We simulate the performance comparison between RL and SC algorithm when the inter-arrival rates of different requests follow different distribution at high, medium or low arrival rates.

This way, we are trying to see if there is a situation in which RL performs better than SC.

1st Simulation:

Exponential Distribution at High Arrival Rate, Erlang Distribution at Medium and Normal Distribution at Low Inter-arrival Rates

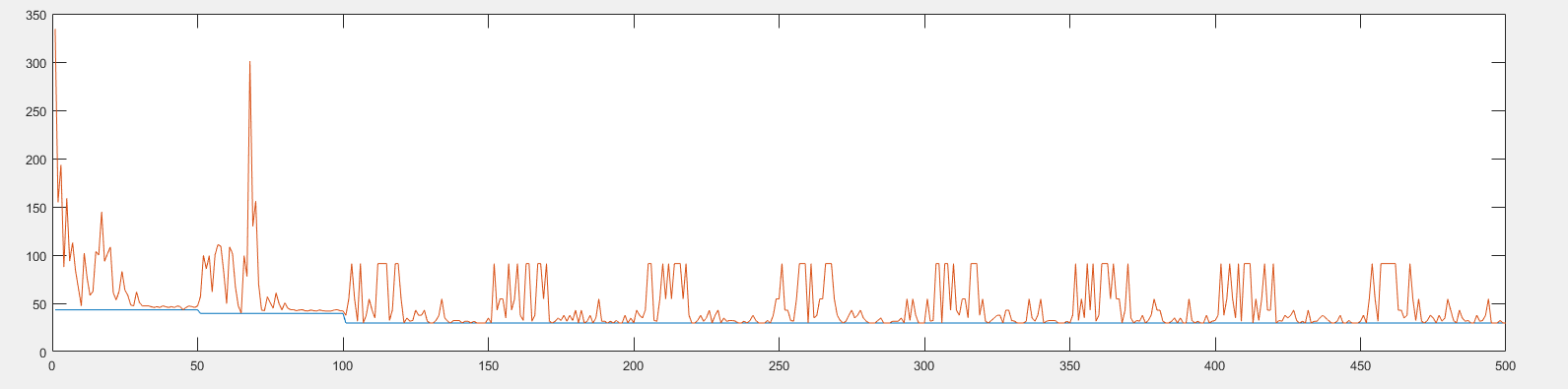


Fig 1.a Average Latencies Comparision (RL-red, SC-Blue)

Setting for Fig 1. Each arrival rate is repeated 10 times, each time frame has 10 requests, Beta=0.6, inter-arrival rates (50,60,70,80,100,150,200, 400,1000,2000). Exponential distribution for 50,60, Erlang distribution from 80 to 200, and Normal distribution above 200.

Ten time frames of each arrival rate is placed adjacent to each other, therefore there is high fluctuation at the beginning and it settles down, and again there is high fluctuation and it settles down again.

We can see that RL behaves less efficiently than RL in all cases.

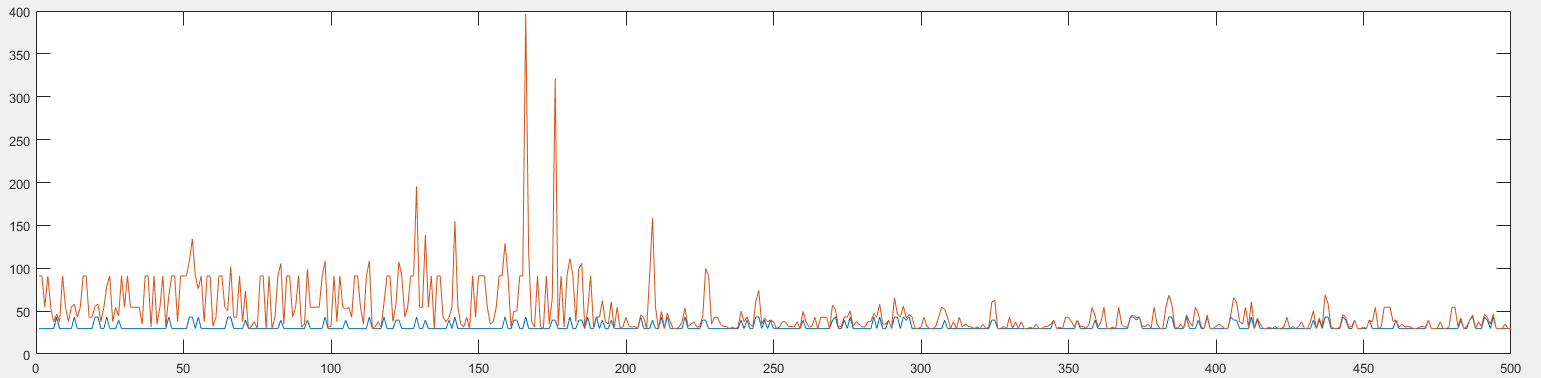
Time Frames Placed Randomly for the above case: 

Fig 1.b Average Latencies for Random Inter-arrival Rates

We can see that RL(red) is highly turbulent in the learning phase, in the beginning, as it learns the best configuration for all inter-arrival rates and then it starts to closely follow the performance of SC around the latter parts.

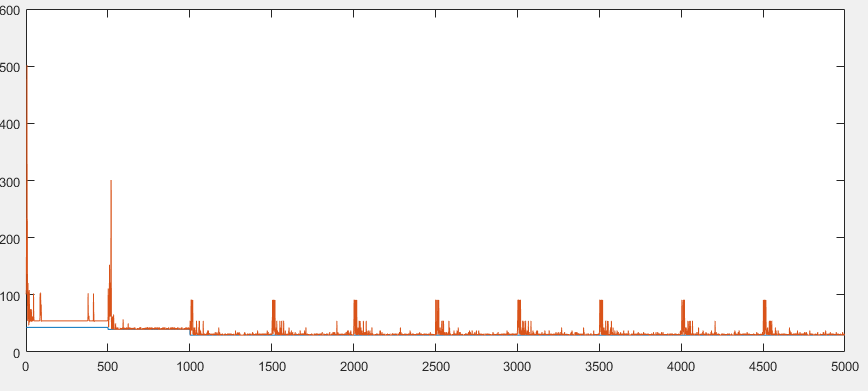
The same setting is repeated for long time. Ie each arrival rate is repeated for 500 times and they are not placed randomly. 

Fig 2.a Average Latencies for Non-random Inter-arrival Rates

The figure is intuitive and resembles with figure 1.a. We can see that in the first 500 time frames, it seems RL converges exactly to one of the sub-optimal configurations, while on the other arrival rates, it is constantly changing between optimal and sub-optimal cases.

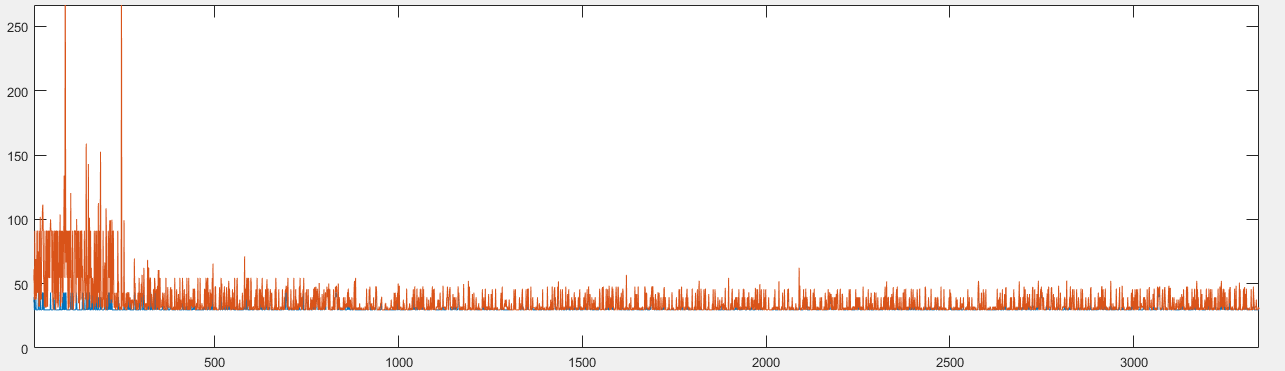
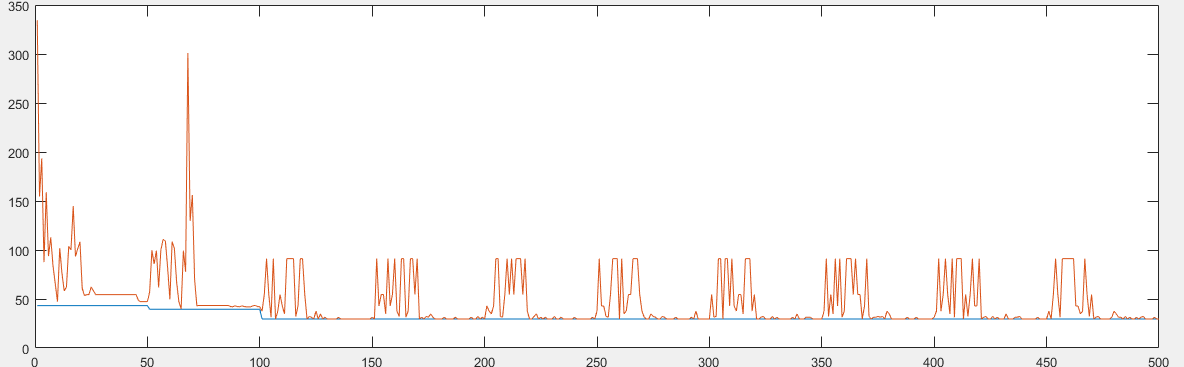
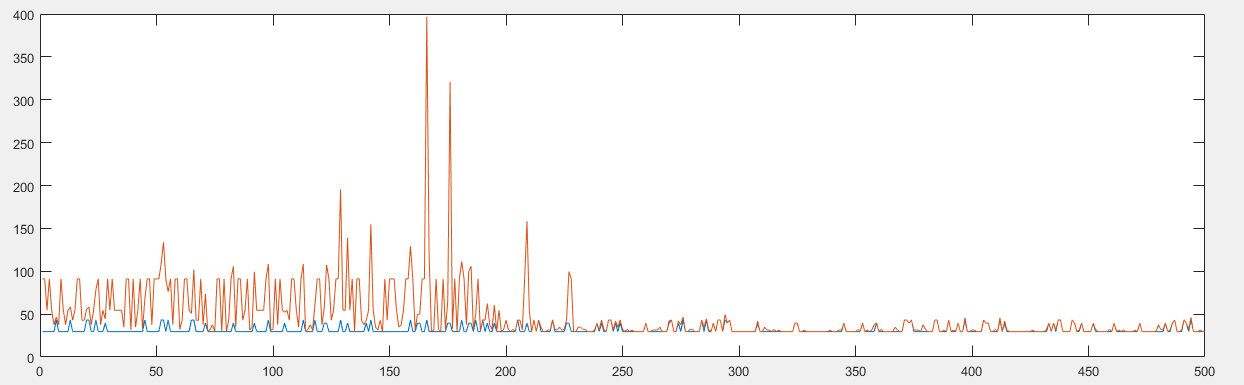


Fig 2.b Average Latencies for Random Inter-arrival Rates

Now we perform the same experiments for learning rate, beta=0.9



2.a Average Latencies for Non\_random inter-arrival rates(beta=0.9)



2.b Average latencies for Random inter-arrival rates (beta=0.9)

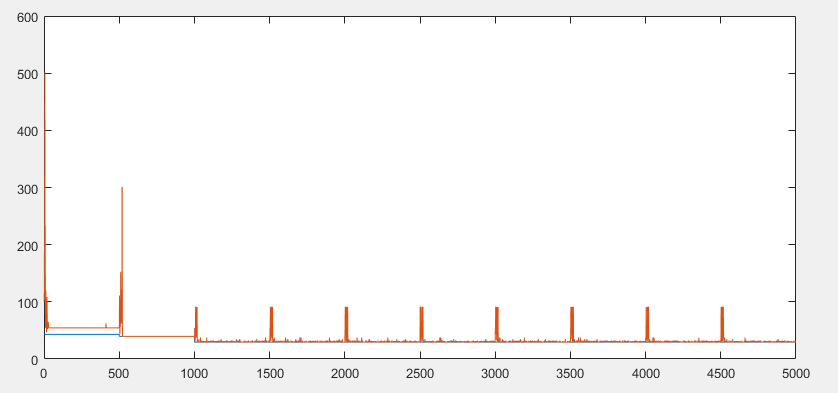


Fig 2.c Average Latencies for Non\_random Arrival rate: beta=0.9

We can see that for the first two arrival rates, the convergence is very good. But, the first arrival rate converges to the sub-optimal configuration, but the second one converges to the optimal configuration.

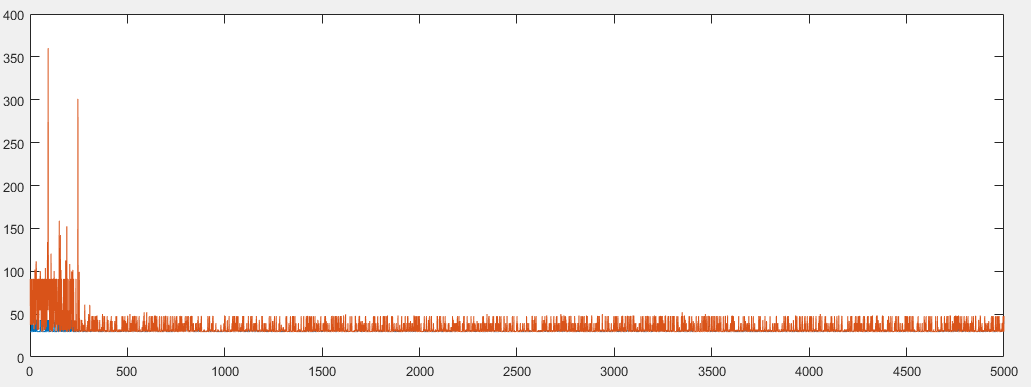


Fig 2.d Average Latencies for Random Arrival Rate, beta=0.9

For beta=0.4

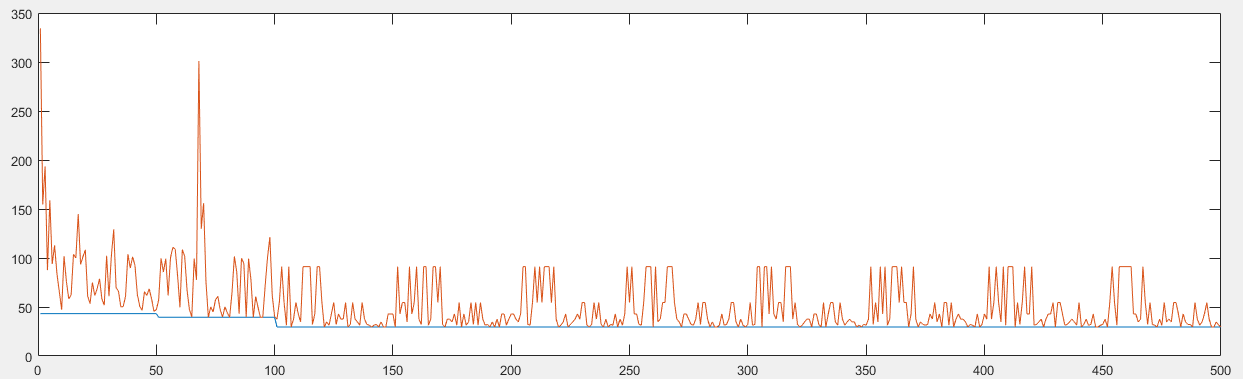


Fig 3.a Average latencies for non-random Arrival rate, beta=0.4

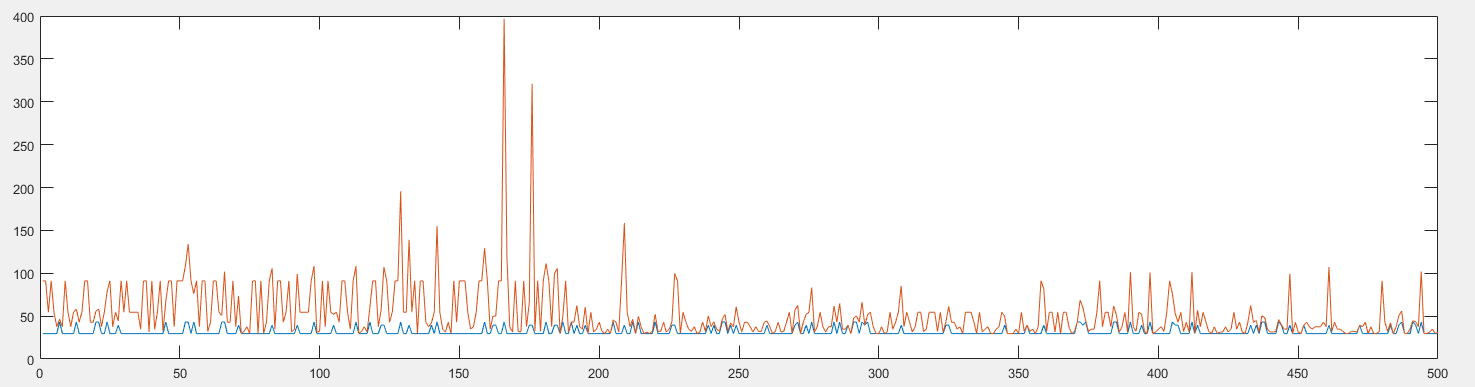
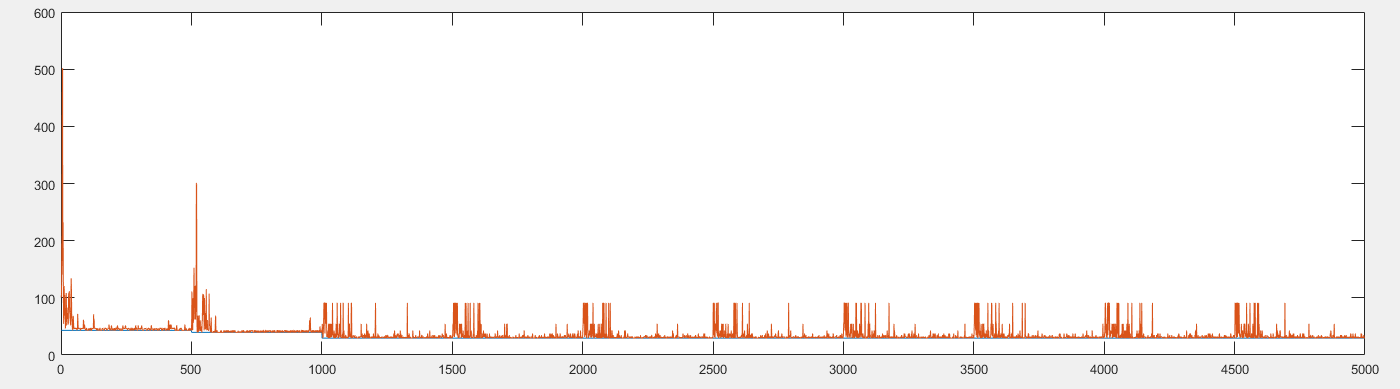


Fig 3.b Average Latencies for random arrival rate, beta=0.4



For 3.c Average Latencies for non-random arrival rate, beta=0.4, each arrival rate repeated 500 times

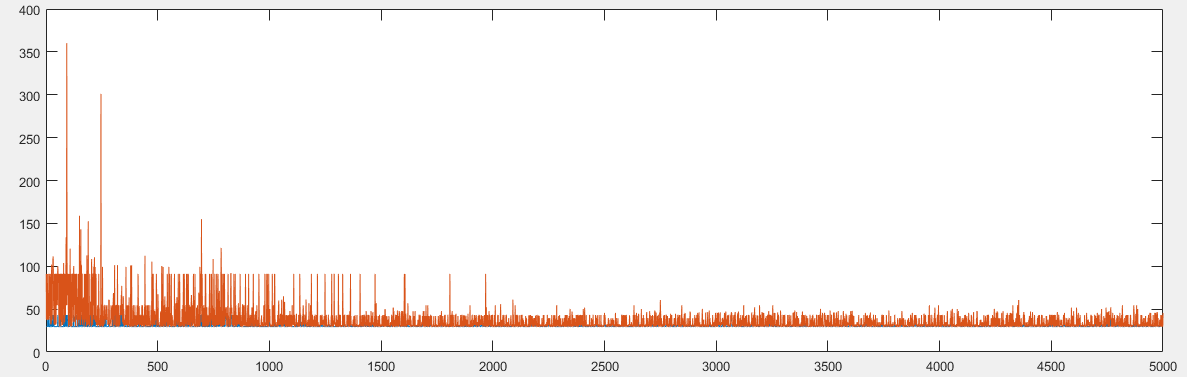


Fig 3.d Average Latencies for random-arrival rates, beta=0.4, each arrival rate repeated 500 times

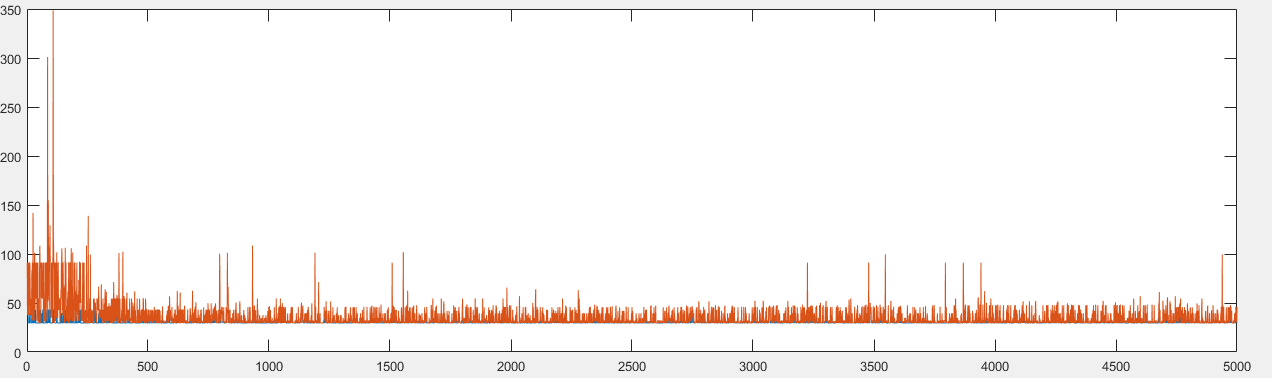


Fig 3.e Average Latencies for random-arrival rates, beta=0.5, each arrival rate repeated 500 times

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Now for Exponential for high arrival rate, ChiSquare for Medium and Erlang for low arrival rate.

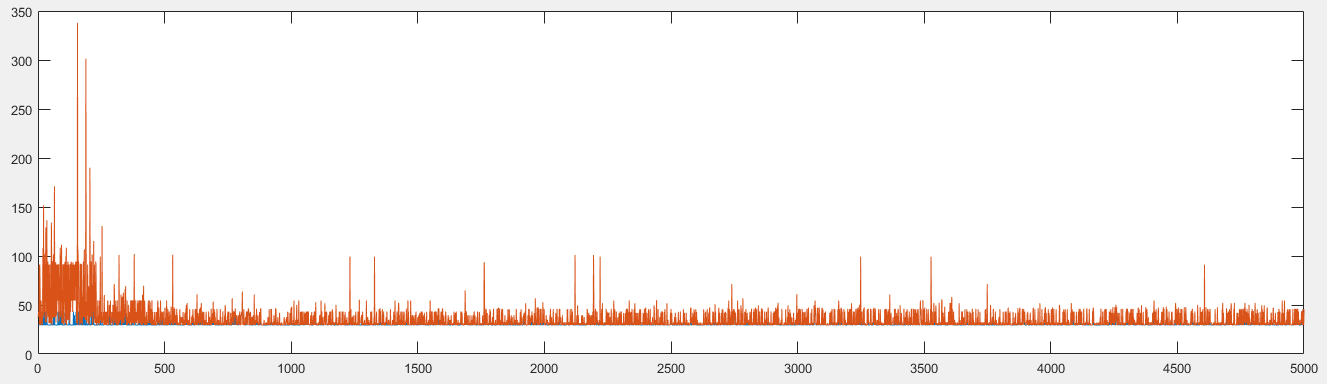


Fig. 4.a Average Latencies for random-arrival rate, beta=0.5,

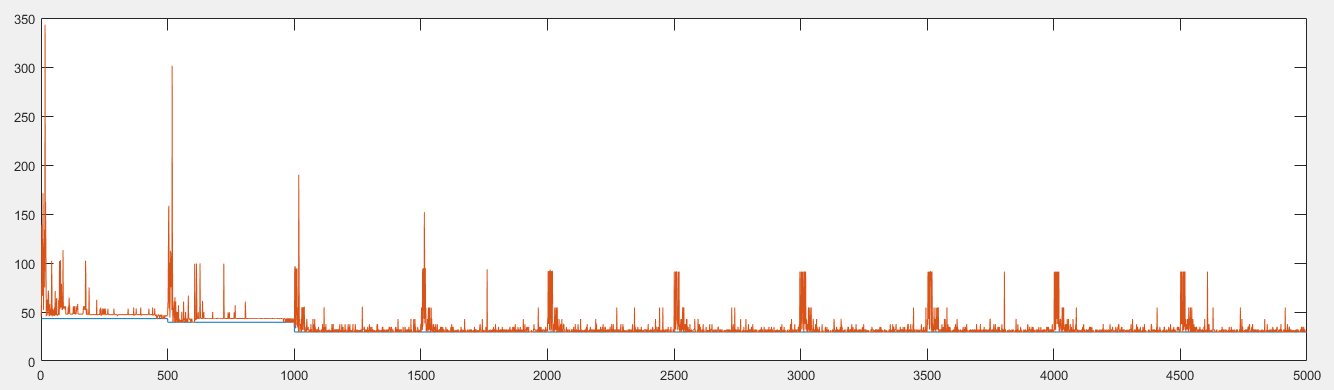


Fig 4.b Average Latencies for non-random-arrival rate, beta=0.5,

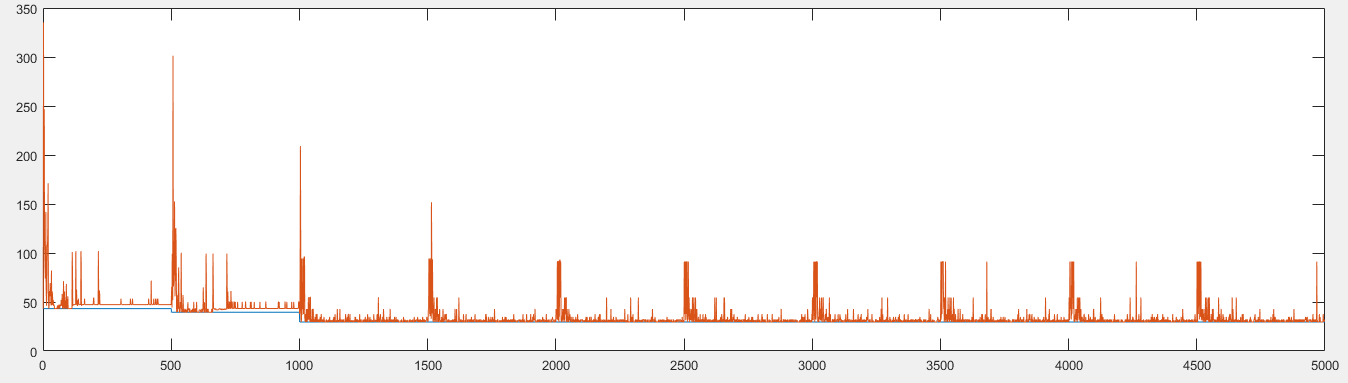


Fig 4.c Average Latencies for non-random arrival rates, beta=0.5, different seed for random numbers

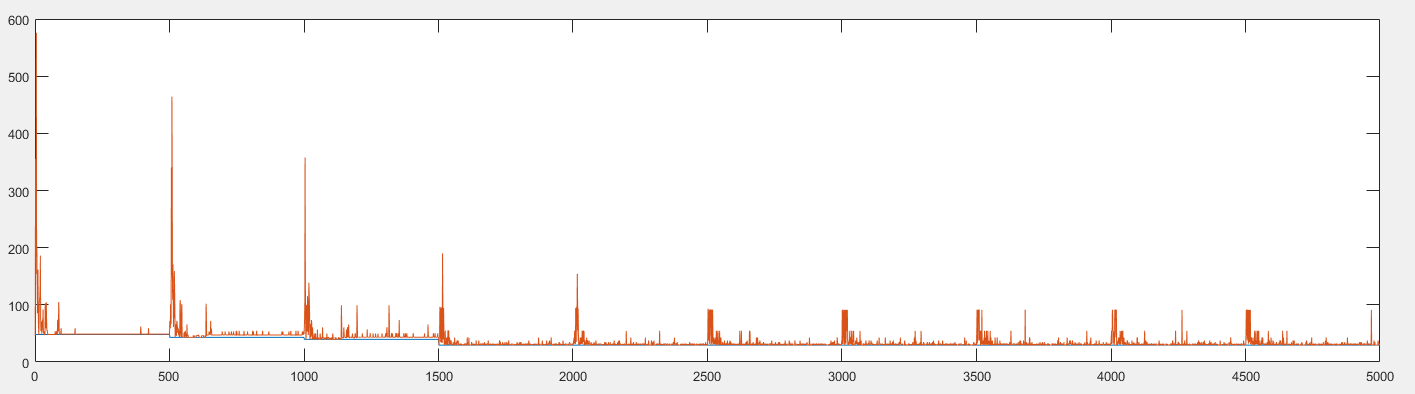


Fig 4.d Average Latencies for non-random arrival rates, beta=0.5,

**Chisquare for high arrival rate, exponential arrival rate for medium and erlang for low arrival rate**

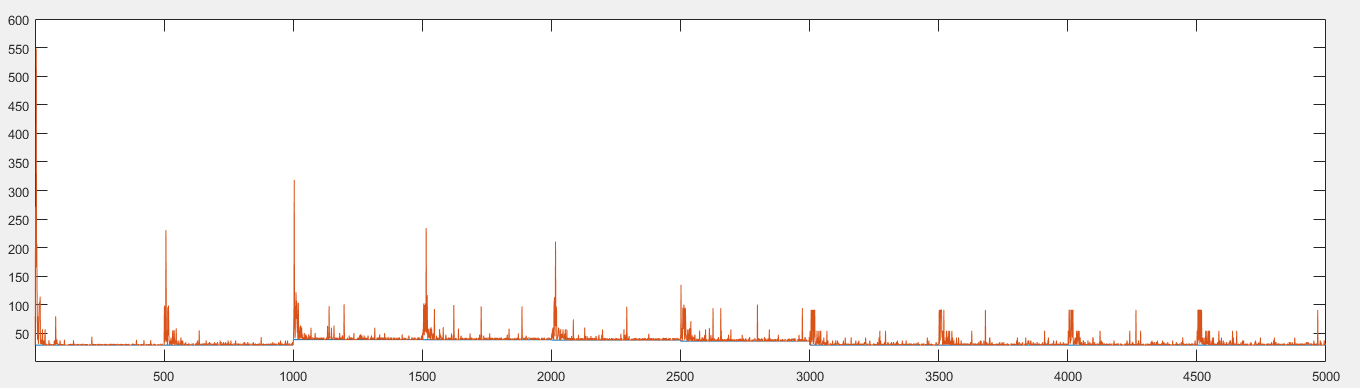


Fig 4.e Average Latencies for non-random arrival rates, beta=0.5,

**Erlang for High arrival rate, Exponential for Medium and Chisquare for low arrival rates**

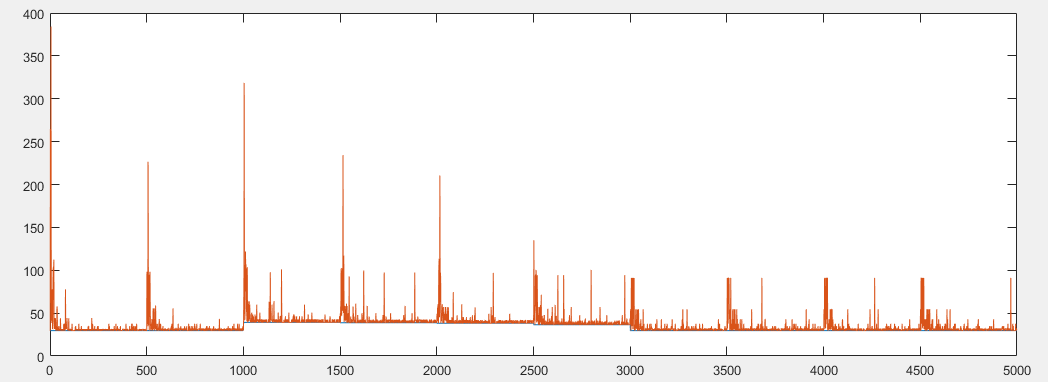


Fig 4.f Average latencies for non-random rates, beta=0.5

**Erlang for High Arrival rate, Chisquare for medium and Exponential for low arrival rates**

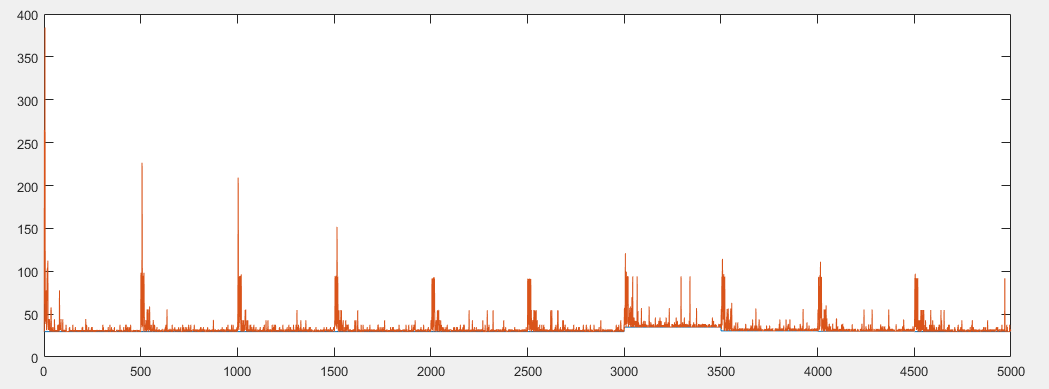


Fig.4.g Average Latencies for non-random rates, beta=0.5

**Exponential for high, Erlang for Medium and Chisquare for Low Arrival rates**

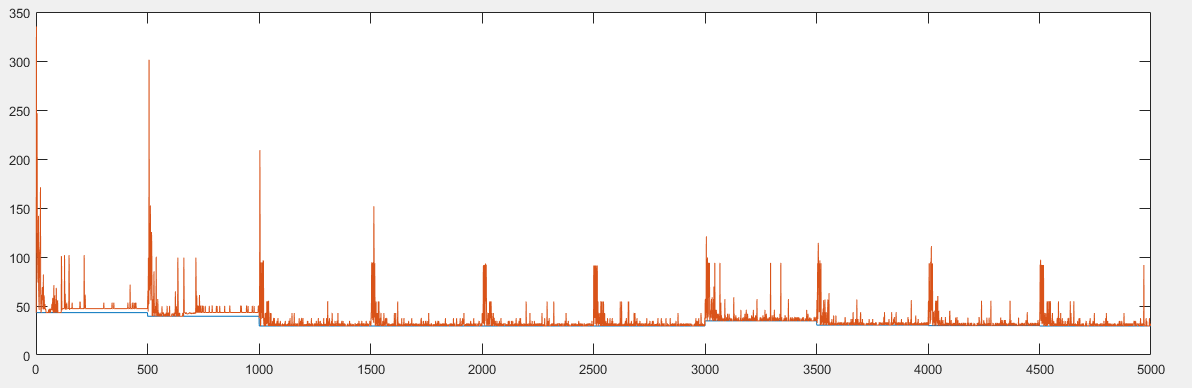


Fig 4.h Average Latencies for non-random rates, beta=0.5

**Exponential for High, ChiSquare for medium and exponential for low**

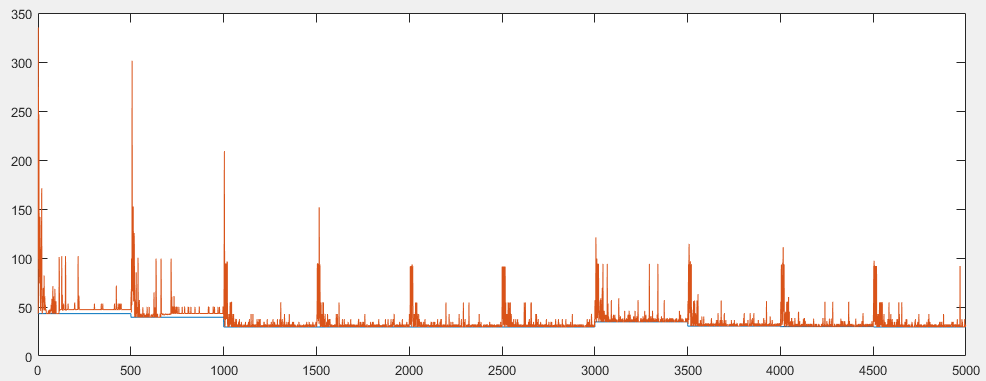


Fig. 4.g Average latencies for non-random arrival rates, beta=0.5

**Exponential for High, Normal for Medium and exponential for low**

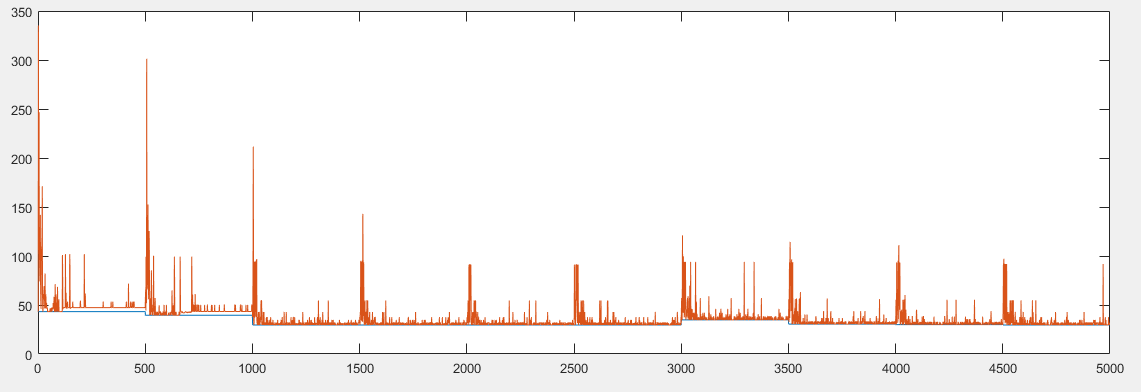


Fig4:h (i) Average Latencies for non-random arrival rates, beta=0.5

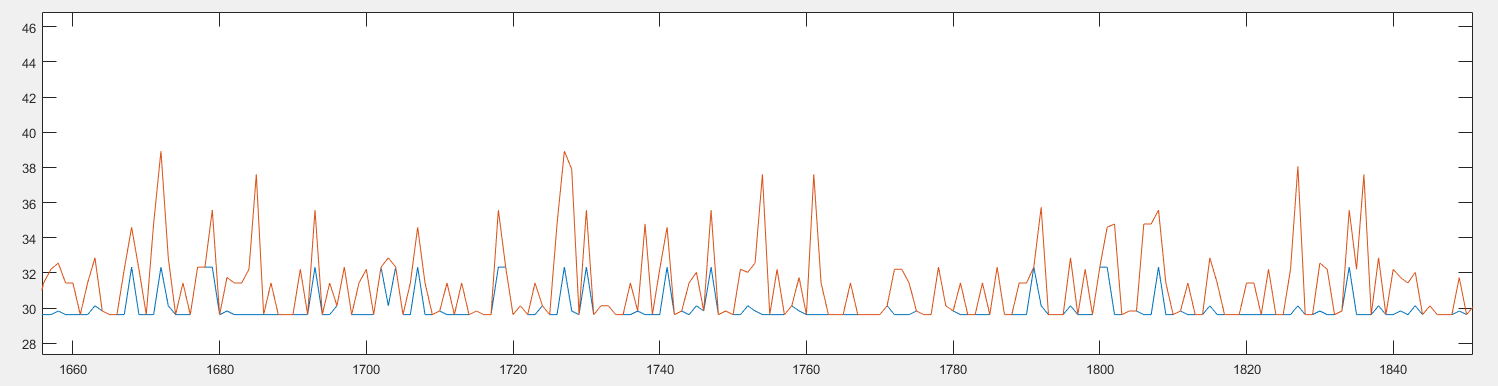


Fig 4h (ii) Average latencies for random arrival rates, beta =0.5, zoomed version

In this figure, we can see that RL always tends to reach the goodness of SC algorithm i.e sometimes it performs as good as the optimum configuration while sometimes it is worse than SC.

**Exponential for High, Uniform for medium and Chi-square for Low Inter-arrival rates**

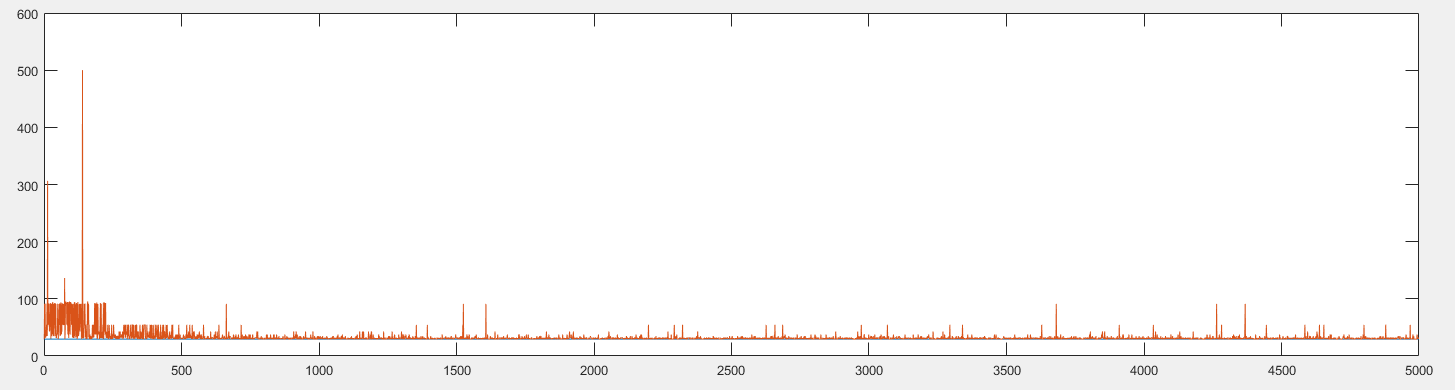


Fig. 4 I Average Latencies for random arrival rates, beta=0.5

**Uniform for High, Exponential for medium and Uniform for low arrival rates**

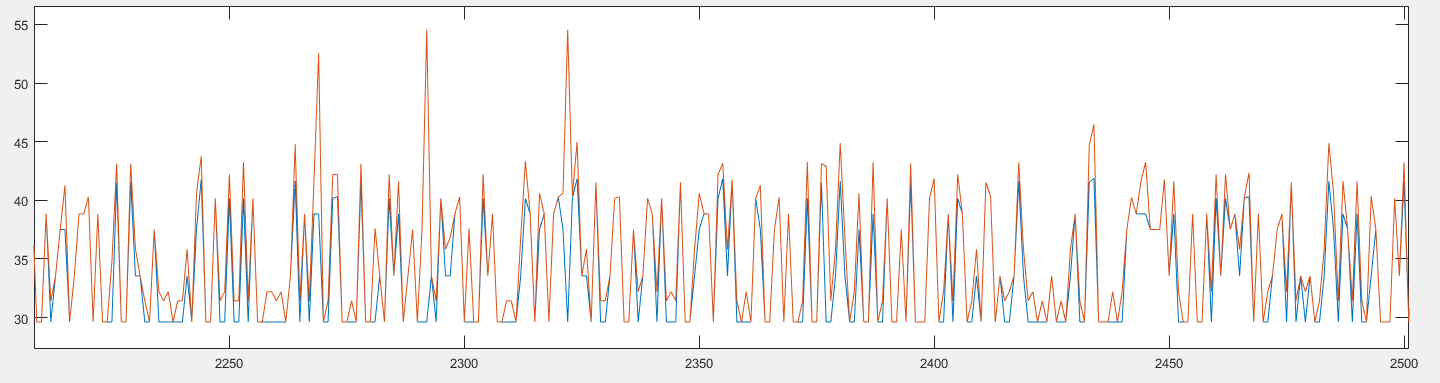
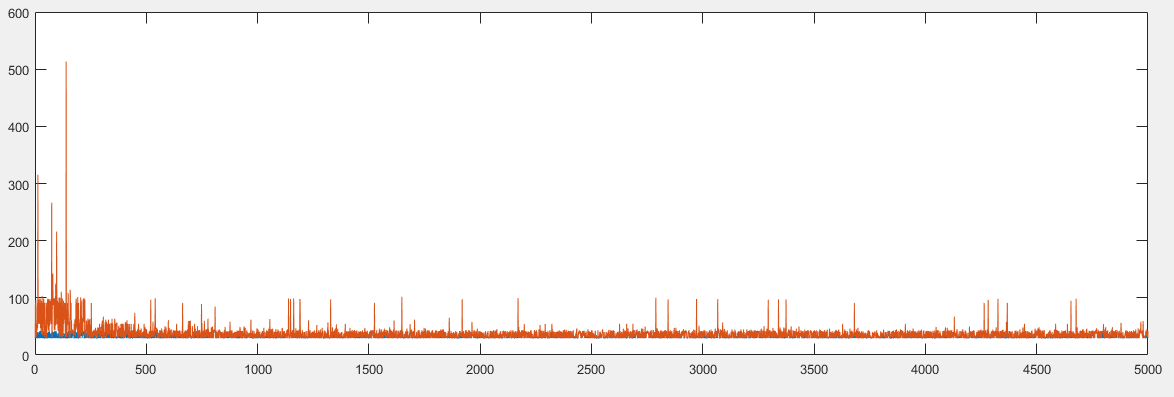


Fig 4.j Average latencies for random arrival rates, beta=0.5



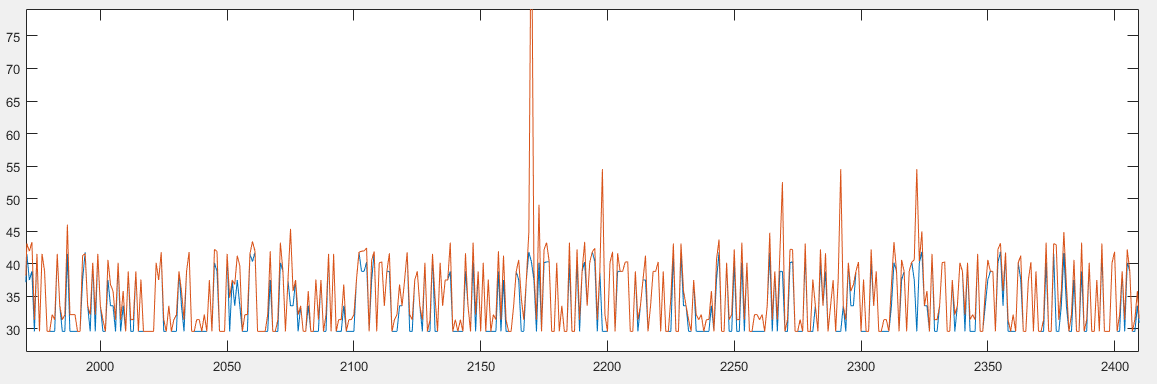
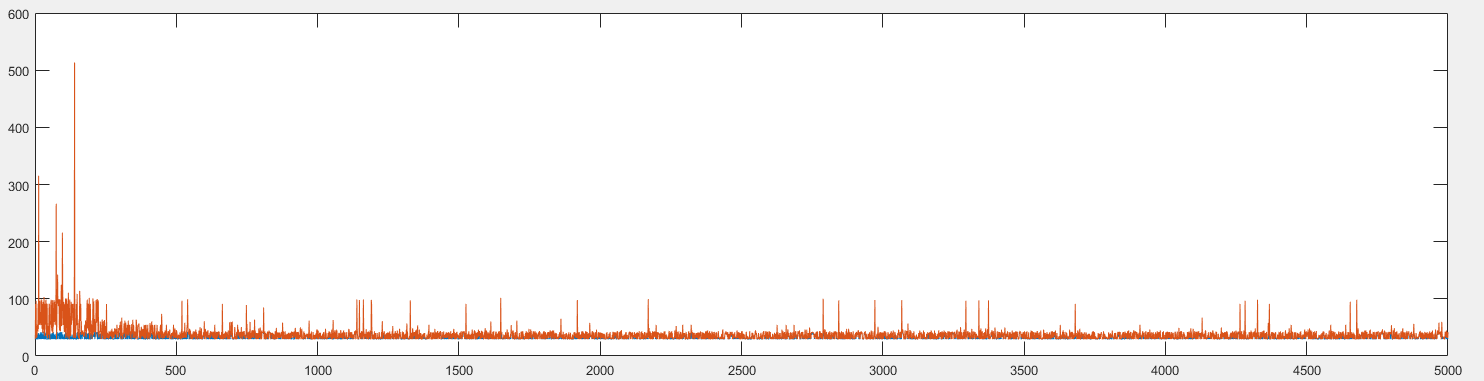


Fig 4. K Average latencies for random arrival rates, beta=0.5 (Un-zoomed and zoomed version)

**Normal for High, Exponential for Medium and Erlang for Low**



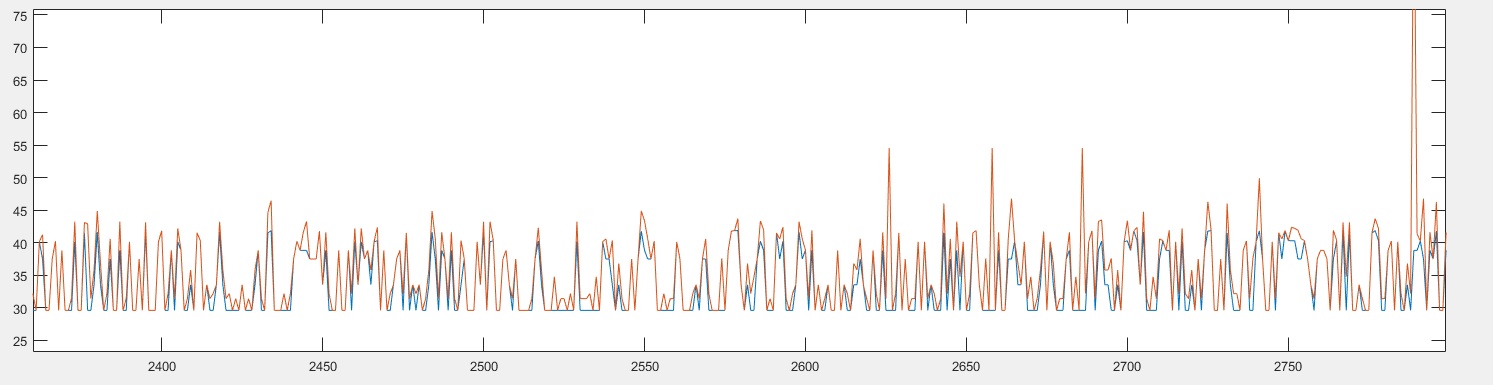


Fig 4.l Average latencies for random arrival rates, beta=0.5, (Un-zoomed and zoomed)

**Observations**:

From the above experiments, we can see that for various combinations of arrival rates, random or non-random, and for different learning rates beta, we do not see a single case where RL performs better than SC. It can be as good as SC by converging to the optimum configuration.

Why would RL not operate better than SC?  
We are running SC algorithm by first observing which configuration is optimum and then using that optimum configuration. Since, we are not using separate execution cycle for RL algorithm; the best configuration for RL would be the same as SC. Therefore, there is not a single case where RL outperforms the SC.