ASSIGNMENT #3

Alireza Mostafizi

I coded Bandit Algorithms in Python 2.7. Attached to the email you can find python code. I manipulated the code to generate the results for parts III and IV through commenting and uncommenting the desired lines. Thus, if you want to double-check the results, you need to comment or uncomment a few lines of the code. As before, please keep in mind that to have the code running you need to have "NumPy" installed for some mathematical operations. I also attached the Excel files which contain the charts used in part III and IV.

Part I: Create Bandit Algorithm

Basically, upon running, the program will ask you to input the SBRDs in a text file format. Just like attached bandits, the first line is the number of arms. Second line should be empty, and the following lines contain tuples which describe probabilities and rewards as suggested in the assignment. After entering the desired file, you need to choose the algorithm. You should enter "UNI" for Incremental Uniform, "UCB" for UCB, and "EPS" for E-Greedy algorithm. After that, you need to enter "N" as the number of steps you are interested in. For E-Greedy algorithm, you have to enter the Epsilon as well.

The results of each step will be shown in each line with the following format:

- "Next arm pull" "Best arm so far" "Cumulative Regret" "Simple Regret"

Also, at the end, you can see the counts of each arm pull.

This result help us check whether the algorithm has recognized the best arm (known by us) or not in two ways:

- The algorithm should report the real best arm as the best arm, based on the accumulated rewards
- The number of pulls for each arm, based on the algorithm, should be different regarding the probability and the rewards. In some algorithms, the better the arm is, the more it should be pulled.

The code has followed the suggested API.

Part II: Bandit Design

The requested Bandits are generated as the input text file format matching the program API. All three Bandits are attached to the email as well.

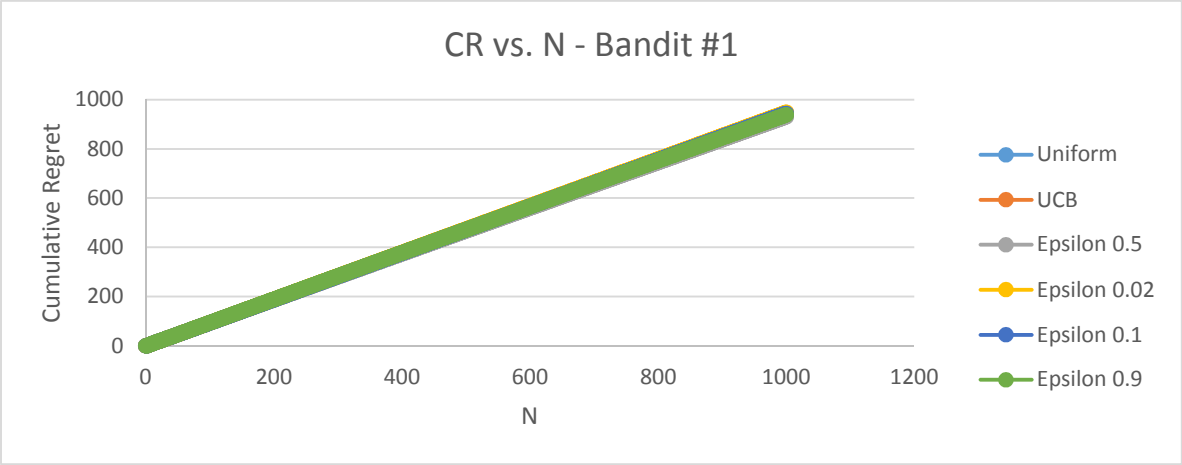
The designed Bandit is fairly the same as Bandit #1. The only difference is the last arm, which has higher probability of 0.7 to return 1. I chose this one mainly because it reveals some interesting behaviors of the algorithms. So:

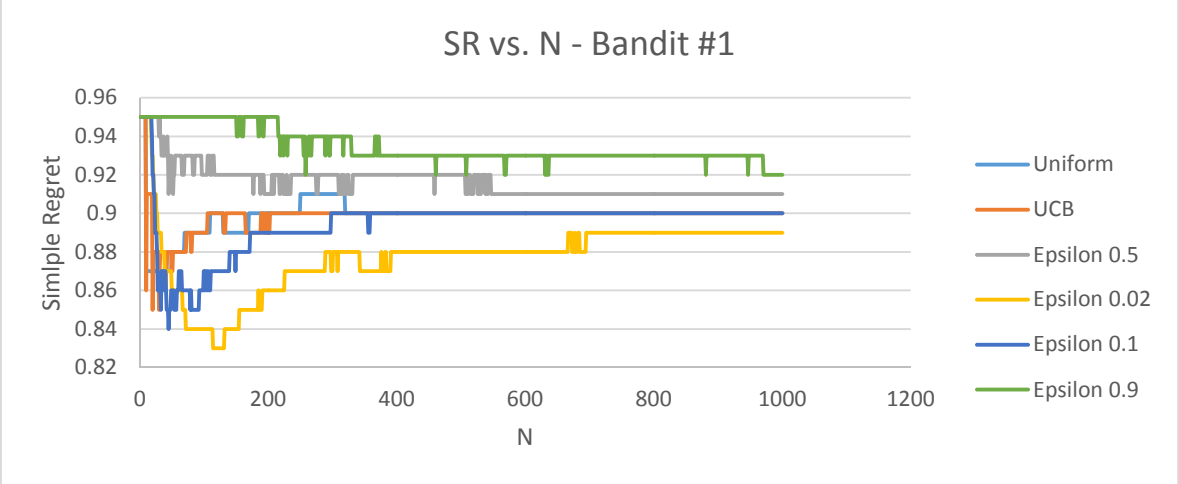
- It has 10 arms
- Nine of the arms return 0.05 for sure
- The last arm returns 1 with the probability of 0.7.
    - So, unlike the Bandit #1, the last arm is obviously and tangibly better than the other ones.

Part III and IV: Evaluating Cumulative and Simple Regret

I combined these two section to evaluate the algorithms in a better way. The algorithms are tested the same as requested for each pair of bandit and algorithm. The number of trials was set to 100 since it seemed that the data converged before that threshold, and the N was set to 1000 as most of the algorithms, in terms of finding the best arm and simple regret, converged by that step. The results of evaluation are as following:

Bandit #1

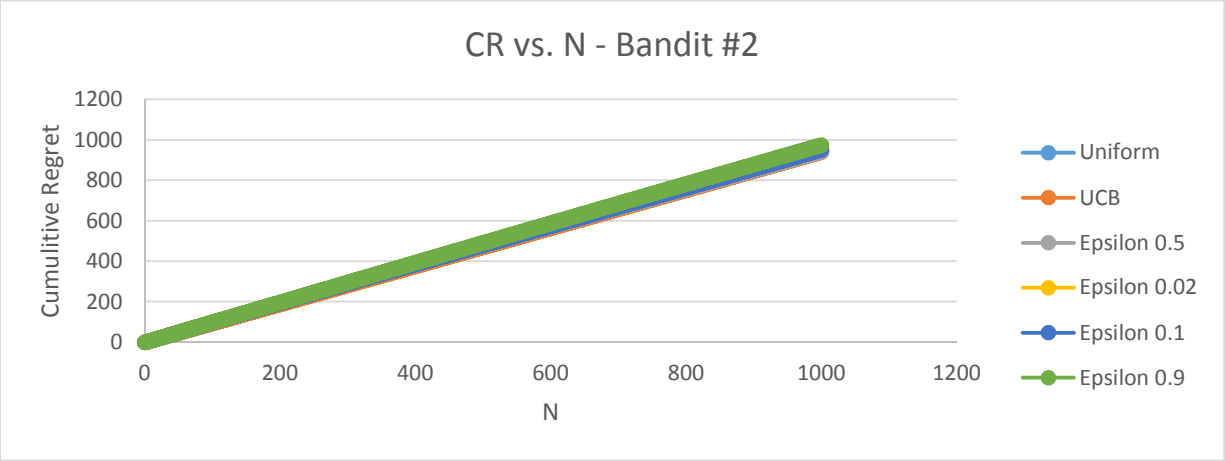


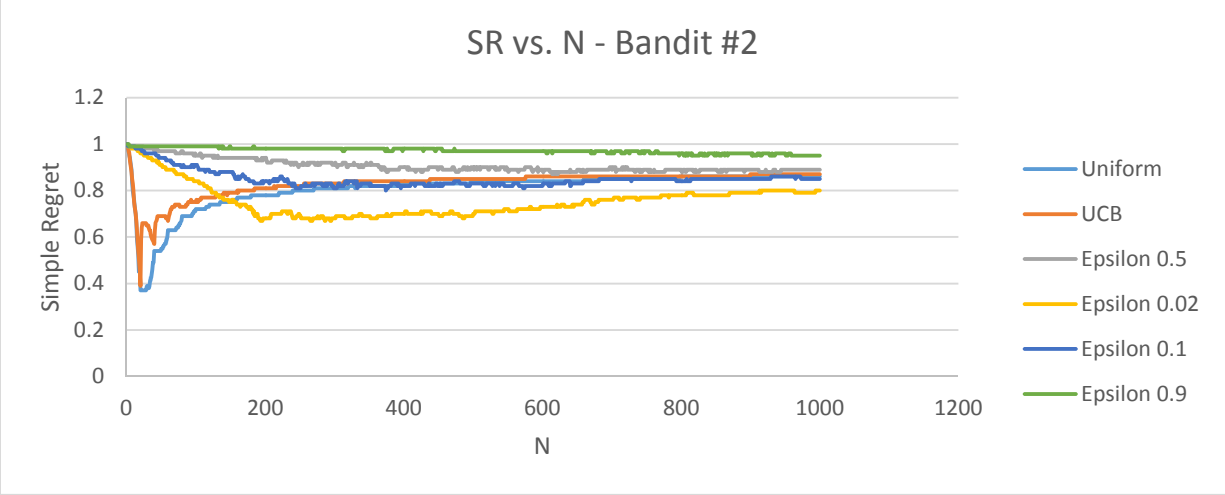


Cumulative Regrets for this bandit are almost the same with all the algorithms. The main reason is that most of the arms are exactly the same, and the other arm is almost the same as the others in terms of its expected reward. So, some of the algorithms may perform better in finding the best arm, but considering the cumulative regret, none of them has advantage over the other one since arms are fairly close in performance.

The Second graph shows Simple Regret versus the number of step. Based on this graph we can conclude that UCB algorithm and 0.5-Greedy algorithm are good at converging to the best arm faster than the other algorithms. I also tested 0.9-Greedy and 0.02-greedy algorithm which don't seem to be good in finding the best arm quickly. Besides, as expected, uniform and 0.1-greedy are behaving almost the same.

Bandit #2

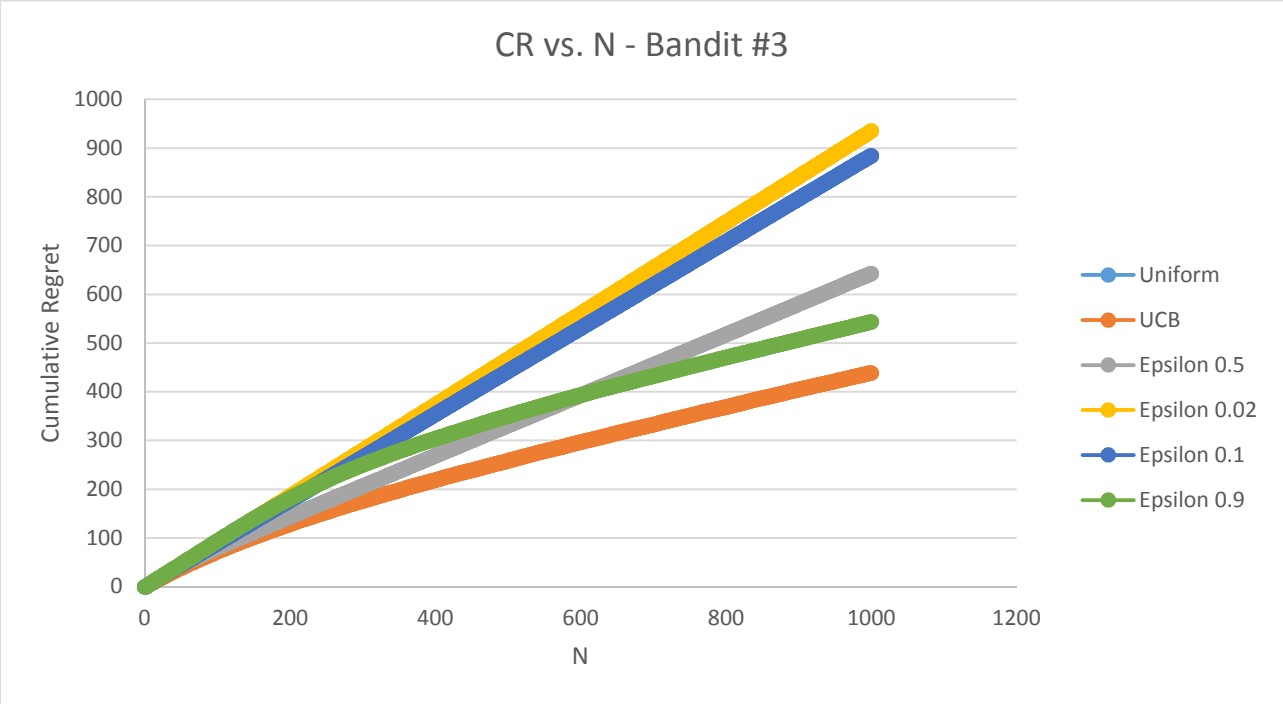


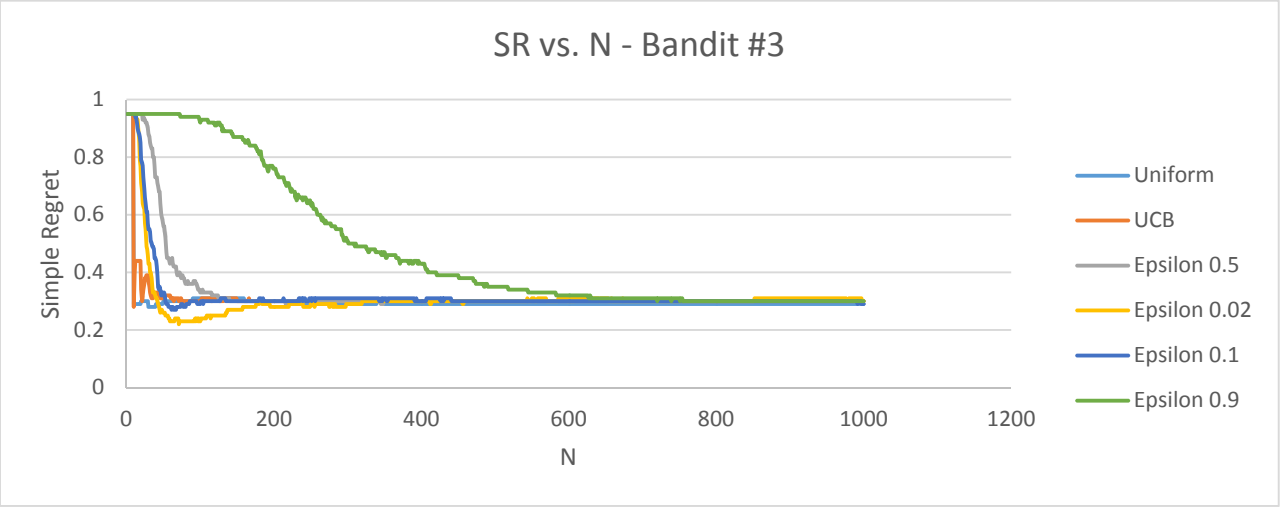


In this case, the arms are not the same, but the differences between them are very small in terms of expected value. This makes finding the best arms harder, and therefore, more undesired jumps between the arms. My intuition is that, since arms are close in performance, these undesired jumps result in an almost random behavior which causes the same cumulative, regardless of the algorithm.

Second graph shows 0.1-greedy converges to the solution sooner than the others. 0.5-greedy also shows a decent behavior in convergence. 0.9-greedy tends to remain in the state it is in, however, on the other hand, 0.02-greedy jumps from an arm to the other. Another interesting point is the behavior of uniform and UCB algorithm. They first diverge greatly, but then converge to the final state.

Bandit #3





In this case, since the best arm is pretty obvious, algorithms show interesting behavior. As expected, UCB and 0.9-Greedy show very good behavior in this scenario, rather than the others. Another interesting point is that, until some point (600) 0.5-greedy outweighs 0.9-greedy. But after that, 0.9-greedy has lower cumulative regret. This is mainly because of the fact that when 0.9-greedy finds the best arm, it sticks to it greatly, so it should have lower CR. This is confirmed by the second graph as well. As you can see, 0.9-greedy reaches convergence around N=600. It is not good in finding the best arm fast, but it is good in terms of cumulative reward. The other ones though, find the best arm faster.