CS6700 Programming Assignment 1

Sabarinath N P *(CS10B020)*

Department of Computer Science and Engineering
IIT Madras

August 2013

Contents

1.1 Introduction

N-armed Bandit problems are the simplest non-associative, evaluative feedback problem that can be used to analyze the fundamental Reinforcement Learning concepts. This reports analyzes and presents the results obtained by simulating N-armed Bandit problems with four *Action-Value* Methods.

- Greedy
- ϵ-greedy
- Soft Max Action Selection
- UCB Algorithm

Each algorithm is simulated by varying its parameters over a range. In addition to the above analysis of the Bandit problem, the number of arms and number of steps simulated in the Bandit problem is varied and optimal parameter is determined for each case.

1.2 Implementation

For the purpose of experimental analysis, the true action-value *(*q(i)) of each arm is pre-determined by sampling *Standard normal distribution.* Different standard normal deviate is used for different bandit problems. Python snippet used to sample $N(0, 1)$

$$q(a) = random.gauss(\mu, \sigma)$$

In this assignment, the values of μ and σ are fixed to 0, 1 respectively. The estimated action-value $Q_0(i)$ of each arm of all the games is initialized to 0. The estimated value after t steps of algorithm is estimated using the following formula.

$$Q_{i+1}(a) = Q_i(a) + \frac{R_i^{'} - Q_i(a)}{i}$$

The reward of each arm $R_i^{'}$ is calculated as follows,

$$R_i^{'} = R_i + random.gauss(0, 1)$$

where R_i is the actual reward and $random.gauss(0,\ 1)$ is the noise term as the true value of each arm is not know from the problem perspective. (If they are known then, it is trivial solution to choose the optimal arm).

The above mentioned four algorithms different only in the process of choosing next arm to play. The method employed in those algorithm to choose best (or randomnly) arm is described below.

1.2.1 Greedy

At each step of the Bandit problem, the arm that maximizes the average reward is played. That is, the arm i^* with maximum estimated reward after t^{th} steps is played at $(t + 1)^{th}$ step. This is a pure Greedy statregy and there is no randomness or exploratory moves involved.

$$i^* = \underset{i}{\operatorname{argmax}}\, Q_i$$

1.2.2 ϵ-greedy

This algorithm introduces randomness or probability of making exploratory moves which is a crucial part in learning. At each step, with probability $\epsilon \ll 1$, randomn exploratory move is played. That is, with probability $1-\epsilon$ the Greedy strategy is followed.

$$i^* = \begin{cases} \text{argmax}\, Q_i & \text{if } rand() > \epsilon \\ randint(0,\ N-1) & \text{otherwise} \end{cases}$$

where N is the number of arms.

1.2.3 Soft Max Action Selection

In this method, the algorithm does a weighted sampling on the normalized exponential probabilities calculated from average estimated values of the arms, after each step. This helps is better than greedy as this has non-zero probability to make a exploratory move.

$$P_t(i) = \frac{e^{Q_t(a)/\tau}}{\sum_{i=1}^{N} e^{Q_t(i)/\tau}}$$

where τ is the temperature parameter. Using the probability calculated above, the weighted sampling is done.

1.2.4 UCB Algorithm

In this method, the exploratory moves are implicitly played by the dynamics of the algorithm. The algorithm chooses the arm i^* that maximizes the sum of average estimated value and the first order moment.

$$i^* = \underset{i}{\text{argmax}}\, Q_i + \sqrt{\frac{2 * \ln T}{t_j}}$$

where T is the total steps played in game, t_j is the number of steps played by arm i.

The Generic simulator runs any given algorithm for different number of arms N and total steps T. But the number of Bandit games over which the values are averages is kept constant at 2000 Three cases simulated, namely,

- T = 1000 and N = 10
- T = 5000 and N = 10
- T = 1000 and N = 1000

1.3 Observation-Analysis

This section explains the different results obtained by simulated the algorithms with varying parameters.

1.3.1 ϵ-Greedy

This *action-value* method is run against several values of ϵ namely [0.0, 0.01, 0.02, 0.05, 0.1, 0.15, 0.20] for the three cases mentioned above. For the all the three cases, the following trend is witnessed.
"*The average reward over all Bandit games formed Unimodal curve over an increasing set of ϵ values*" The values of ϵ for which the average value and % optimality action is maximized for the three cases is listed below.

- T = 1000 and N = 10
  - maximum average value is obtained at ϵ=0.05
  - maximum optimality action is obtained at ϵ=0.1
- T = 5000 and N = 10
  - maximum average value is obtained at ϵ=0.01
  - maximum % optimality action is obtained at ϵ=0.05
- T = 1000 and N = 1000
  - maximum average value is obtained at ϵ=0.1
  - maximum % optimality action is obtained at ϵ=0.3

It can be observed clearly that with increase in number of steps T the optimal ϵ value decreases. This is attributed by the fact that exploratory moves result in relatively low reward values when the number of steps completed in the Bandit game is high. Hence low epsilon value is good after large number of steps.

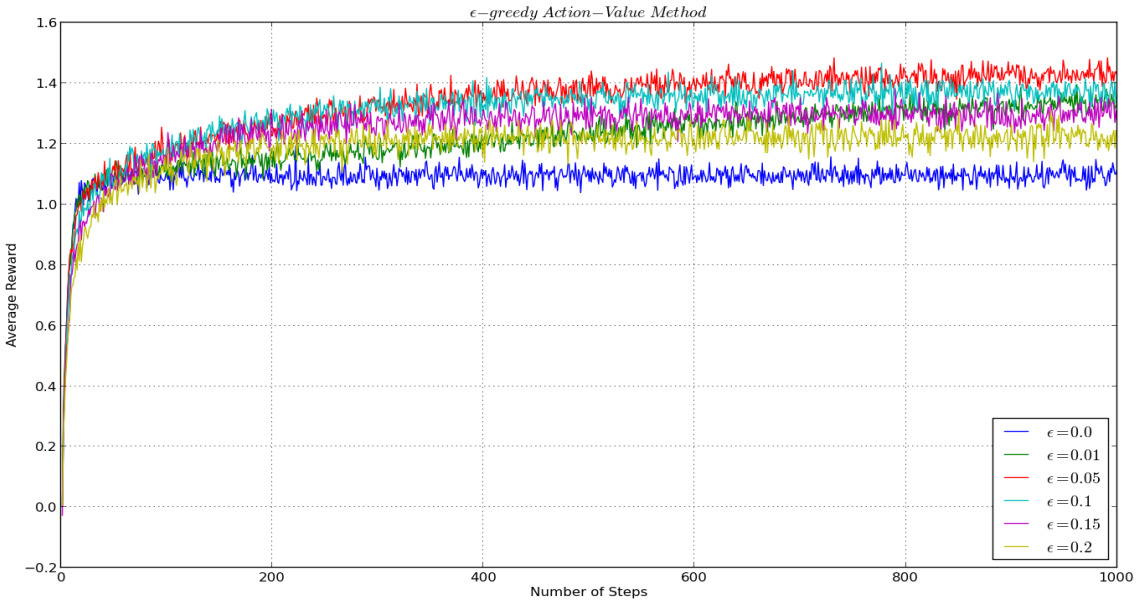
$$\underset{\epsilon}{\operatorname{argmax}} \sum_{i=1}^{N} \frac{Q_i}{N} \propto \frac{1}{T}$$

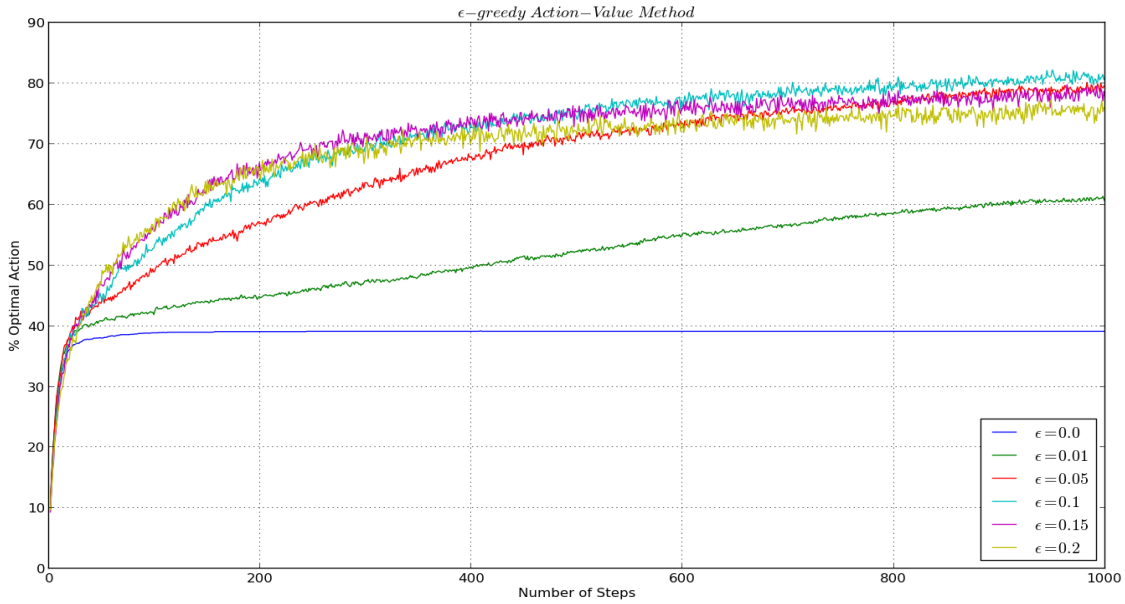
When N is fixed.

The optimality, as expected is high for low values of ϵ when the number of steps increases. This is clear from the fact that after large number of steps, it is difficult to choose optimal arm when ϵ value is high, though the good arms are known by the dynamics of the game.

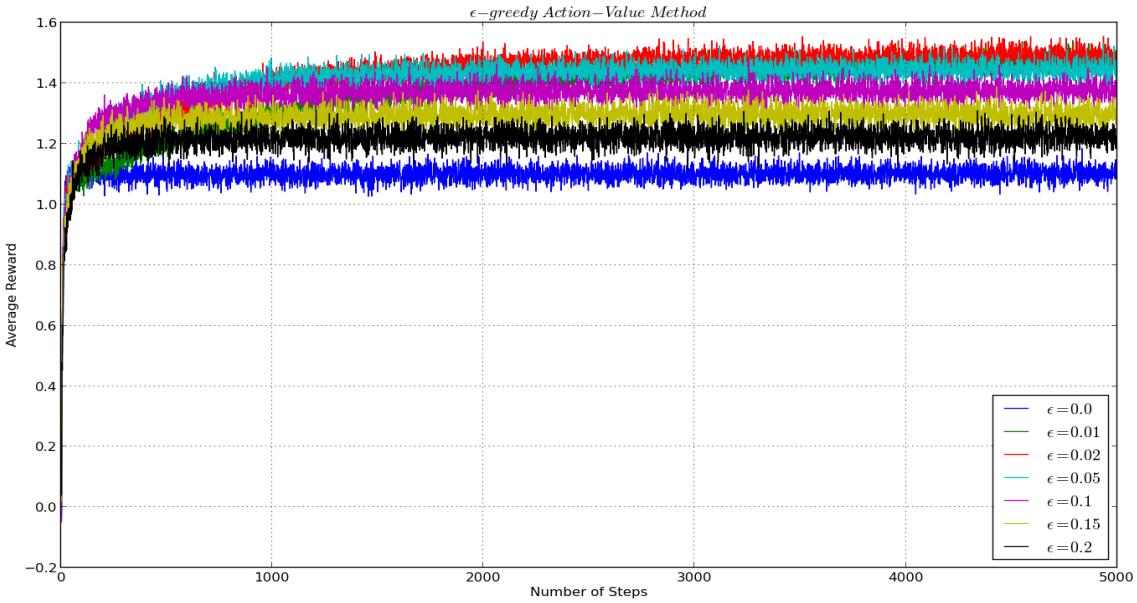
When the number of arms N increases, a change in behaviour of the algorithm can be noticed. It performs better for higher values of ϵ when T is kept constant. This is mainly because exploratory moves play a vital role in playing good arms. So low values of ϵ has high probability of not choosing the globally optimal (but sticks to local optimum) move when number of actions or arms is very high. This fact is strengthened by the fact high ϵ values result in playing the optimal action more frequently. Given below, the observed Graphs

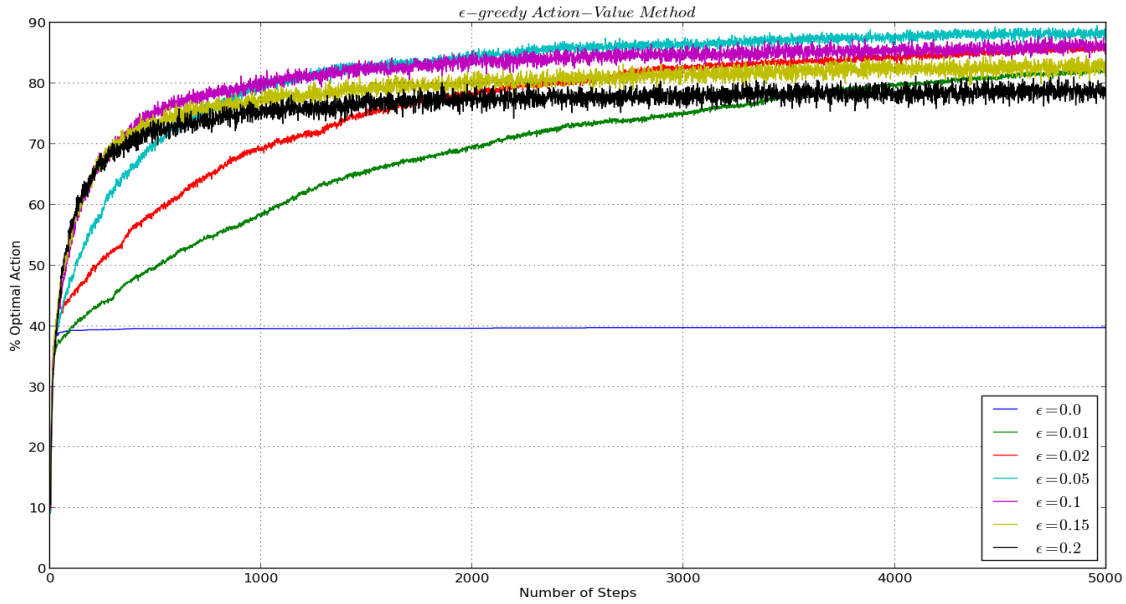
- T = 1000 and N = 10



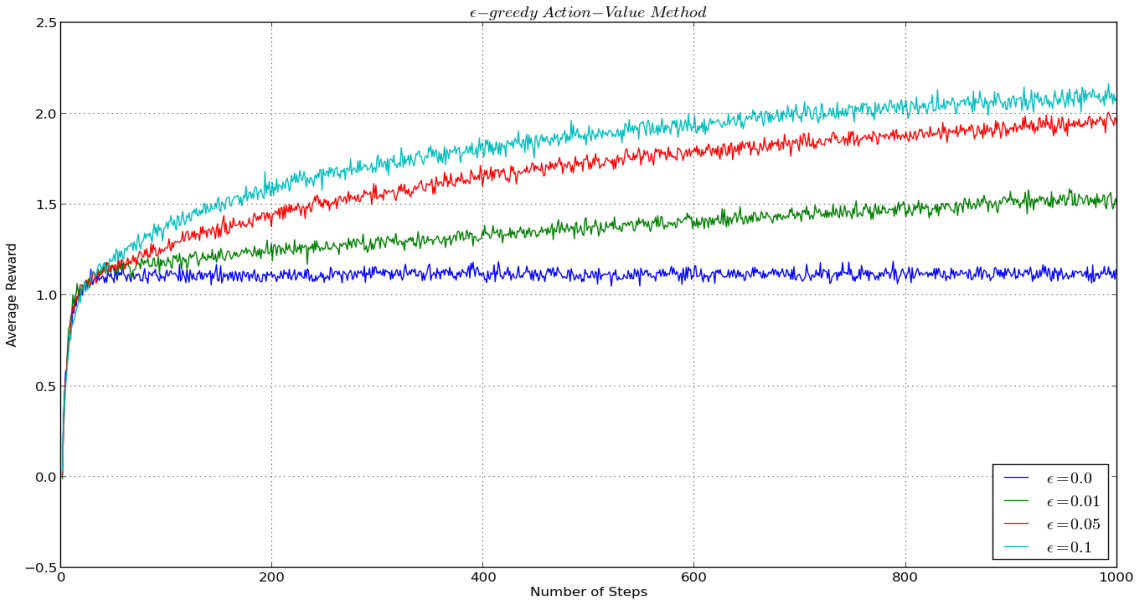


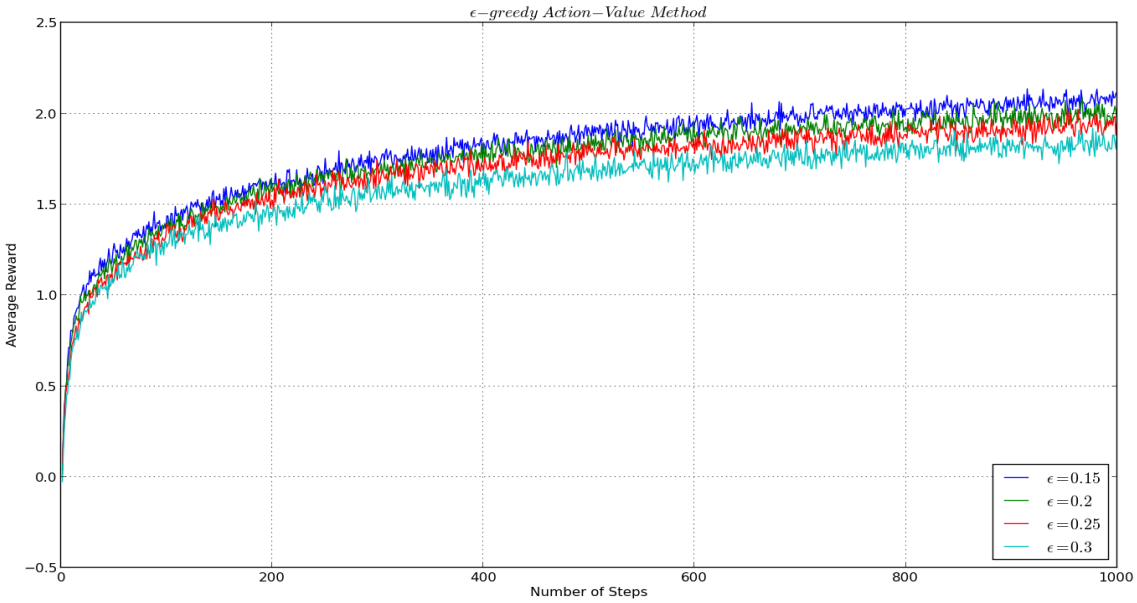
- T = 5000 and N = 10

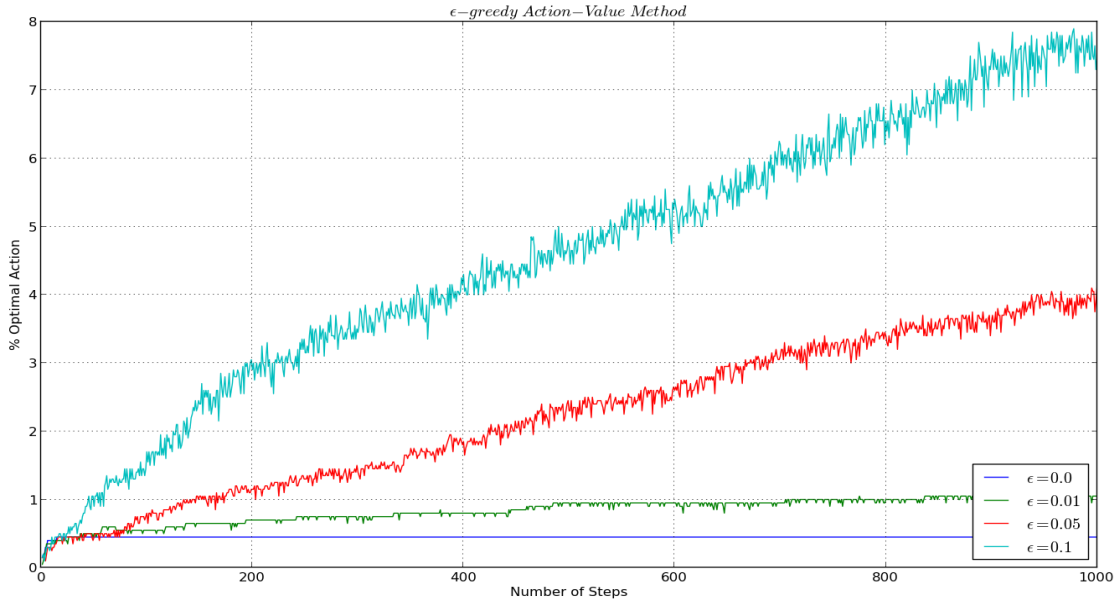


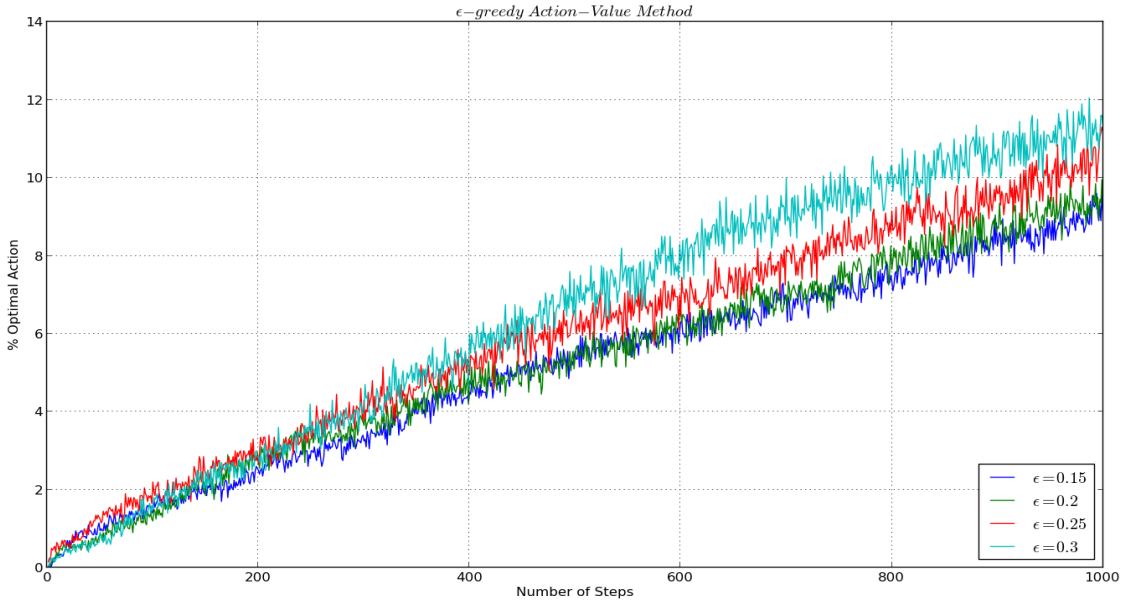


- T = 1000 and N = 1000






$\epsilon-greedy\ Action-Value\ Method$
% Optimal Action
Number of Steps
$\epsilon=0.0$
$\epsilon=0.01$
$\epsilon=0.05$
$\epsilon=0.1$


$\epsilon-greedy\ Action-Value\ Method$
% Optimal Action
Number of Steps
$\epsilon=0.15$
$\epsilon=0.2$
$\epsilon=0.25$
$\epsilon=0.3$

1.3.2 Soft max Action Selection

This *action-value* method is run against several values of Temperature namely [0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3] for the three cases mentioned above. For the all the three cases, the following trend is witnessed.
"*The average reward over all Bandit games formed Unimodal curve over an increasing set of Temperature values*" The values of Temperature for which theaverage value and % optimality action is maximized for the three cases is listed below.

- T = 1000 and N = 10
  - maximum average value is obtained at τ=0.25
  - maximum optimality action is obtained at τ=0.2
- T = 5000 and N = 10
  - maximum average value is obtained at τ=0.15
  - maximum % optimality action is obtained at τ=0.2
- T = 1000 and N = 1000
  - maximum average value is obtained at τ=0.25
  - maximum % optimality action is obtained at τ=0.3

It can be observed clearly that with increase in number of steps T the optimal τ value decreases. This is attributed by the fact that exploratory moves result in relatively low reward values when the number of steps completed in the Bandit game is high. Hence low epsilon value is good after large number of steps.

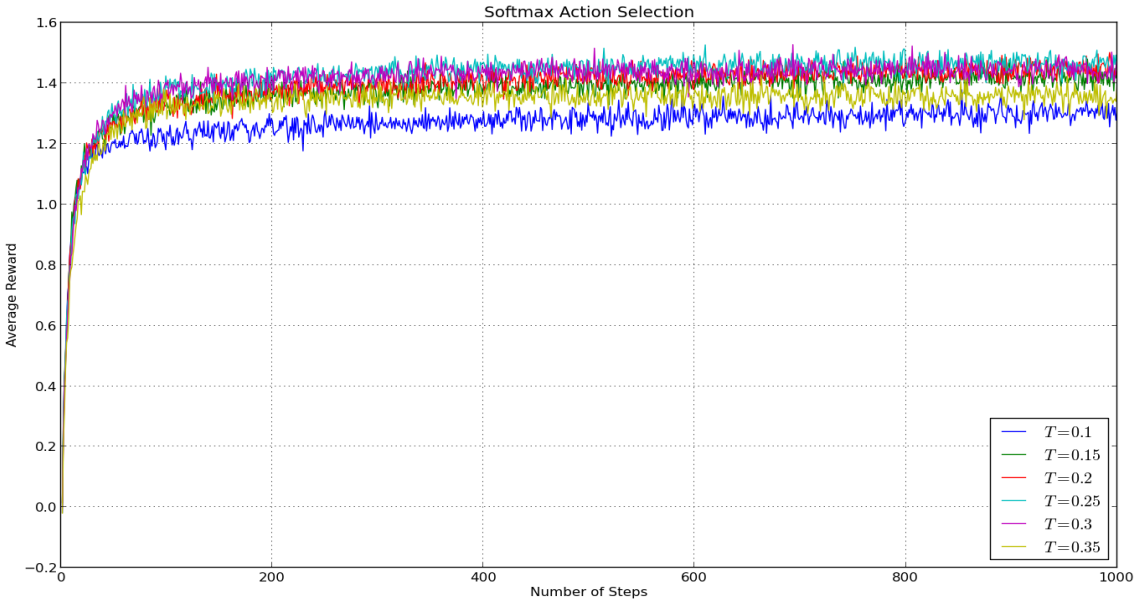
$$\underset{\tau}{\text{argmax}} \sum_{i=1}^{N} \frac{Q_i}{N} \propto \frac{1}{T}$$

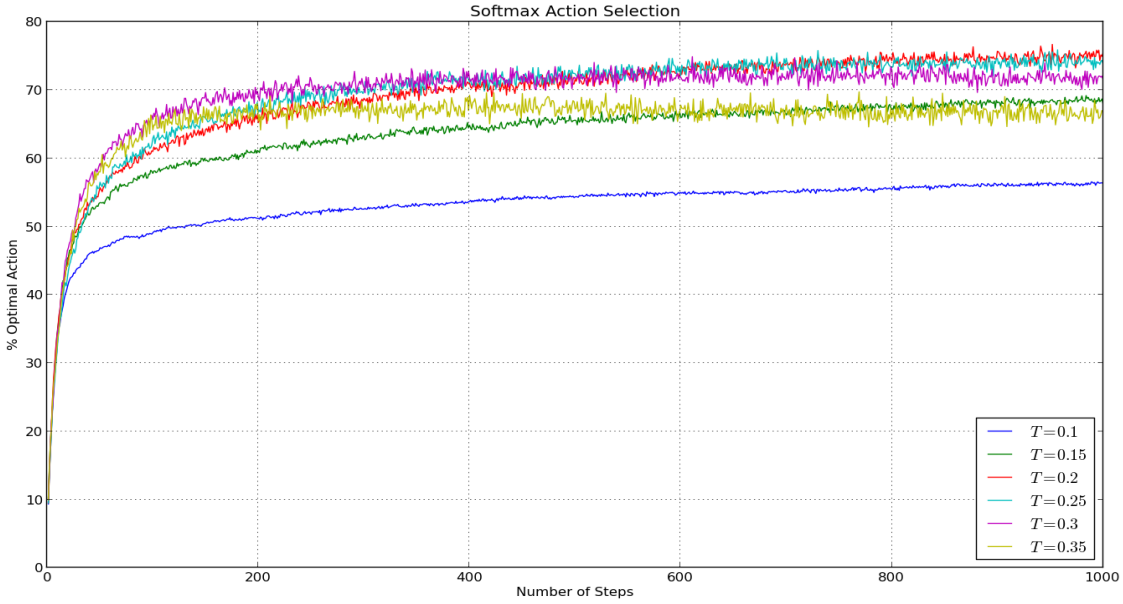
When N is fixed.

The optimality, as expected is high for low values of τ when the number of steps increases. This is clear from the fact that after large number of steps, it is difficult to choose optimal arm when τ value is high, though the good arms are known by the dynamics of the game.

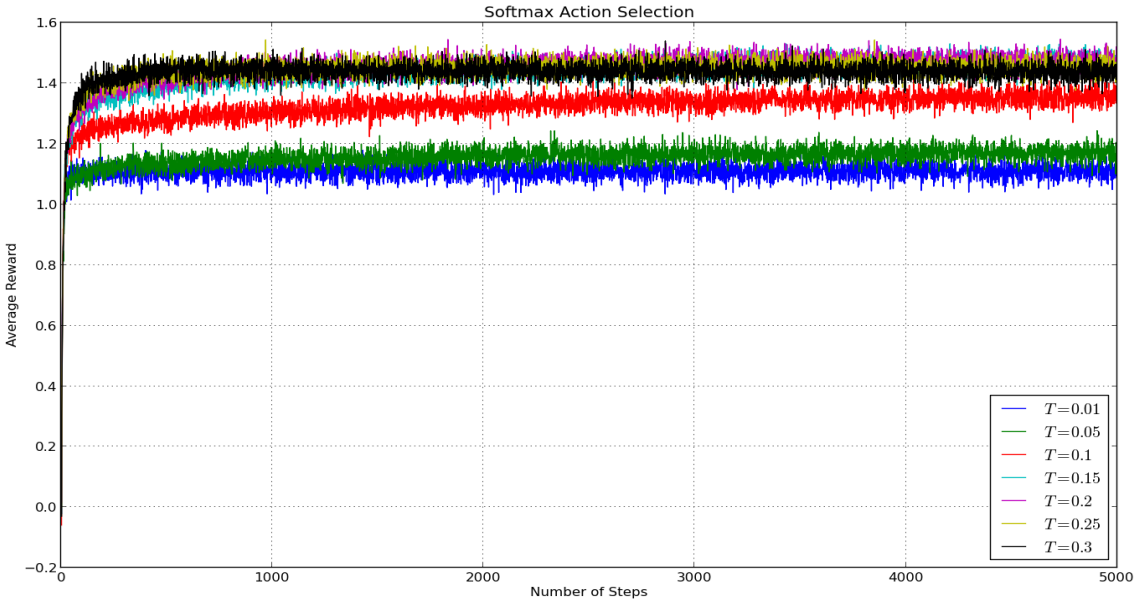
When the number of arms N increases, a change in behaviour of the algorithm can be noticed. It performs better for higher values of τ when T is kept constant. This is mainly because exploratory moves play a vital role in playing good arms. So low values of τ has high probability of not choosing the globally optimal (but sticks to local optimum) move when number of actions or arms is very high. This fact is strengthened by the fact high τ values result in playing the optimal action more frequently. Given below, the observed Graphs

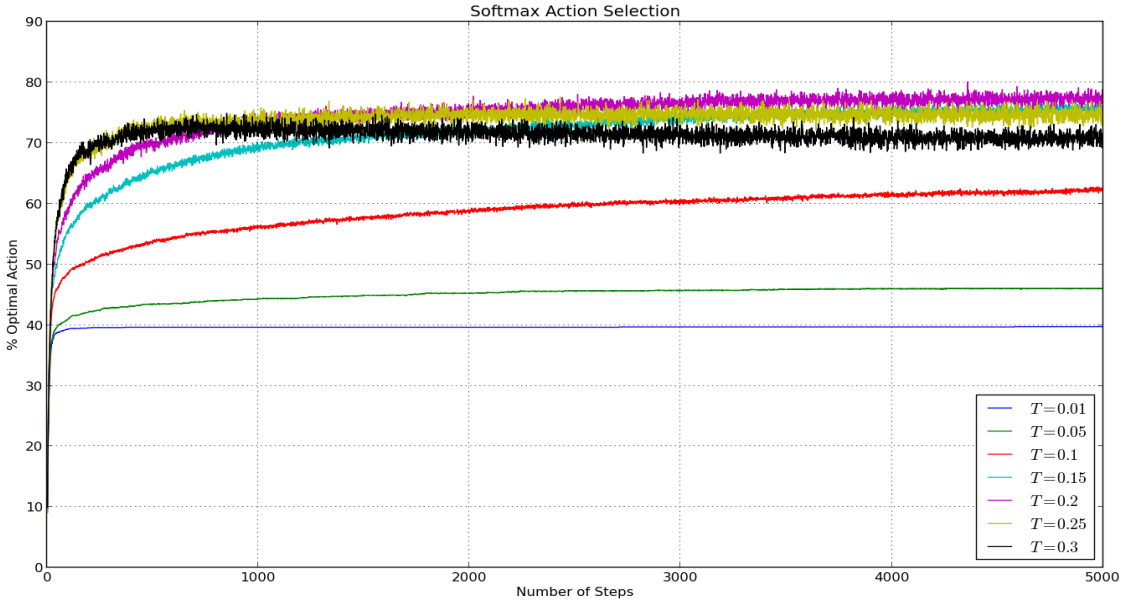
- T = 1000 and N = 10



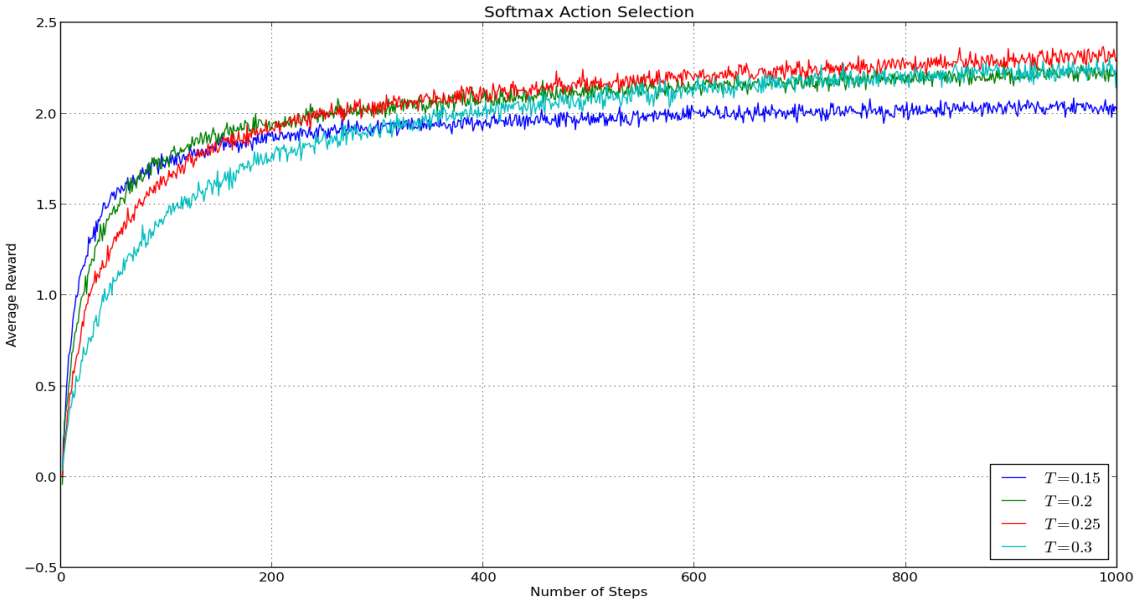


- T = 5000 and N = 10





- T = 1000 and N = 1000



1.3.3 Upper Confidence Bound

Since this algorithm does not a parameter to vary, it can be compared with other algorithms in-terms of average reward and % optimality action.

From the graphs it can be observed that with increase in the number of steps T, the average reward obtained at each step increases slightly. This is due to the fact that in the initial stages of the algorithm, it tries all the arms though they are very bad. This is given by the exploratory factor $\sqrt{\frac{2*\ln T}{t_j}}$ which reduces the net reward if an arm is played many times. The dynamics of the algorithm is such that whenever it plays a bad arm it learns and plays it less frequent than the other good arms played before.

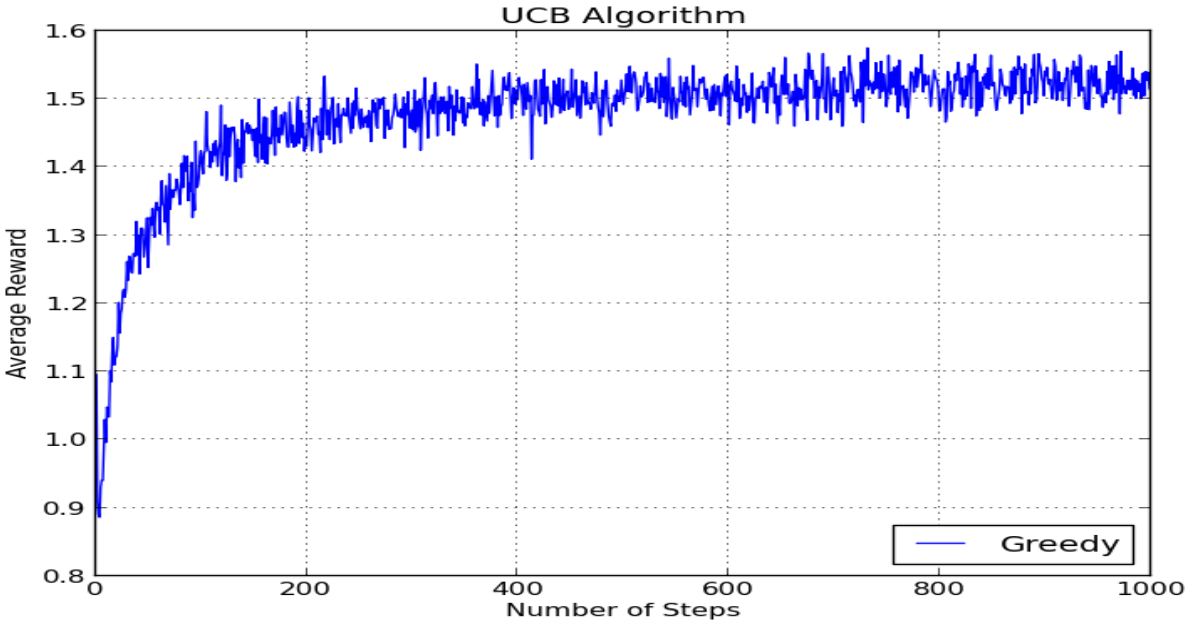
Also the number of time optimal action is being played increases with number of steps.

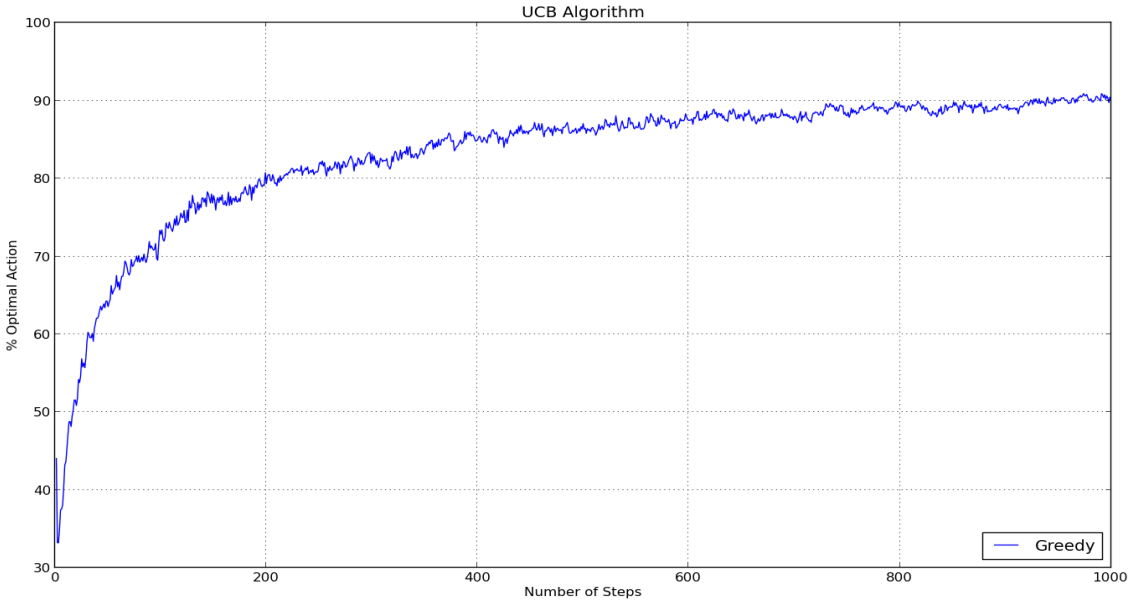
The performance for N = 1000 is drastically different when compared to N = 10. This is because of the same fact that the algorithm tries to play all the arms though they are bad. That is one of the important reason for bad average reward in the first few steps. This is also evident from the optimality plot.

The UCB algorithm performs better in case of N = 10. This is due to the fact that UCB algorithm tries to play all the arms. But the frequency with which an arm is played is directly related to the reward obtained from that arm. So, this achieves high average reward when number of steps increases. But when number of arms increases N = 1000, the algorithm plays many bad arms and hence it performs worse in the first few steps. These can be clearly observed from the graphs obtained.

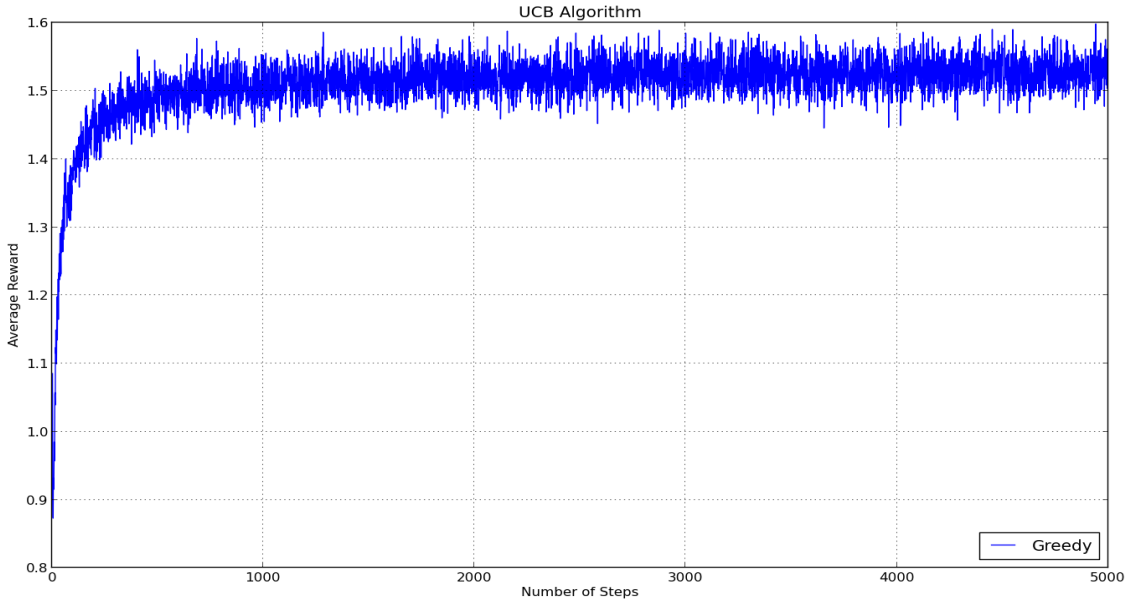
Given below, the observed Graphs

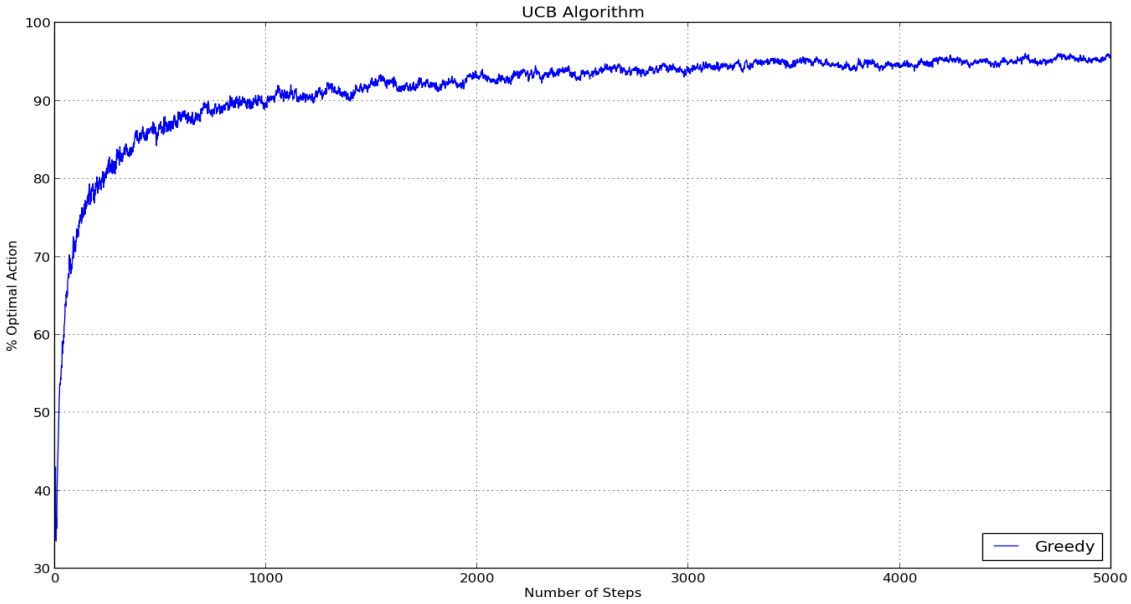
- T = 1000 and N = 10



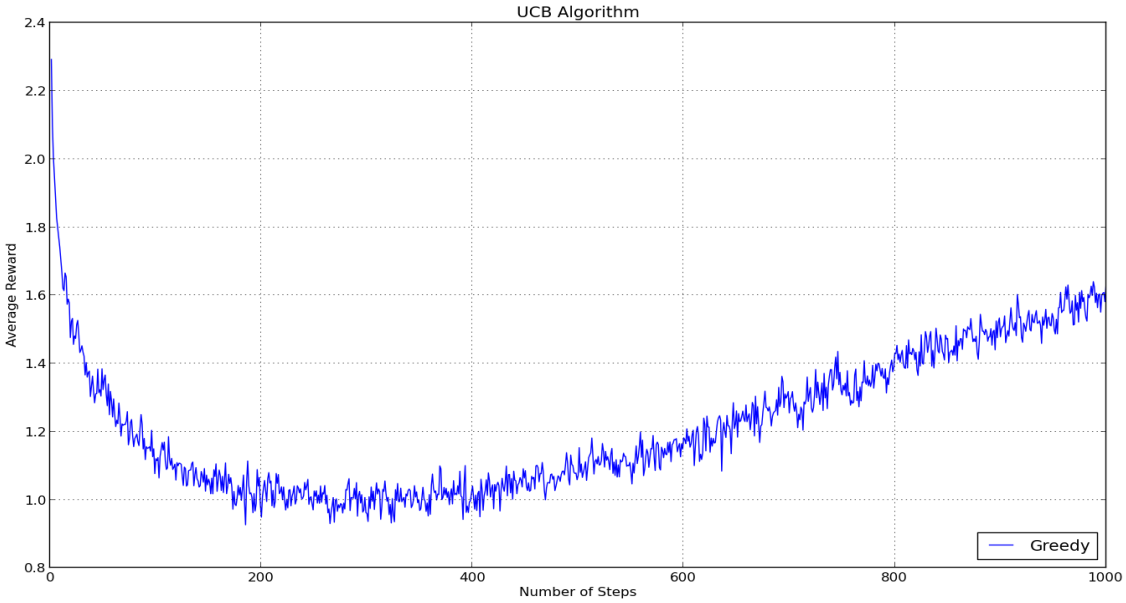


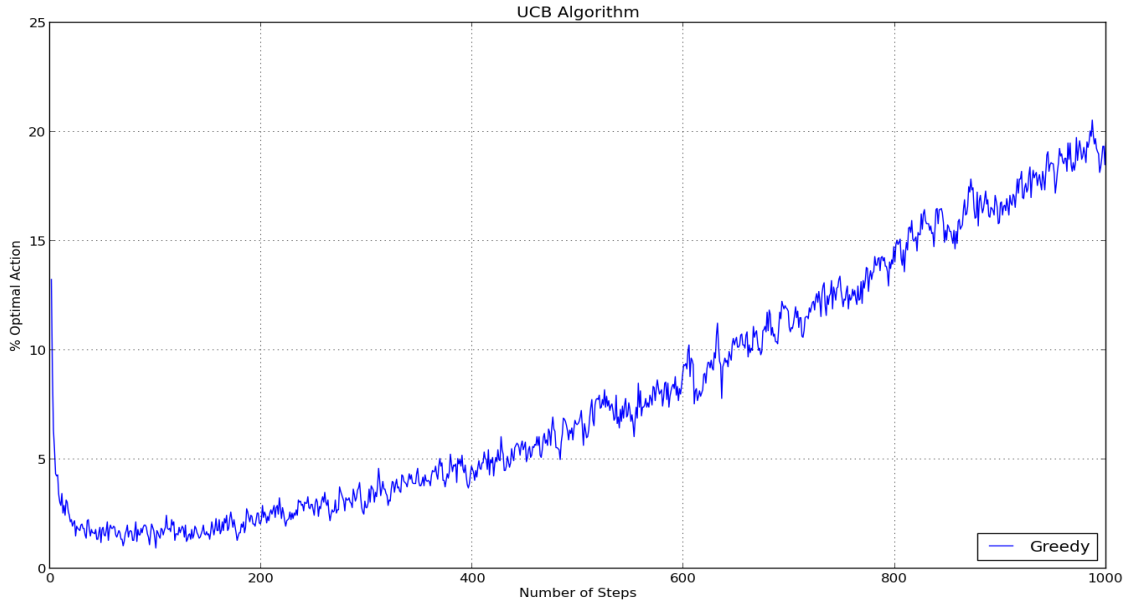
- T = 5000 and N = 10





- T = 1000 and N = 1000





1.4 Algorithm Comparison

From the graphs it is clear that

- T = 1000 and N = 10, UCB algorithm performs better than other two algorithms
- T = 5000 and N = 10, UCB performs better
- T = 1000 and N = 1000, Soft Max action selection performs better.

The above experiments are run for True-action values sampled from Standard normal distribution.

1.5 Conclusion

The above experiments essentially prove the following facts

- In the first few steps of any algorithm, greater the number of exploratory moves played, greater the average reward.
- After a large number of steps, lower the number of exploratory moves played, greater the average reward.
- When number of arms(or actions) increases, though exploratory moves perfroms very bad in the first few steps, it performs relatively well after large number of steps when compared to problems with less number of arms.