 Heuristic 1

A comparison of the performance of the various heuristics above is shown in Figure 4.

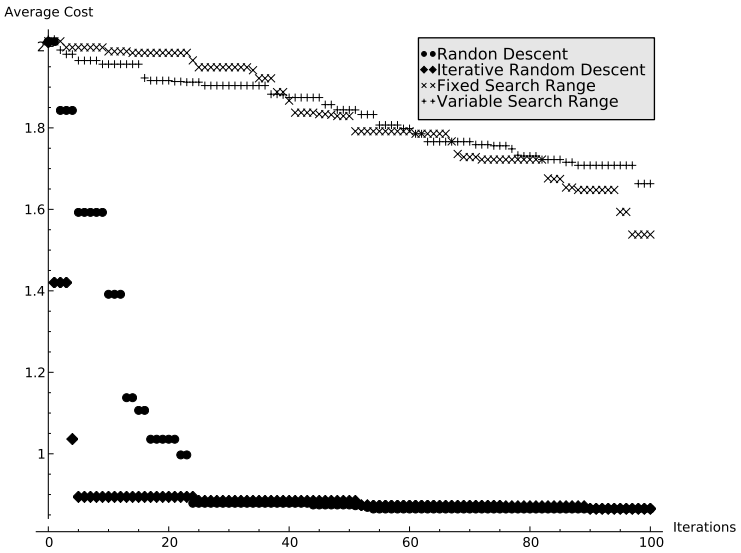


Figure 4: Comparing local search heuristics

As shown, the most efficient of the chosen heuristics is the 'Random Descent'. This algorithm is nonetheless computationally expensive. Some scenarios of Section 4 taking various hours to run (despite the use of [**?**]). The aim of the work presented here is to gain a global understanding of PoA behaviour and as such a large number of optimisations are needed. The next sections will demonstrate further heuristics.

3.2 Heuristic 2

This heuristic is based on the assumption that the arrival rate λ_2 at the second station is in fact Markovian. For $p = 0$ this is in fact not an assumption **??** however for $p > 0$ the arrival rates at the second station no longer exhibit Markovian behaviour. Nevertheless as shown in Figure 5 this heuristic performs very favourably and in fact the non Markovian behaviour seems to not affect the performance of our heuristic.

The arrival rate at the second station is given by:

$$\lambda_2 = \Lambda \times (1 - p \times (1 - \pi_{n_1}^{(1)})) \tag{7}$$

Where $\tau = (n_1, n_2)$ and $\pi_i^{(1)}$ is the probability that the first station is in state $0 \leq i \leq n_1$. Note that the first station corresponds to an $M/M/c/n_1$ queue and so we have:

$$\pi_i^{(1)} = \frac{1}{i!} \left(\frac{\Lambda}{\mu_1} \right)^i \pi_0 \qquad 0 \leq i \leq c_i \tag{8}$$

$$\pi_i^{(1)} = \frac{1}{c_i^{i-c_i} c_i!} \left(\frac{\Lambda}{\mu_1}\right)^i \pi_0 \qquad c_i < i \leq K \tag{9}$$

where

$$\pi_0^{(1)} = \left[\sum_{n=0}^{c_i-1} \frac{1}{n!} \left(\frac{\Lambda}{\mu_1}\right)^n + \sum_{n=c_i}^{K} \frac{1}{c_i^{n-c_i} c_i!} \left(\frac{\Lambda}{\mu_1}\right)^n\right]^{-1} \tag{10}$$

Using the Markovian assumption as to the distribution of λ_2 a similar formula for $\pi_i^{(2)}$ can be obtained, replacing the relevant parameters.

Using this and recalling (**??**):

$$E[C_i] \approx \pi_{n_i}^{(i)} \beta_i + \sum_{j=0}^{c_i} \frac{\pi_j^{(i)}}{\mu_i} + \sum_{j=c_i}^{n_i-1} \frac{\pi_j^{(i)}(j+1)}{\mu_i c_i} \tag{11}$$

and so:

$$E[C] \approx E[C_1] + \frac{\lambda_2 E[C_2]}{\Lambda} \tag{12}$$

Note that the approximation of (11) is in fact an equality for $i = 1$. Using (12) the cost of any policy τ can be approximated immediately and so τ^* can be heuristically obtained by an exhaustive search of the solution space. To obtain an actual mean cost of τ^* the simulation model needs to only be run once. A comparison of heuristics 1 and 2 is shown in Figure 5, it is evident that the assumption made with regards to the Markovian nature of λ_2 do not seem to have a large effect. The fact that results for $\Lambda > 130$ are not shown for heuristic 1 are due to the fact that they were not obtained as they required more than 80 hours of computations on [**?**].

Before considering various results in Section 4, one final heuristic is proposed that only applies if $c_1 = c_2 = 1$.

3.3 Heuristic 3

In [**?**] thresholds n^* are obtained for single server queues that optimise the expected cost. n^* is shown to be the solution to the following inequality:

$$\leq n^* \leq \tag{13}$$

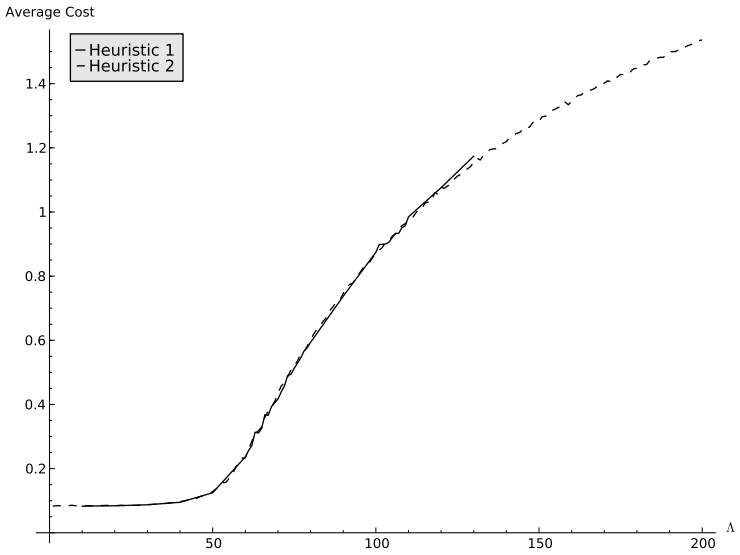


Figure 5: Comparing Heuristic 1 and 2

Using (13) another potential policy is $\tau = (n_1^*, n_2^*)$. Figure 6 shows a comparison of these three heuristic approaches for a series of single server queues.

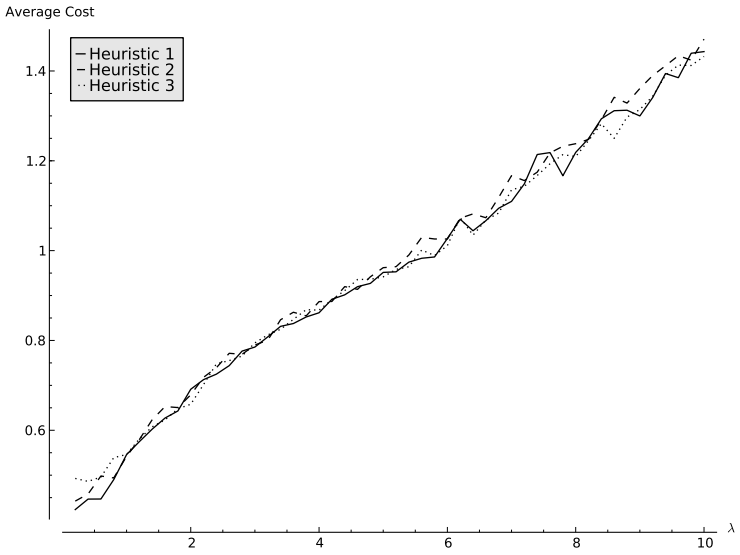


Figure 6: Comparing all heuristics

All the heuristics seem to behave adequately (this has been confirmed through a large quantity of experimentation). For the purposes of this paper heuristic 2 will be used from now on given the very low computational cost.

4 Results

- Scenarios...

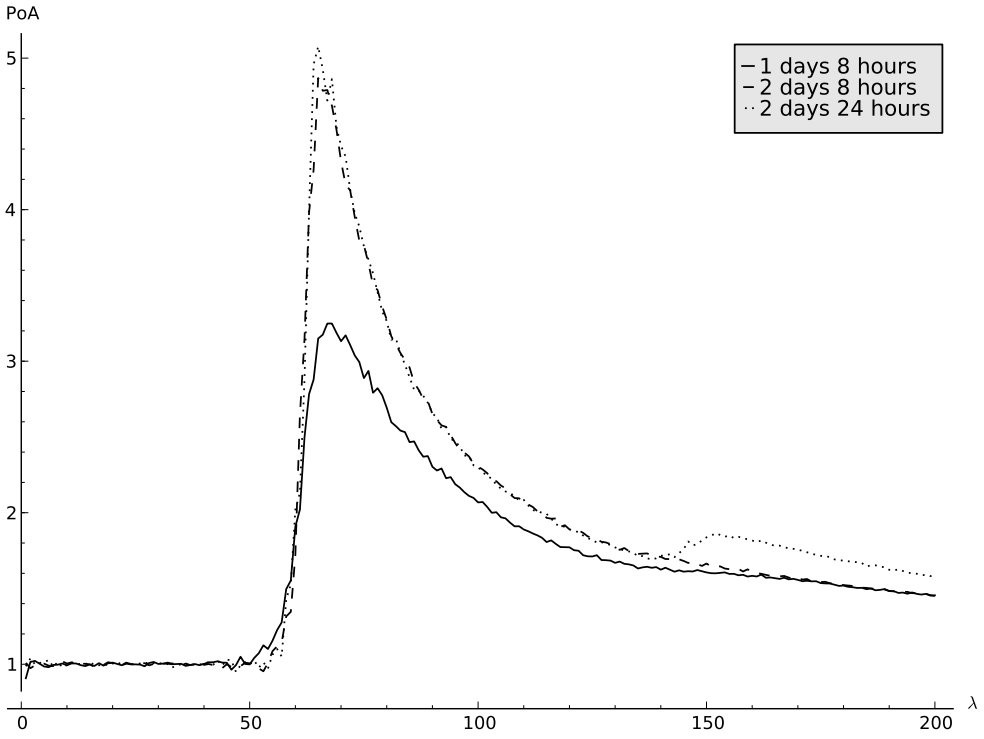


Figure 7: PoA for varying lambda

5 Conclusions and Further work

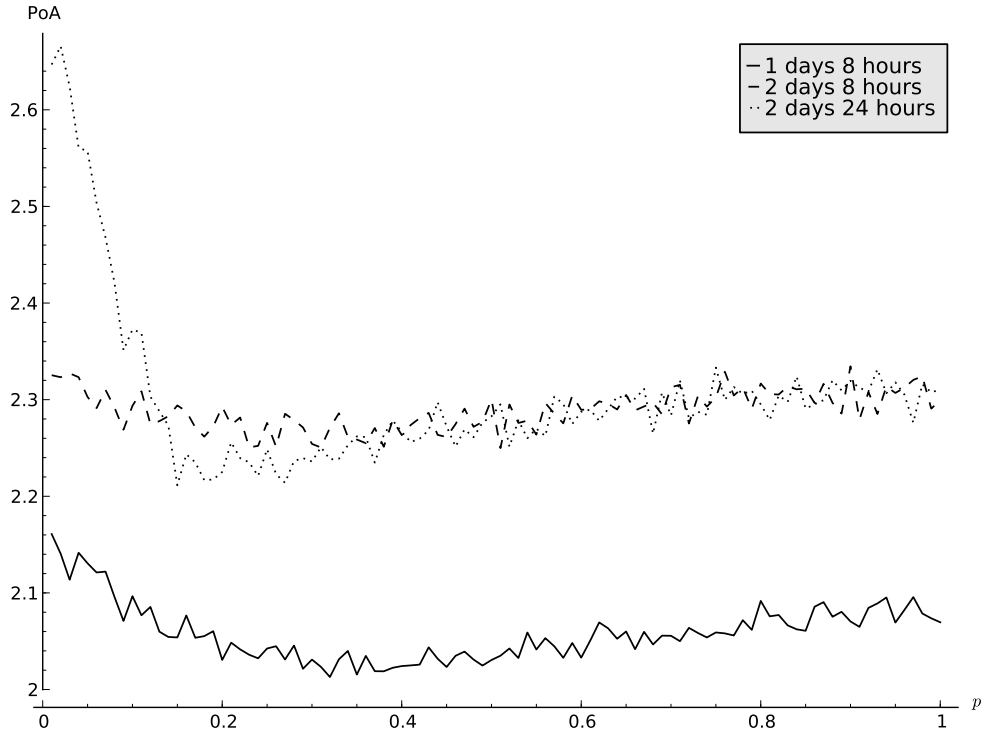


Figure 8: PoA for varying exit prob

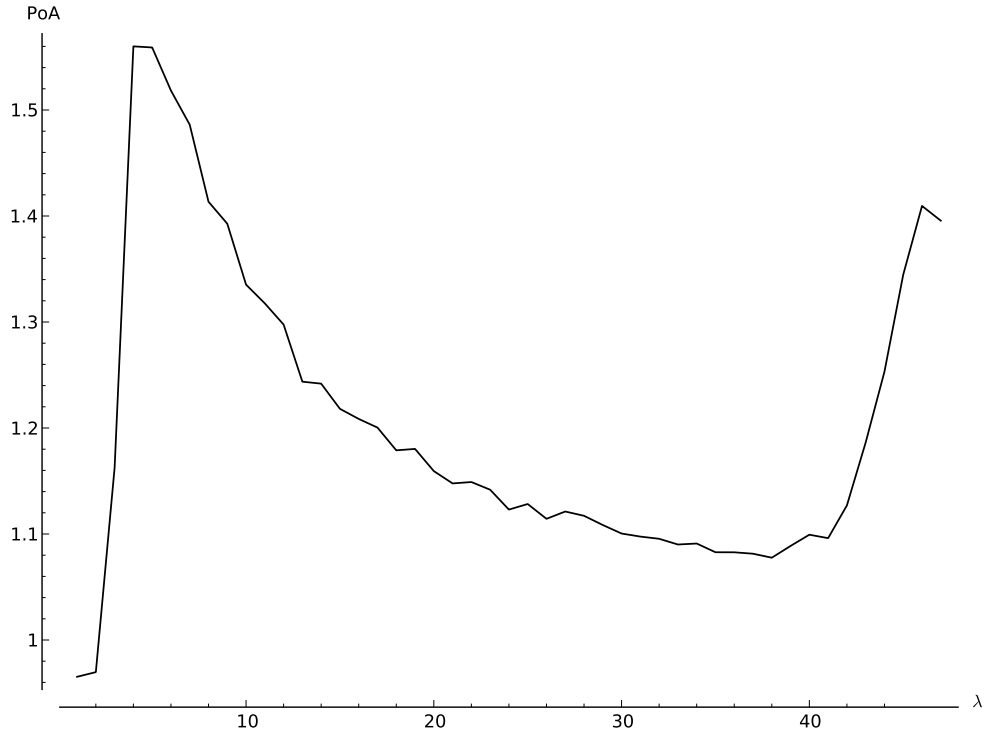


Figure 9: Another set of scenarios