CS 533 Homework 5
Investigating Bandit Algorithms and Objectives

Eugene Seo

05-31-2017

Part I: Create Bandit Algorithms

I implemented three bandit algorithms: Incremental Uniform, UCB, and ϵ-Greedy. (Run my python files: python Main.py bandit_num (e.g. python Main.py 1))

1. **Incremental Uniform**
   Incremental Uniform algorithm selects an arm through all arms. Thus, each arm has evenly equal number of pulls.

Algorithm 1 Incremental Uniform

```
def IncrementalUniform(bandit, num_pulls, T):
    for n in range(1, num_pulls+1):
        arm_index = n % num_arms - 1 # select an arm
        num_pulls_arm[arm_index] += 1
        r = bandit.Pull(arm_index) # pull the selected arm
        total_rewards += r

        # update average rewards
        old_mean = avg_rewards[arm_index]
        new_mean = old_mean + (r-old_mean)/num_pulls_arm[arm_index]
        avg_rewards[arm_index] = new_mean

        # update the best arm with the best average reward
        if new_mean > best_avg_reward:
            best_avg_reward = new_mean
            best_arm = arm_index

        # record the simple regret and cumulative regret
        sr = bandit.opt_reward - best_avg_reward
        cr = n * bandit.opt_reward - total_rewards
```

2. **UCB**

UCB algorithm chooses an arm based UCB values, calculated by sum of value and the degree of exploration of the arm. Thus, if some arms haven't been explored enough yet, then they would have a high change to be selected although their values are not close to the best.

Algorithm 2 UCB

```
def UCB(bandit, num_pulls, T):
    for n in range(num_arms, num_pulls):
        # Action choice by UCB after n pulls
        ucb = avg_rewards + np.sqrt(2*np.log(n)/num_pulls_arm)
        arm_index = np.argmax(ucb) # select an arm
        num_pulls_arm[arm_index] += 1
        r = bandit.Pull(arm_index) # pull the selected arm
        total_rewards += r
        ...
```

3. **ϵ-Greedy**
   ϵ-Greedy algorithms selects the best arm with probability ϵ, otherwise it chooses any arm randomly.

Algorithm 3 ϵ-Greedy

```
def EGreedy(bandit, num_pulls, E, T):
    # The best arm is selected with prob E, otherwise randomly select an arm
    prob = [E] + other_probs + [1-E-sum(other_probs)]
    for n in range(1, num_pulls+1):
        arms_order = [best_arm] + other_arms
        arm_index = np.random.choice(arms_order, 1, p=prob)[0] # select an arm
        num_pulls_arm[arm_index] += 1
        r = bandit.Pull(arm_index) # pull the selected arm
        total_rewards += r
        ...
```

In addition to bandit algorithms, I built bandit class (Bandit.py) to design bandits. In this bandit class, it defines the parameters of each arm (reward and probability), and it has a Pull(a) function to return a reward after pulling the corresponding arm.

Algorithm 4 Bandit Class

```
class Bandit:
    def __init__(self, n):
        self.num_arms = n
        self.param_arms = np.zeros((n,2))
        self.opt_reward = 0

    def NumArms(self):
        return(self.num_arms)

    def SetParams(self, a, r, p):
        self.param_arms[a][0] = r
        self.param_arms[a][1] = p
        self.opt_reward = np.max(self.param_arms[:,0] * self.param_arms[:,1])

    def Pull(self, a):
        reward = self.SBRD(self.param_arms[a][0], self.param_arms[a][1])
        return reward[0]

    def SBRD(self, r, prob):
        r_list = [r, 0.0]
        reward = np.random.choice(r_list, 1, p=[prob, 1-prob])
        return reward
```

Part 2: Bandit Design

I used three bandits with different parameters.

1. **Bandit 1**
   Bandit 1 has 10 arms, and the last arm (10) generates the best reward 1 with probability 0.1. Other arms always give low reward 0.05. Thus, the best arm should be arm 10 for bandit 1.

Arm ID	1	2	3	4	5	6	7	8	9	10
Reward	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	1
Prob	1	1	1	1	1	1	1	1	1	0.1

Table 1: The parameters of Bandit 1 arms

2. **Bandit 2**
   Bandit 2 has 20 arms, and each arm has a slightly incremental reward with same probability, 0.1. Thus, the best arm should be the last arm 20, which could give the best reward 1 with same probability.

Arm ID	1	2	3	4	5	6	7	8	9	10
Reward	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
Prob	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Arm ID	11	12	13	14	15	16	17	18	19	20
Reward	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1
Prob	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Table 2: The parameters of Bandit 2 arms

3. **Bandit 3**
   Bandit 3 has 10 arms, and each arm can equally generate the best reward 1 with different probability. Thus, the best arm should be the arm 10, which has the highest probability to generate the best reward.

Arm ID	1	2	3	4	5	6	7	8	9	10
Reward	1	1	1	1	1	1	1	1	1	1
Prob	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0

Table 3: The parameters of Bandit 3 arms

Part 3: Evaluating Cumulative Regret

Part 4: Evaluating Simple Regret

I run algorithms with N = 2000000, T = 5 for bandit1 and T = 10 for bandit 2,3. In this report, I show the final graph averaging each of the T trials.

1. Bandit 1

- **Incremental Uniform**
  With Incremental Uniform algorithm, the cumulative regret for bandit 1 keeps increasing. It's because every arms take turn to be pulled so that they have an equal chance to be selected. It causes linearly cumulated regret from other poor arms.

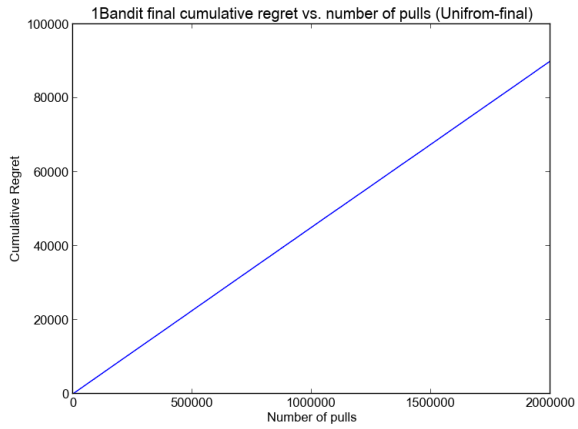

Figure 1: Bandit 1 final cumulative regret - uniform

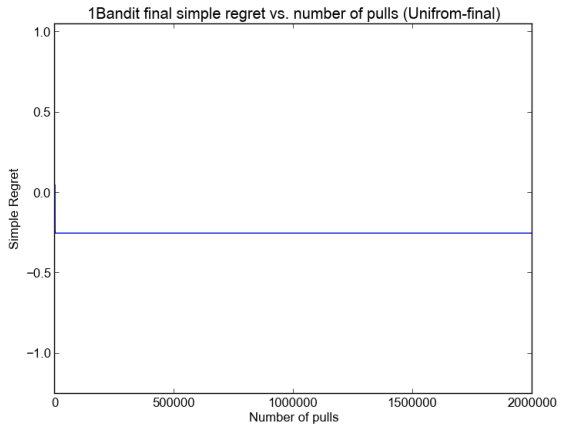

Figure 2: Bandit 1 final simple regret - uniform

T	1	2	3	4	5
Best Arm ID	10	10	10	10	10

Table 4: The best arms returned from Incremental Uniform algorithm for Bandit 1

- **UCB**

  With UCB algorithm, the cumulative regret for bandit 1 converges to a certain point. By learning enough pulls with exploration behavior, this algorithm eventually can capture the best arm, generating better rewards later on. It makes the cumulative rewards converge.

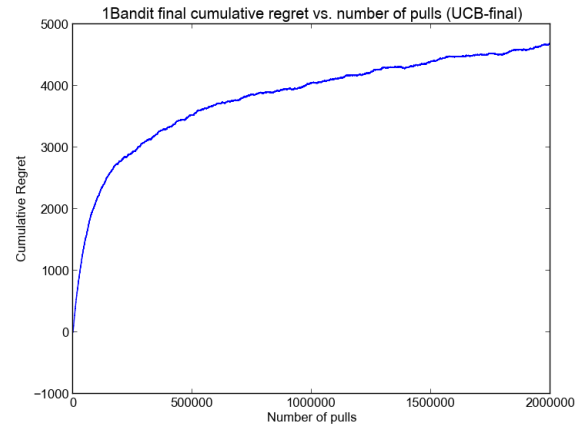

Figure 3: Bandit 1 final cumulative regret - UCB

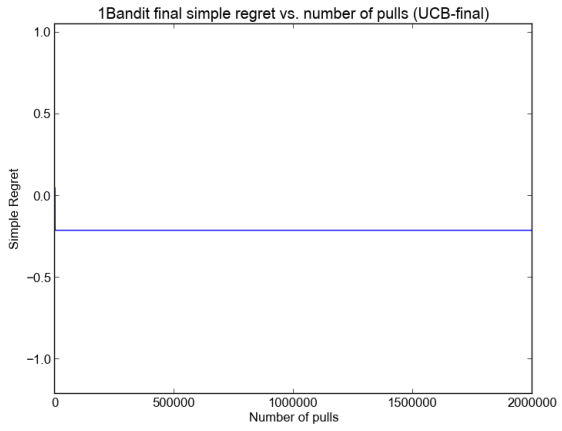

Figure 4: Bandit 1 final simple regret - UCB

T	1	2	3	4	5	6	7	8	9	10
Best Arm ID	10	10	10	10	10	10	10	10	10	10

Table 5: The best arms returned from UCB algorithm for Bandit 1

- **ϵ-Greedy**

  ϵ-Greedy algorithm also makes the cumulative regret linear. I think it's because the half chance of random (poor) arm pulls make the cumulative rewards to be linear.

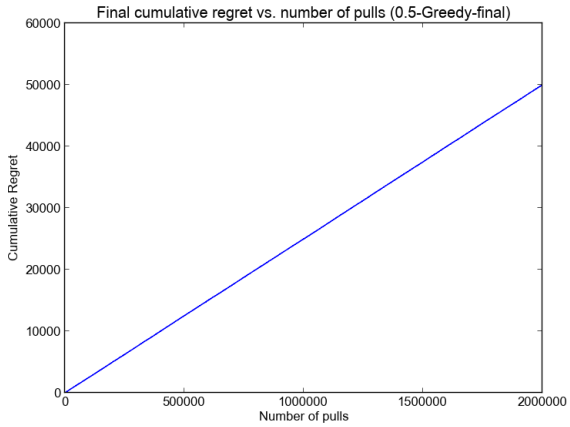

Figure 5: Bandit 1 final cumulative regret - 0.5-greedy

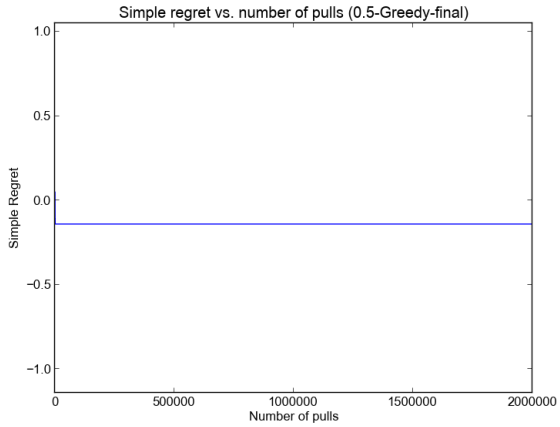

Figure 6: Bandit 1 final simple regret - 0.5-greedy

T	1	2	3	4	5	6	7	8	9	10
Best Arm ID	10	10	10	10	10	10	10	10	10	10

Table 6: The best arms returned from 0.5-Greedy algorithm for Bandit 1

2. Bandit 2

- **Incremental Uniform**

  Similarly, Incremental Uniform algorithms generates the linearly increasing cumulative rewards due to the even chance of pulling of all arms.

Figure 7: Bandit 2 final cumulative regret - uniform

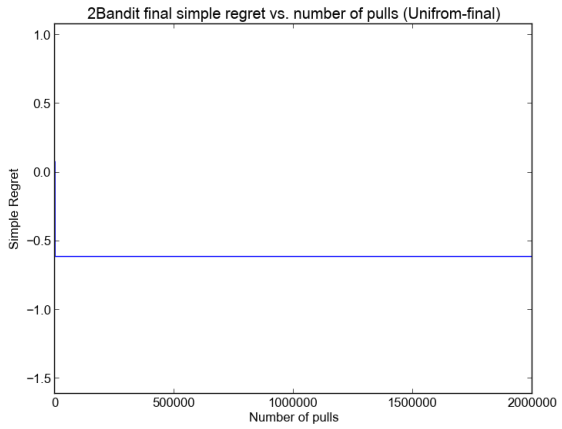

Figure 8: Bandit 2 final simple regret - uniform

T	1	2	3	4	5
Best Arm ID	19	15	13	14	18

Table 7: The best arms returned from Incremental Uniform algorithm for Bandit 1

- **UCB**

  In bandit 2, UCB algorithm also guarantees to converge the cumulative regret since it learns enough pulls with exploration behavior, However, comparing to bandit 1, the speed of convergence is a little slower because bandit 2 does not have extremely different arms' features while bandit 1 has extremely different two types of arms.

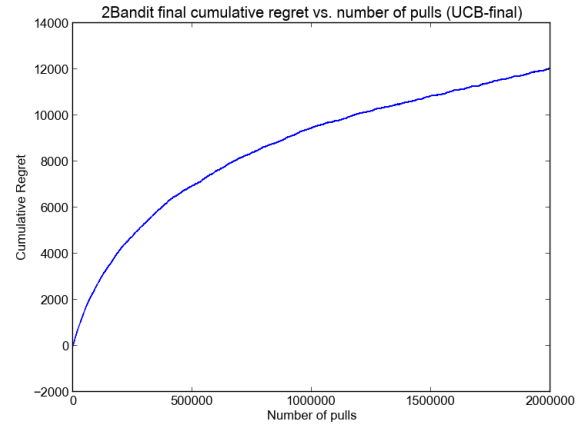

Figure 9: Bandit 2 final cumulative regret - UCB

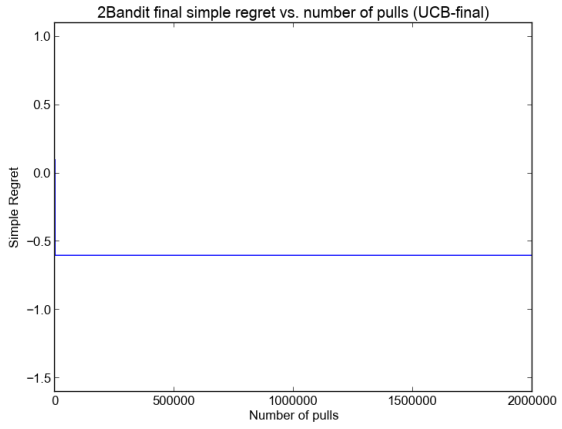

Figure 10: Bandit 2 final simple regret - UCB

T	1	2	3	4	5	6	7	8	9	10
Best Arm ID	7	11	20	18	16	20	18	16	18	17

Table 8: The best arms returned from UCB algorithm for Bandit 2

- **ϵ-Greedy**

  Similarly, ϵ-Greedy algorithm for bandit 2 also shows linearly increasing the cumulative rewards due to the half chance of random pulling.

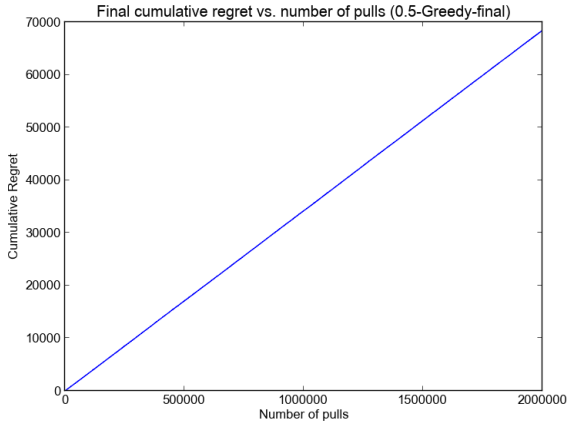

Figure 11: Bandit 2 final cumulative regret - 0.5-greedy

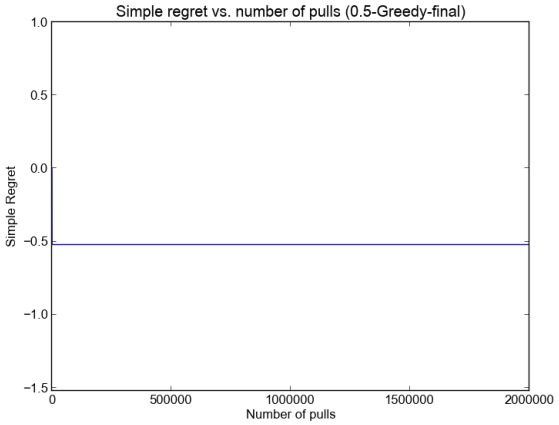

Figure 12: Bandit 2 final simple regret - 0.5-greedy

T	1	2	3	4	5	6	7	8	9	10
Best Arm ID	17	18	5	19	12	18	16	18	20	18

Table 9: The best arms returned from 0.5-Greedy algorithm for Bandit 2

3. Bandit 3

- **Incremental Uniform**

  Similarly, Incremental Uniform algorithms generates the linearly increasing cumulative rewards due to the even chance of pulling of arms.

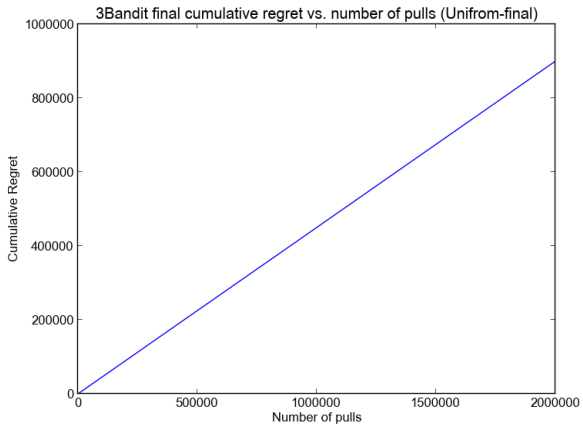

Figure 13: Bandit 3 final cumulative regret - uniform

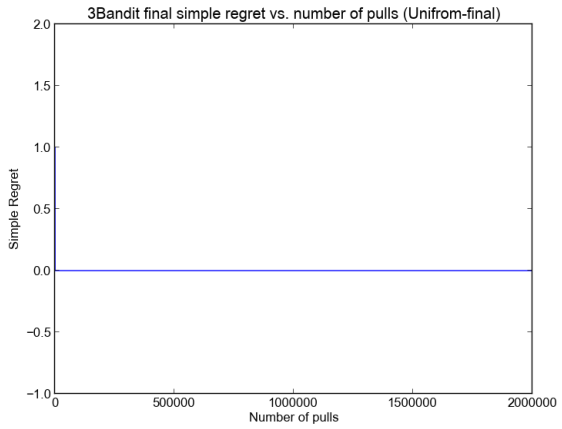

Figure 14: Bandit 3 final simple regret - uniform

T	1	2	3	4	5
Best Arm ID	3	6	2	4	3

Table 10: The best arms returned from Incremental Uniform algorithm for Bandit 2

- **UCB**

  In bandit 3, UCB algorithm converges the fastest and the magnitude of regrets are also very small among three bandits. It happens because all arms have the best rewards with different probability. However, the algorithms easily can make confuse to find the best arms since all could make the best reward.

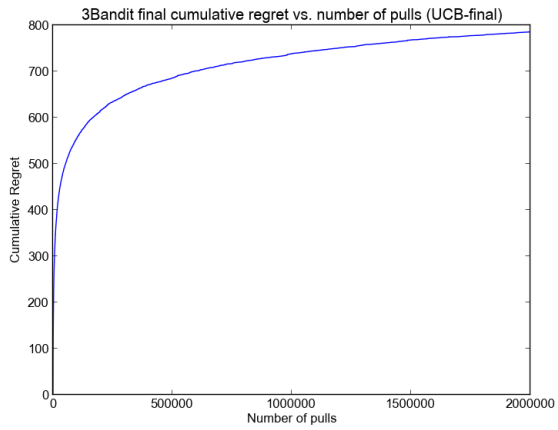

Figure 15: Bandit 3 final cumulative regret - UCB

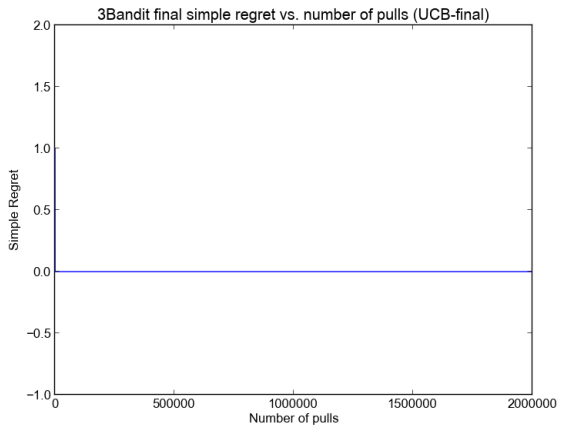

Figure 16: Bandit 3 final simple regret - UCB

T	1	2	3	4	5	6	7	8	9	10
Best Arm ID	6	3	3	3	4	4	3	5	4	5

Table 11: The best arms returned from UCB algorithm for Bandit 3

- **ϵ-Greedy**

  Similarly, ϵ-Greedy algorithm for bandit 2 also shows linearly increasing the cumulative rewards due to the half change of random pulling.

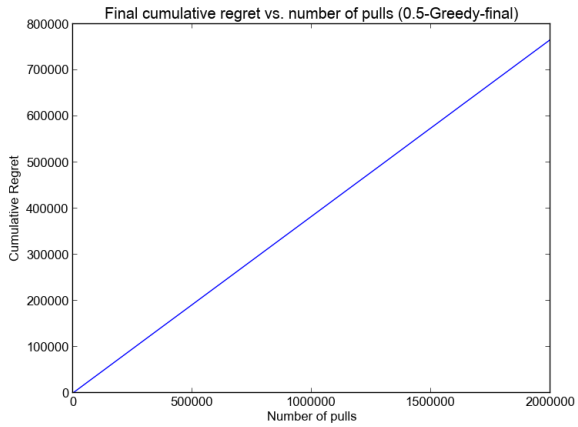

Figure 17: Bandit 3 final cumulative regret - 0.5-greedy

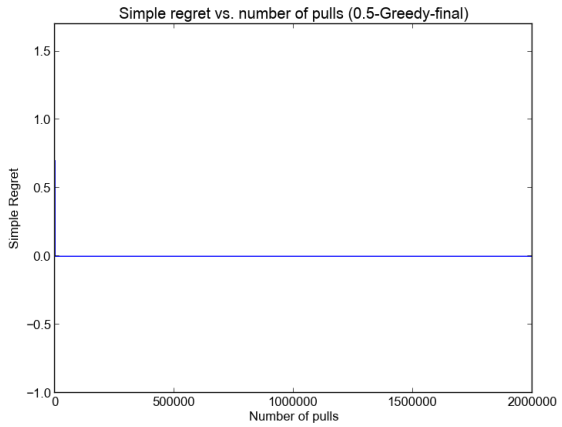

Figure 18: Bandit 3 final simple regret - 0.5-greedy

T	1	2	3	4	5	6	7	8	9	10
Best Arm ID	6	6	1	4	10	4	9	10	10	10

Table 12: The best arms returned from 0.5-Greedy algorithm for Bandit 3

4. **Final regrets comparison with all algorithms for each bandit**

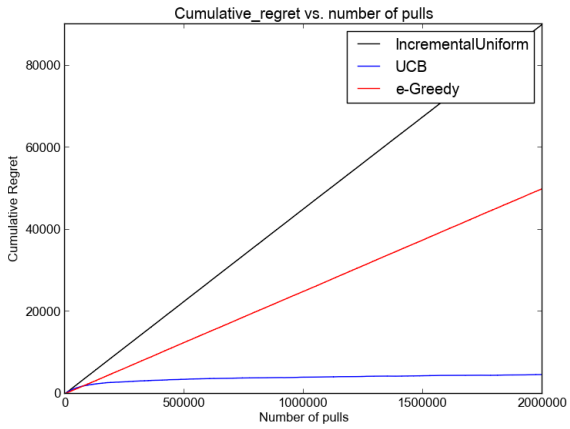

Figure 19: cumulative regret - bandit 1

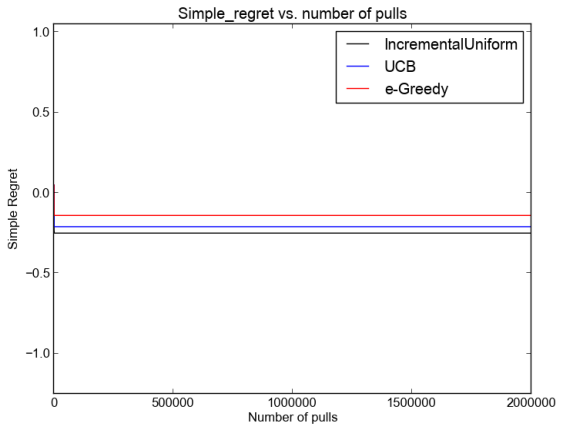

Figure 20: simple regret - bandit 1

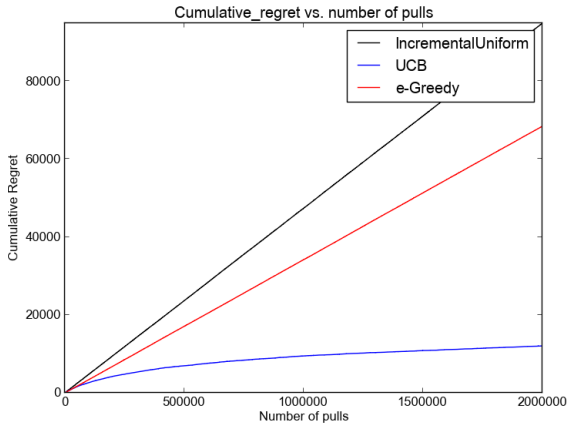

Figure 21: cumulative regret - bandit 2

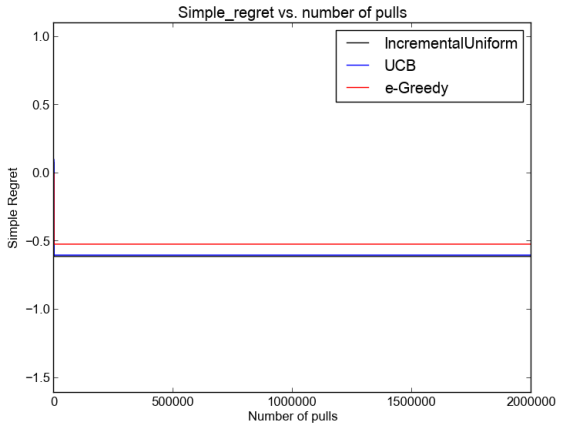

Figure 22: simple regret - bandit 2

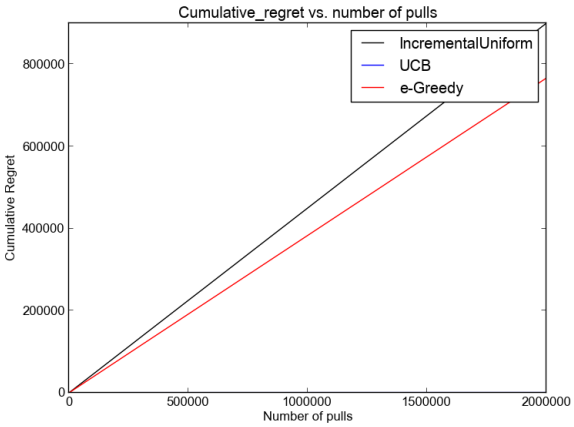

Figure 23: cumulative regret - bandit 3

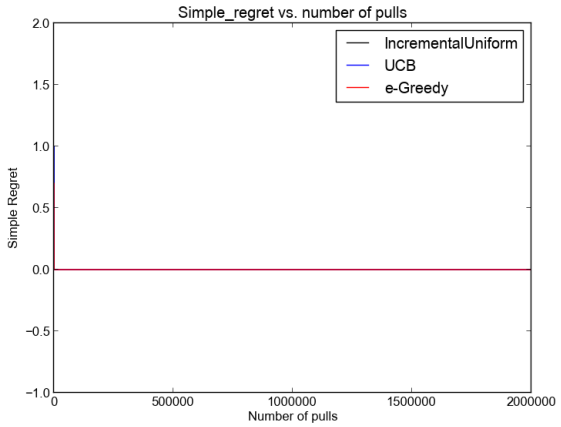

Figure 24: simple regret - bandit 3

- Analysis of cumulative regrets
  Overall, UCB algorithm shows lower cumulative regrets for the three bandits.

Since UCB algorithm more focuses on the best arms while Incremental Uniform algorithm takes even chance to select the best arm and 0.5-Greedy also has a half chance of selecting other arms, UCB is able to have lower cumulative rewards from repeated pulls of the best arm.

- Analysis of simple regrets

  If our concern is not the overall cumulative regrets but finding a good arm quickly, then random based arm selection, which is 0.5-greedy algorithm, is the worst since it could end up pulling not best arms over and over. UCB algorithm also too much focuses on current best arm before exploring other arms, so it would take more time to know the real best arm. Therefore, in this case, Incremental Uniform algorithms can be the best algorithm to figure out the instant best arm by pulling arms through all possible arms. For bandit 1, it has very distinctive arms, so Incremental Uniform algorithm is enough to run to know the best arm. However, bandit 2 and 3 has somehow similar features of arms, so even Incremental Uniform algorithm cannot work best as it does in bandit 1.