Aldar Saranov

Aldar.Saranov@ulb.ac.be

Université libre de Bruxelles

Assignment 3
Multi-Armed Bandits

Discipline: INFO-F409 Learning dynamics (M-INFOS/F277)

Teachers: Tom Lenaerts, Roxana Rădulescu

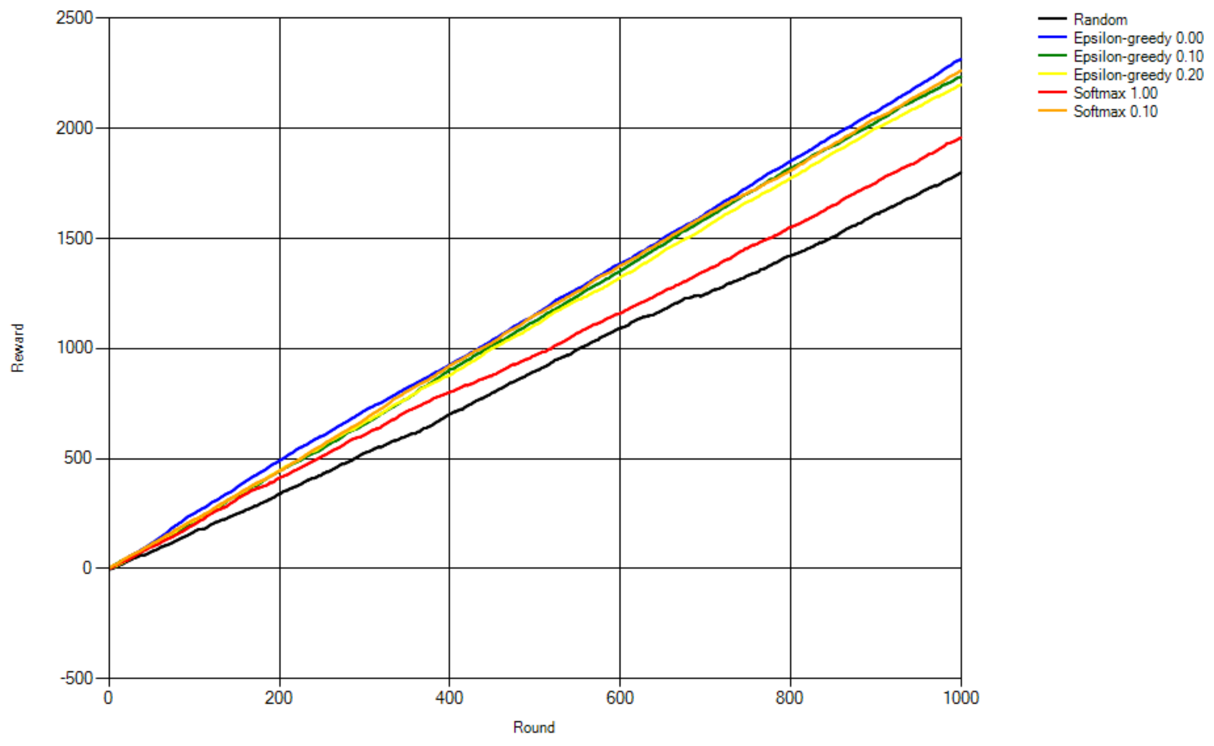
November 2016

N-Armed Bandit.

The specified problem was simulated using an application (code is located in the same archive). Application was developed using C# language for Windows platform.

Exercise 1

Reward per round

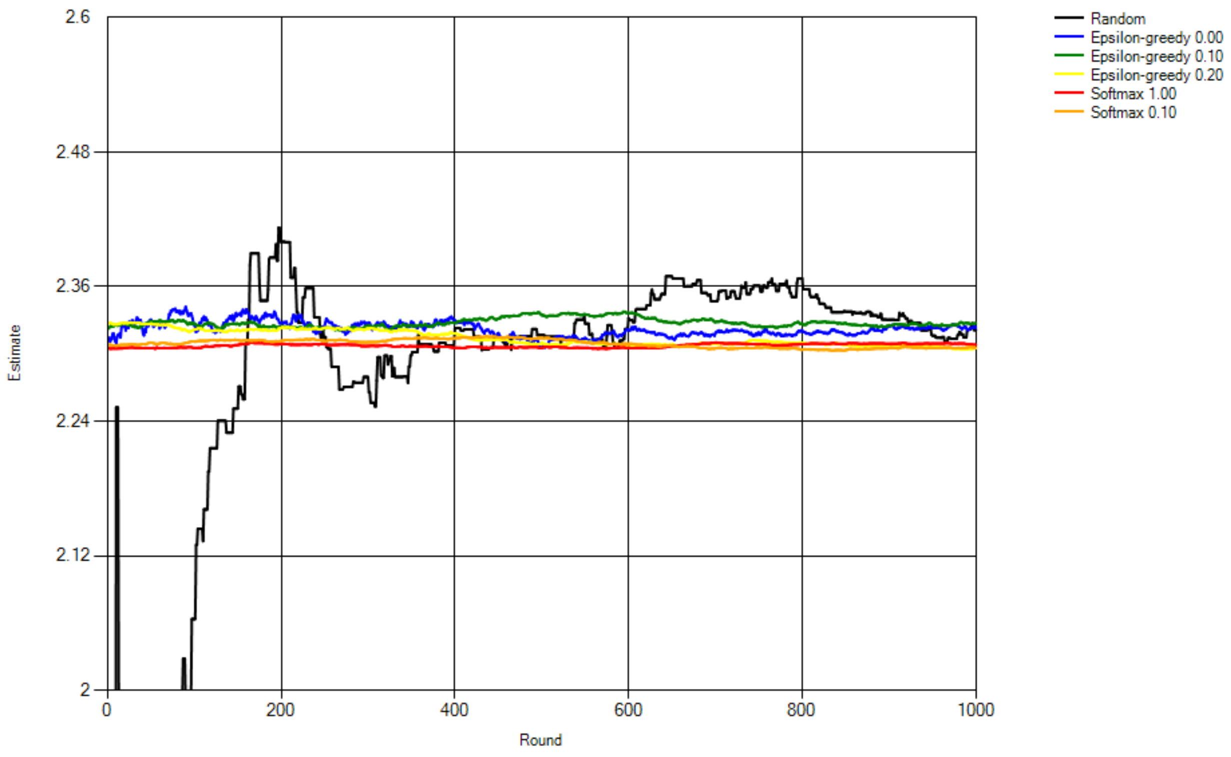


The algorithms achieve rewards in following descending order:

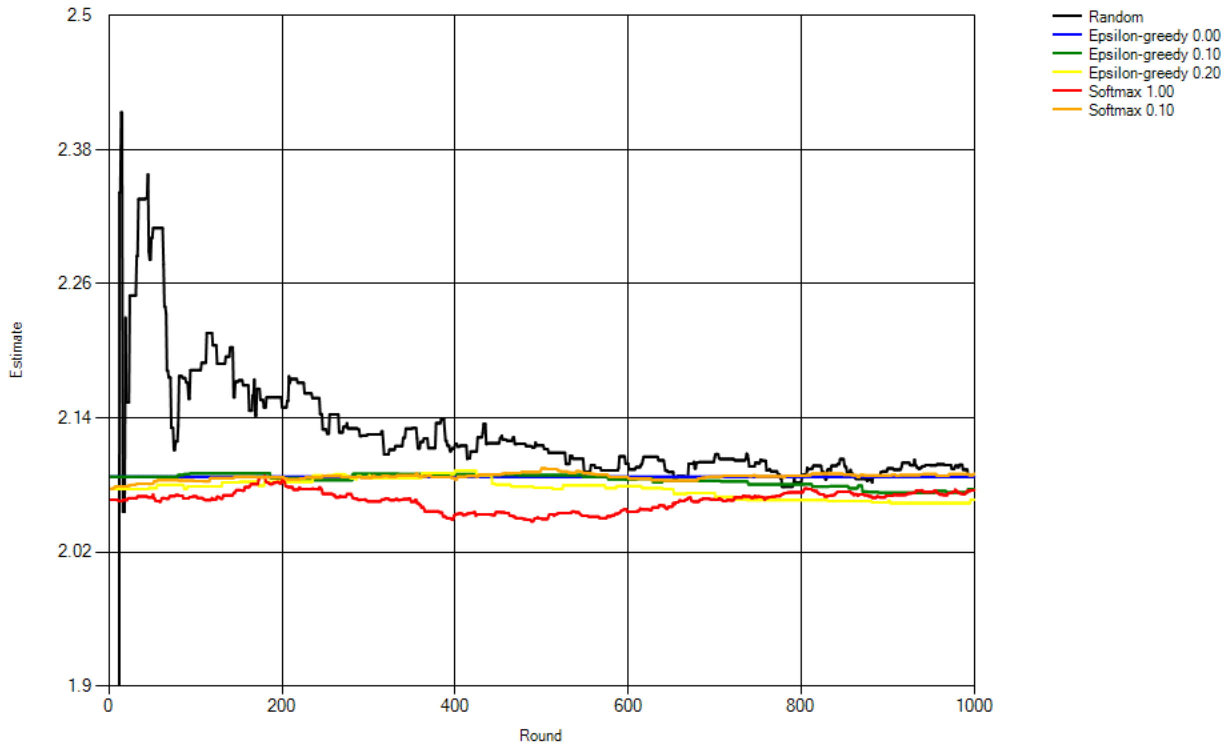
1. Epsilon greedy 0.0
2. Softmax 0.1
3. Epsilon-greedy 0.1
4. Epsilon-greedy 0.2
5. Softmax 1.0
6. Random

Random showed the worst result since it does not do much of exploiting. Epsilon greedy 0.0 showed the best result obviously because it stands for the best combination of exploration and exploiting.

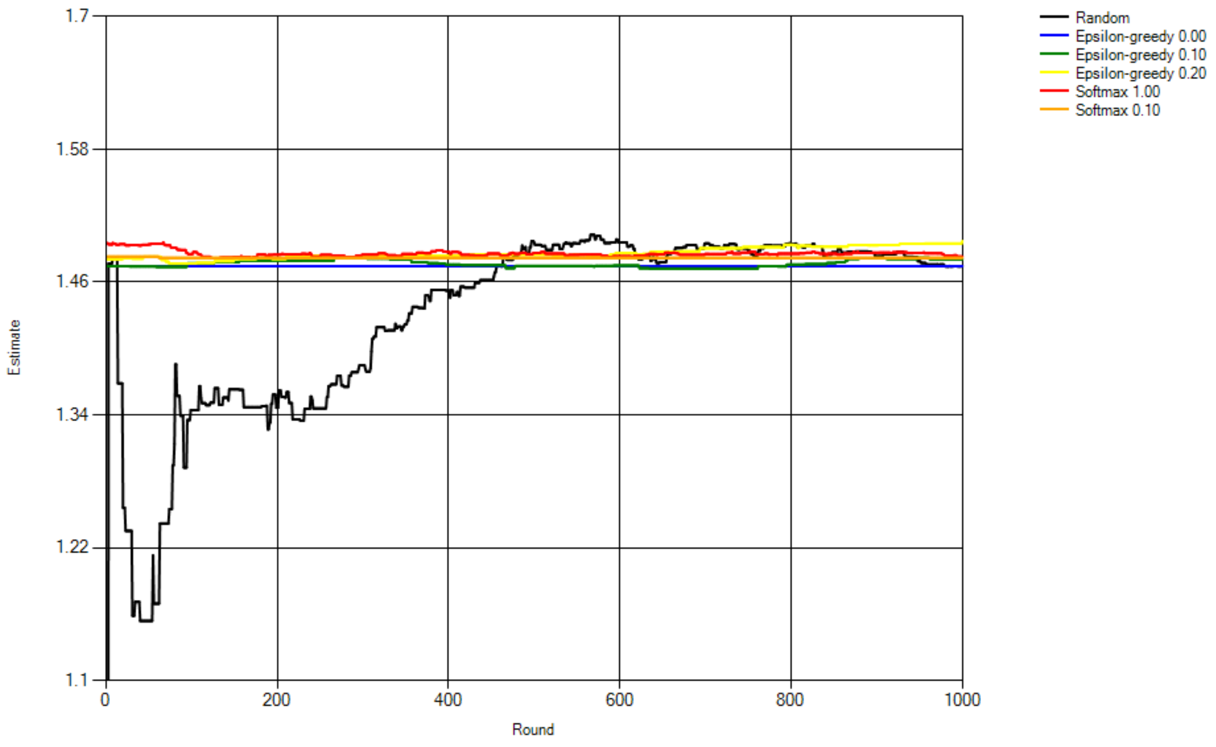
Arm 1


2.6
2.48
2.36
2.24
2.12
2
Estimate
0
200
400
600
800
1000
Round
Random
Epsilon-greedy 0.00
Epsilon-greedy 0.10
Epsilon-greedy 0.20
Softmax 1.00
Softmax 0.10

Arm 2

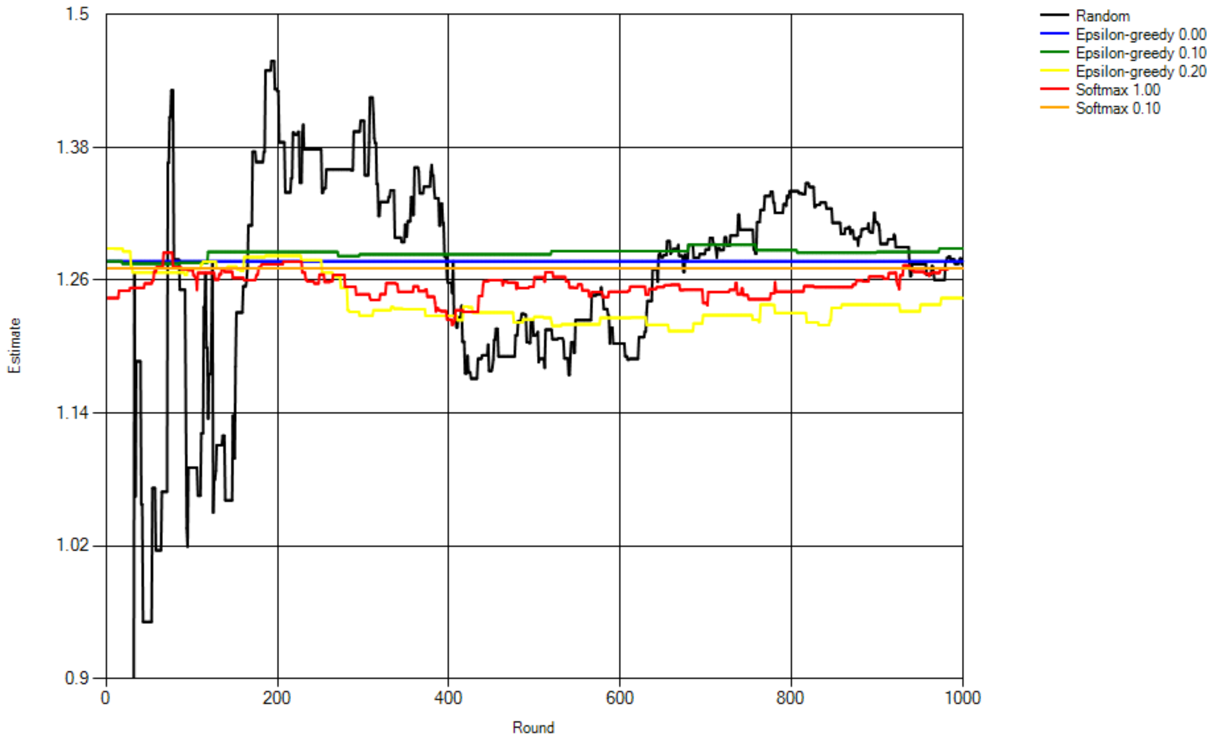

2.5
2.38
2.26
2.14
2.02
1.9
Estimate
0
200
400
600
800
1000
Round
Random
Epsilon-greedy 0.00
Epsilon-greedy 0.10
Epsilon-greedy 0.20
Softmax 1.00
Softmax 0.10

Arm 3



1.7
1.58
1.46
1.34
1.22
1.1
0
200
400
600
800
1000
Round
Estimate
Random
Epsilon-greedy 0.00
Epsilon-greedy 0.10
Epsilon-greedy 0.20
Softmax 1.00
Softmax 0.10

Arm 4



1.5
1.38
1.26
1.14
1.02
0.9
0
200
400
600
800
1000
Round
Estimate
Random
Epsilon-greedy 0.00
Epsilon-greedy 0.10
Epsilon-greedy 0.20
Softmax 1.00
Softmax 0.10

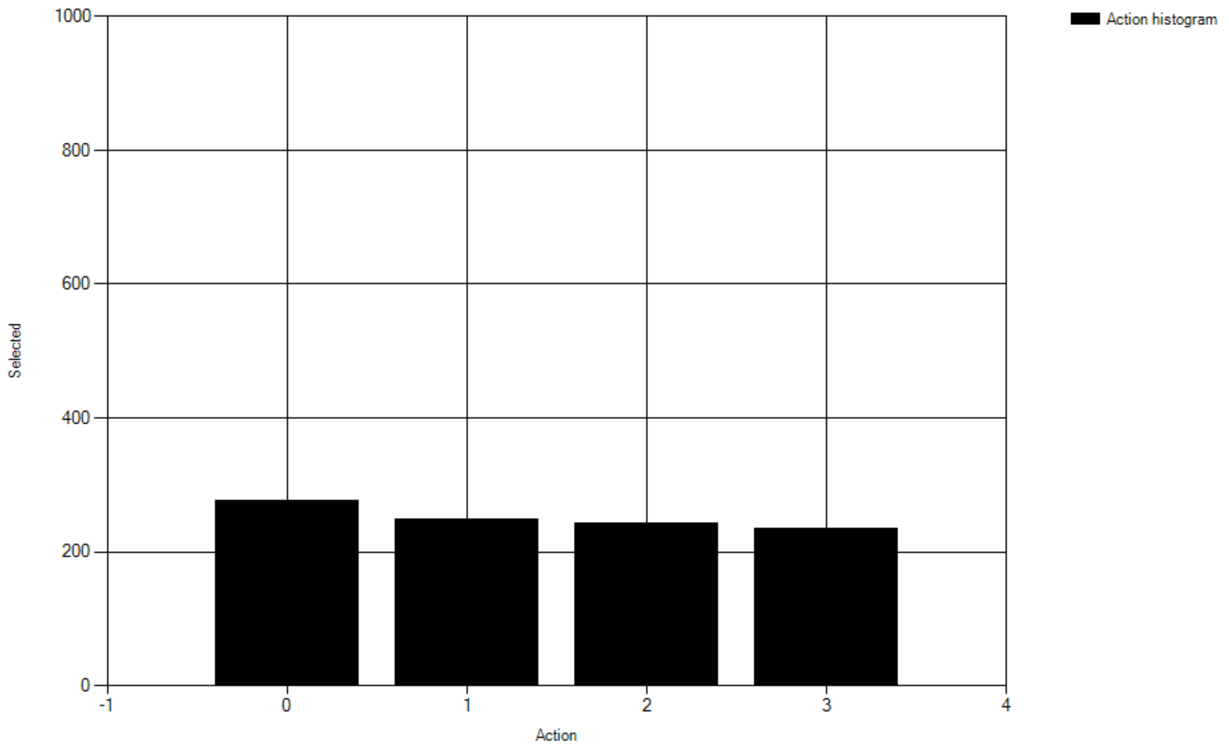
Based on research of these 4 arms estimations we can establish a table of ranks of the action selection algorithms.

Algorithm rank\Arm number	1	2	3	4
1	Softmax 0.1	Softmax 0.1	Epsilon-greedy 0.2	Epsilon-greedy 0.1
2	Epsilon-greedy 0.2	Epsilon-greedy 0.0	Softmax 1.0	Epsilon-greedy 0.0
3	Softmax 1.0	Random	Softmax 0.1	Random
4	Random	Epsilon-greedy 0.1	Epsilon-greedy 0.1	Softmax 0.1
5	Epsilon-greedy 0.0	Softmax 1.0	Epsilon-greedy 0.0	Softmax 1.0
6	Epsilon-greedy 0.1	Epsilon-greedy 0.2	Random	Epsilon-greedy 0.2

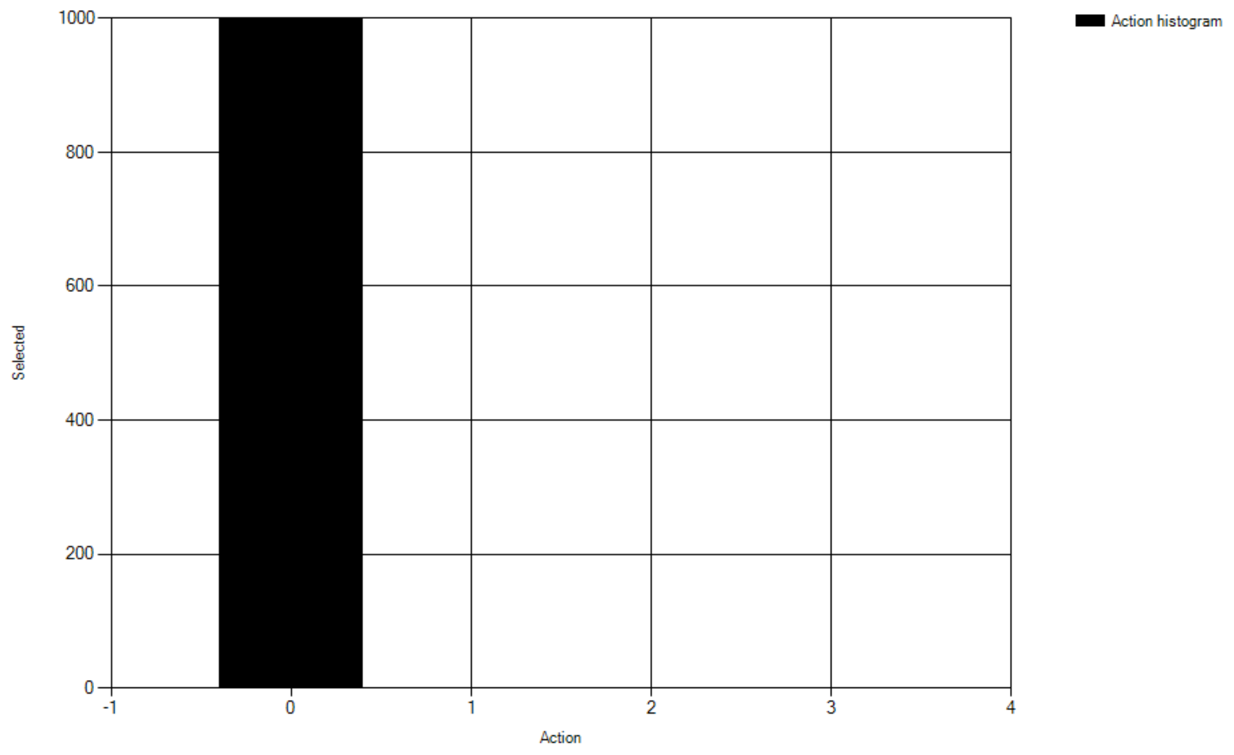
From this table we can infer that in average (in a general case) softmax 0.1 proved to be a more precise action selection algorithm.

“Softmax 0.1” and “Epsilon-greedy 0.2” tend to achieve a stable estimation faster than the others.

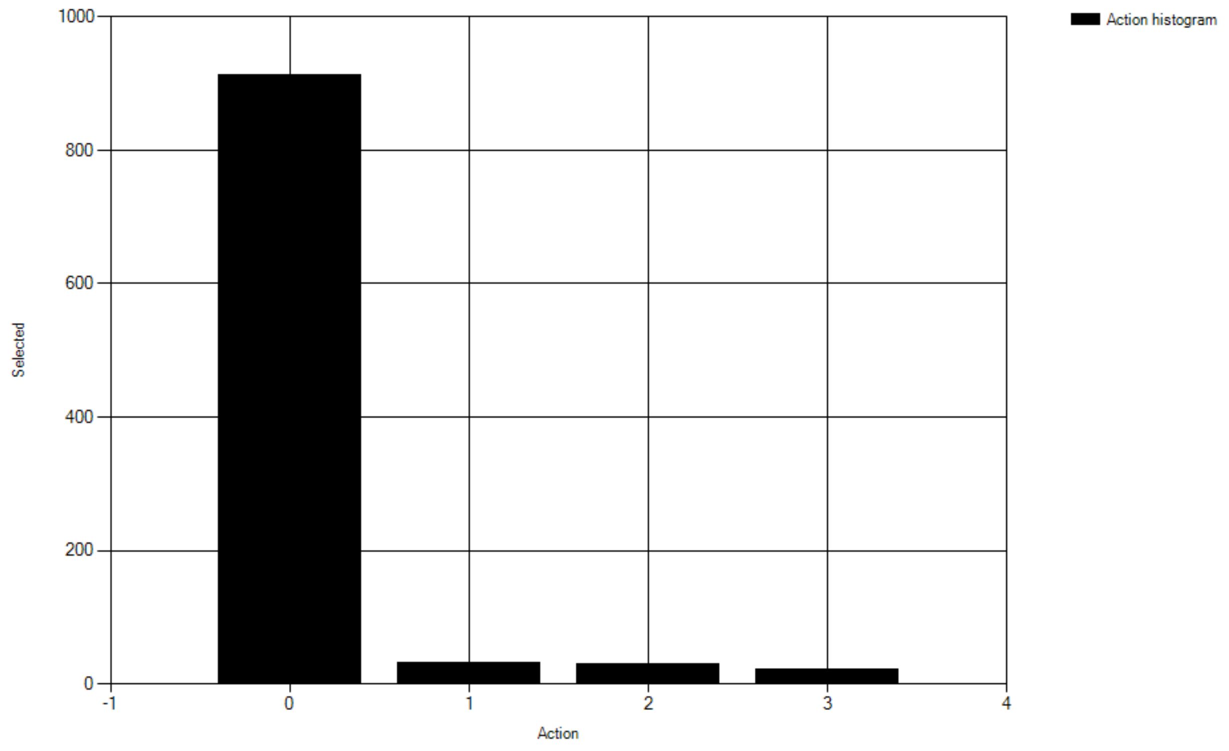
Random



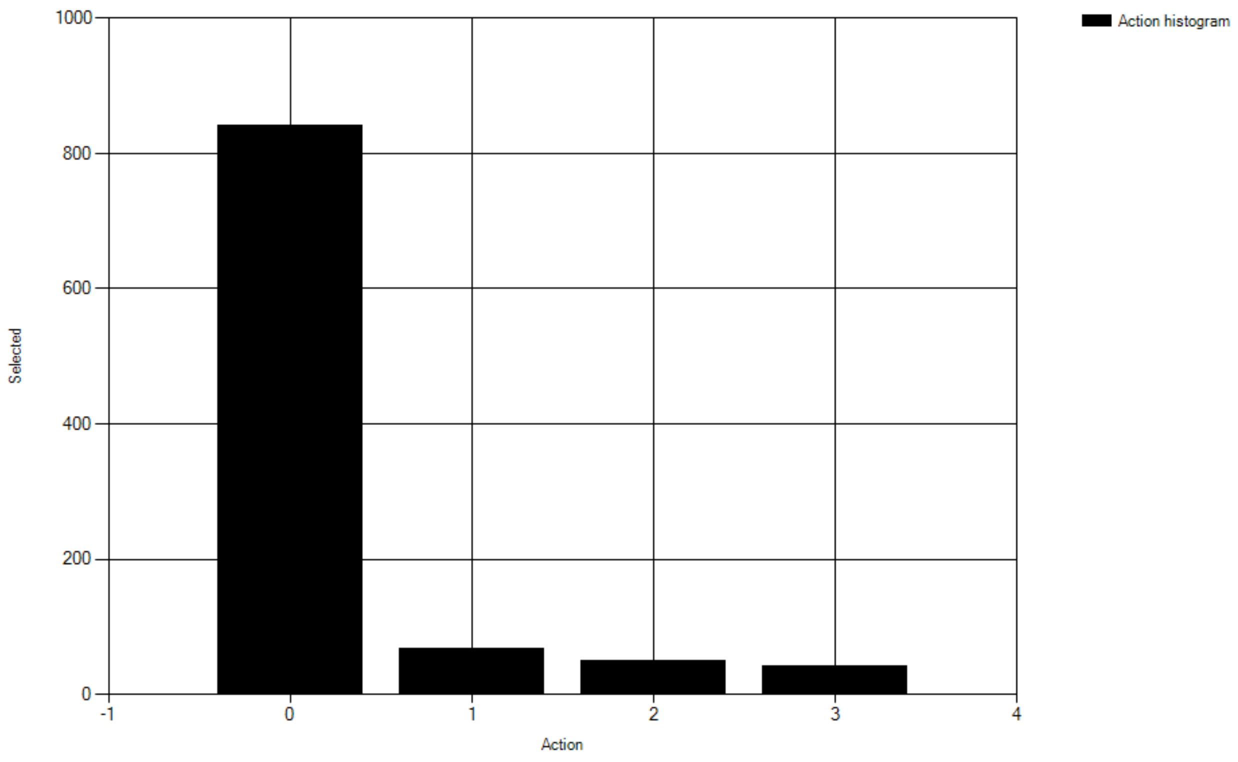
Greedy 0.0



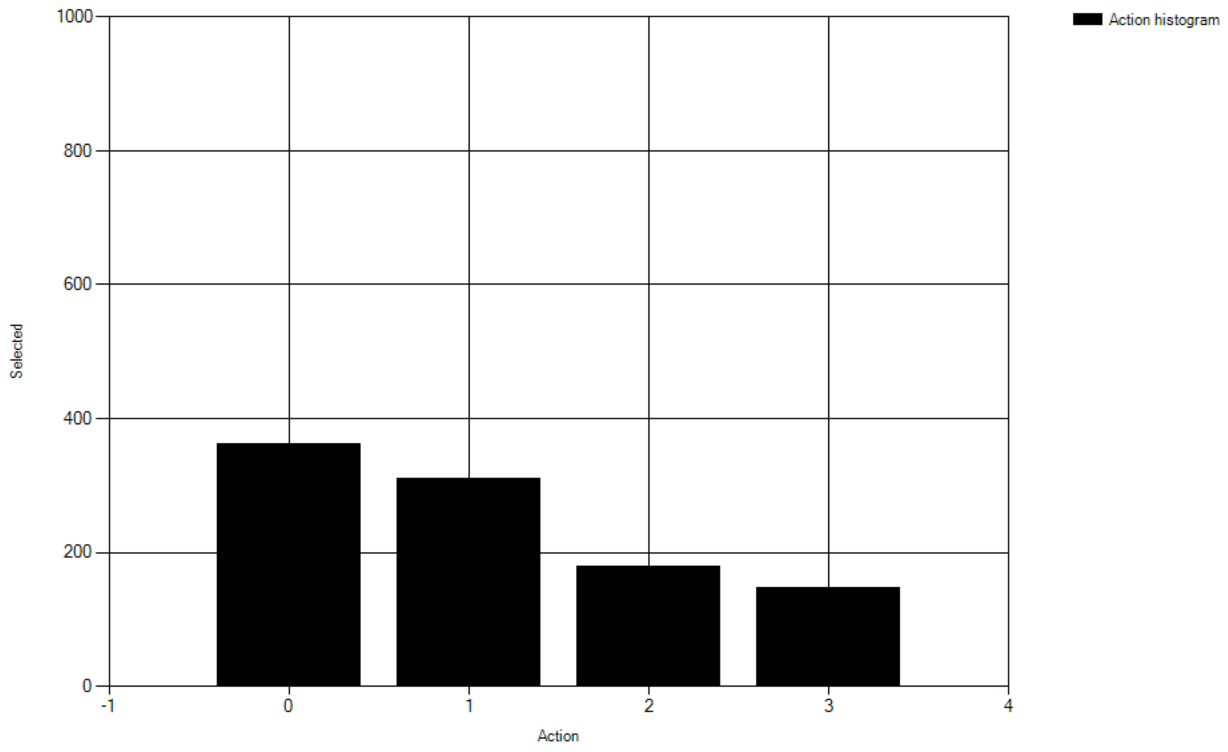
Greedy 0.1



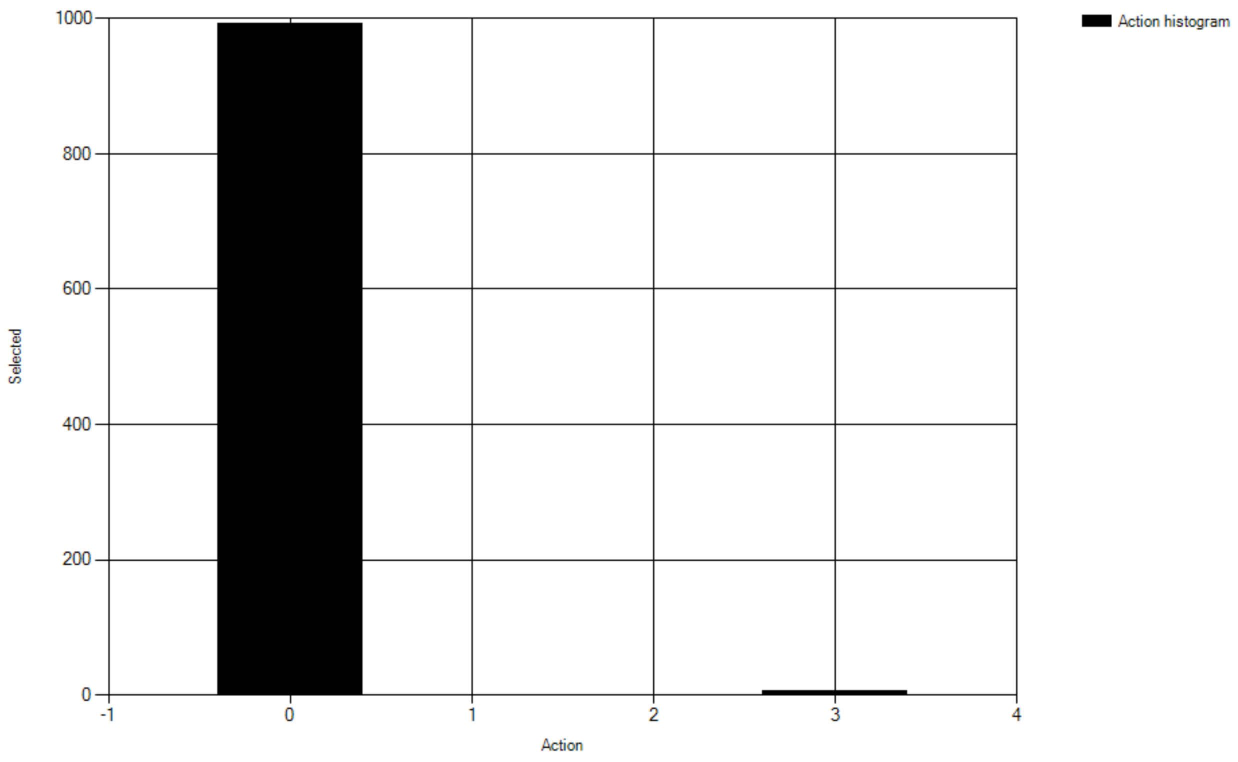
Greedy 0.2



Softmax 1.0



Softmax 0.1

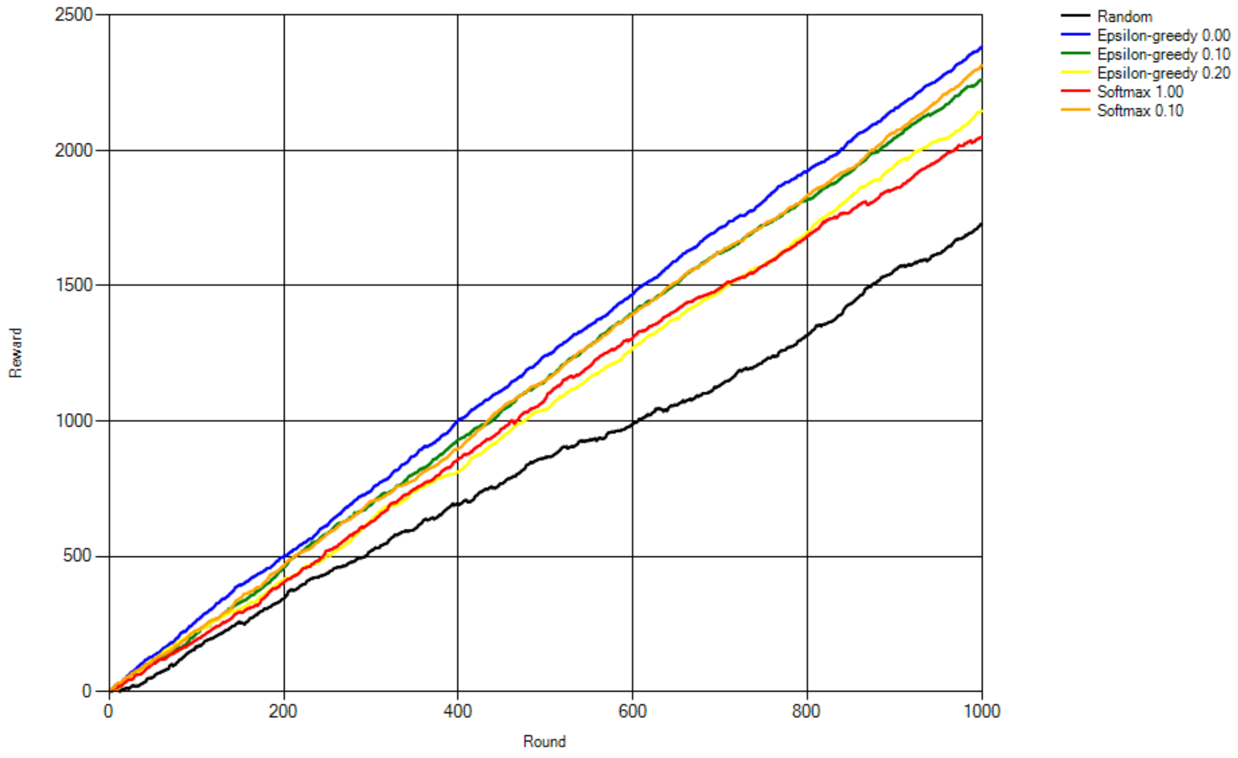


In the action histograms we can see:

1. Random does not tend to exploit.
2. The more is coefficient of Greedy, the more it is tending to explore.
3. Softmax 1.0 has minor tend to exploit and major to explore.
4. Softmax 0.1 has major tend to exploit and minor to explore.

Exercise 2

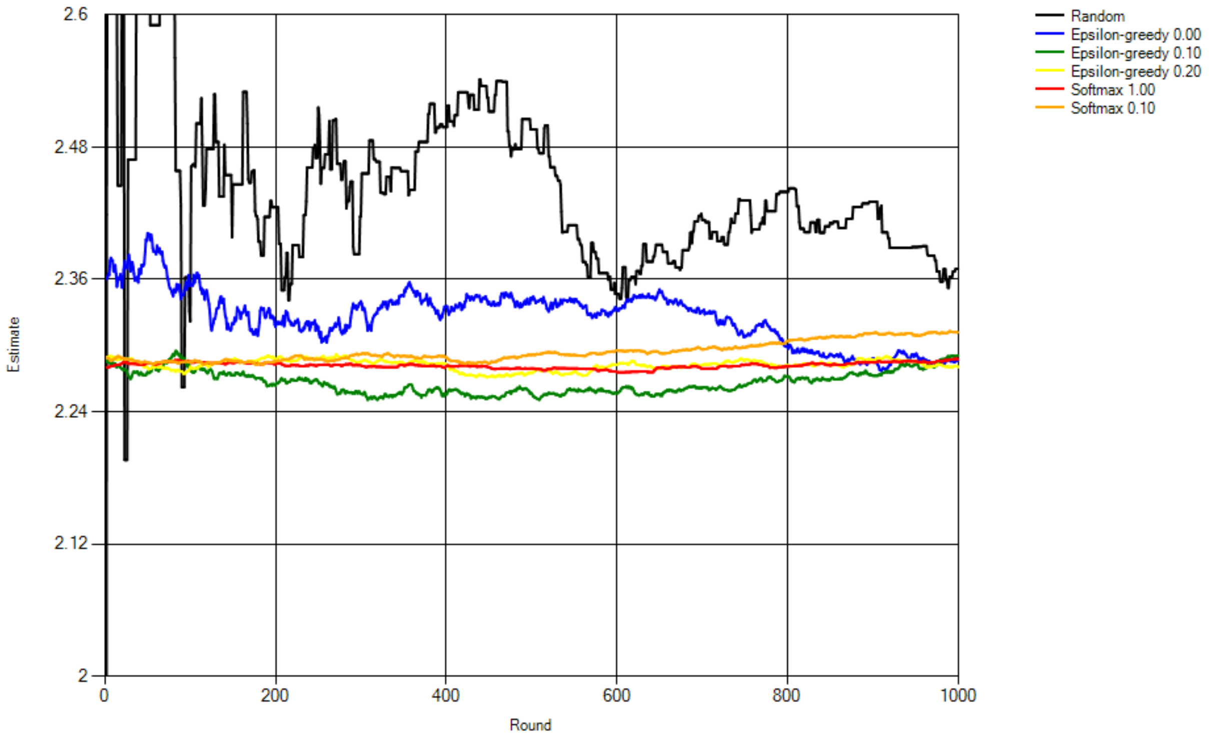
Reward per round



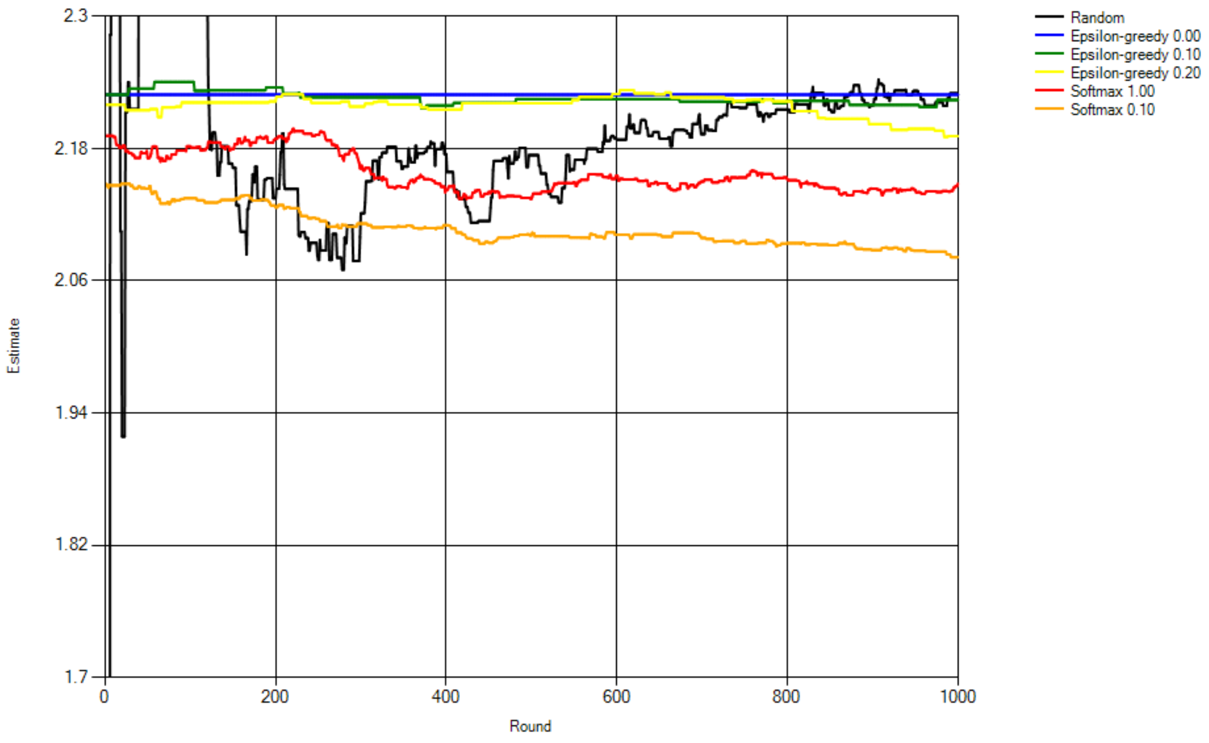
The algorithms achieve rewards in following same order as in previous exercise.

This means that deviation does not affect the performance of selection algorithm in respect to the other algorithms.

Arm 1

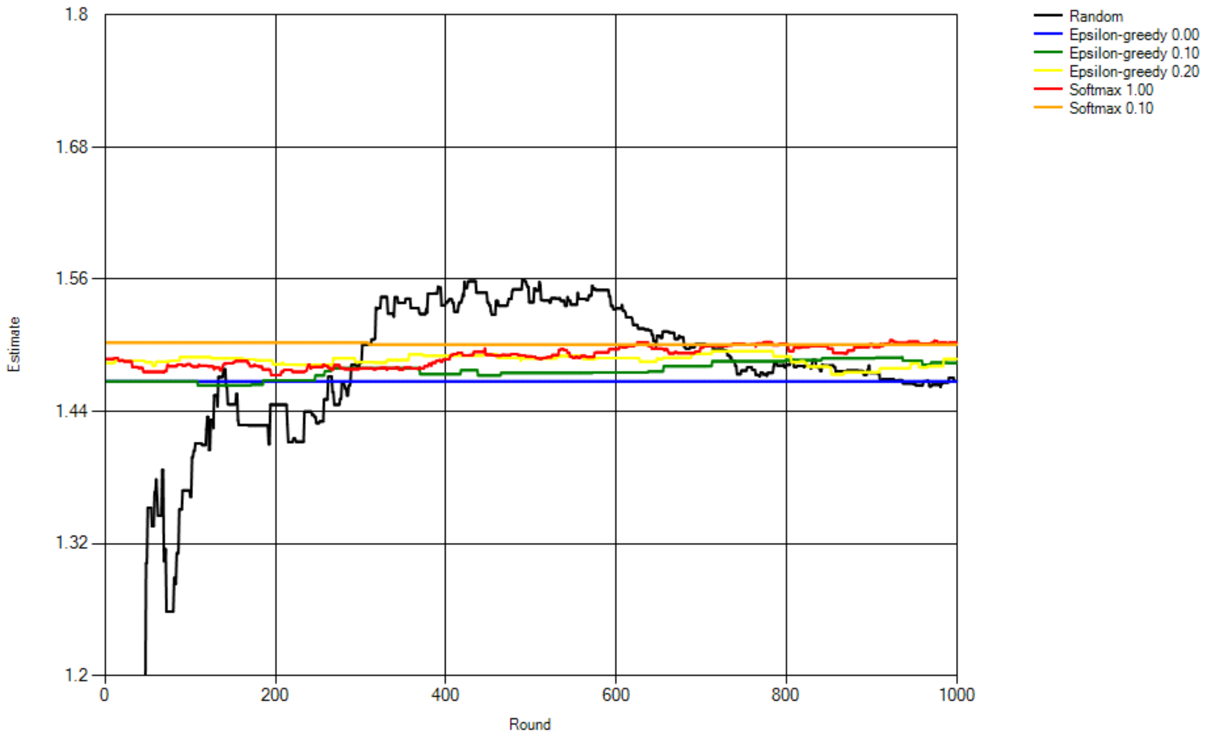


Arm 2



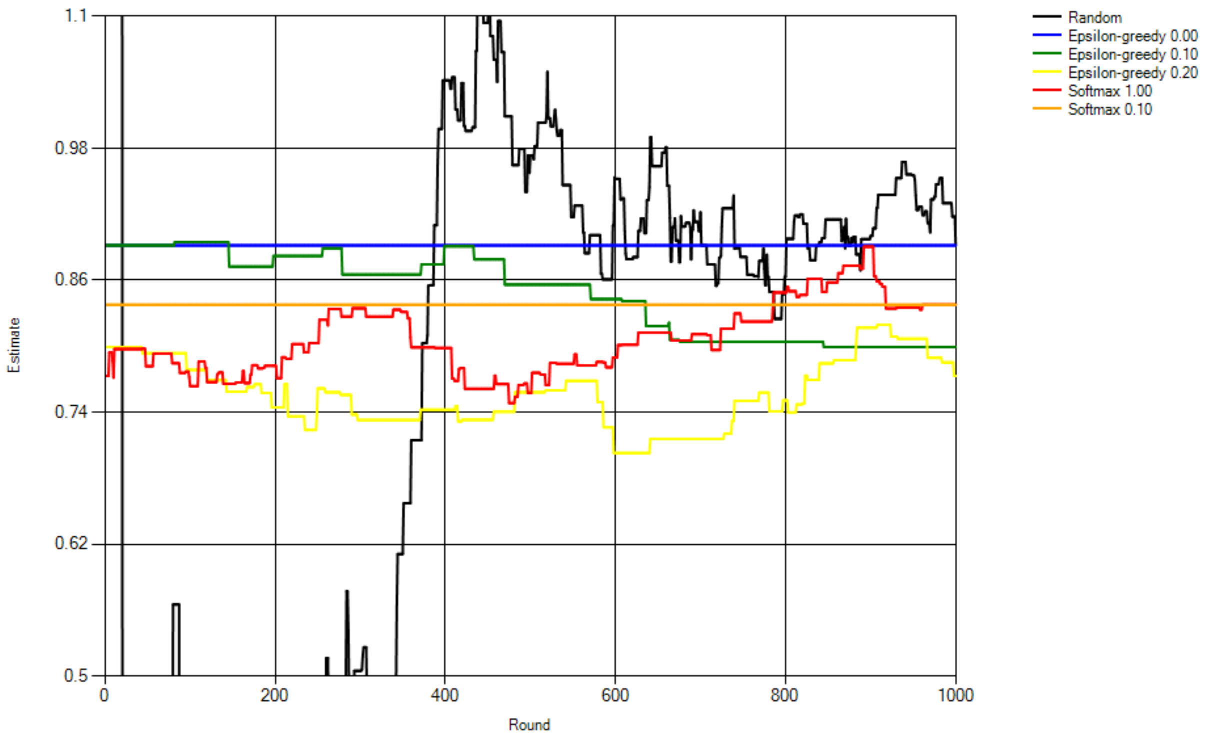
2.3
2.18
2.06
1.94
1.82
1.7
Estimate
0
200
400
600
800
1000
Round
Random
Epsilon-greedy 0.00
Epsilon-greedy 0.10
Epsilon-greedy 0.20
Softmax 1.00
Softmax 0.10

Arm 3



1.8
1.68
1.56
1.44
1.32
1.2
Estimate
0
200
400
600
800
1000
Round
Random
Epsilon-greedy 0.00
Epsilon-greedy 0.10
Epsilon-greedy 0.20
Softmax 1.00
Softmax 0.10

Arm 4


1.1
0.98
0.86
0.74
0.62
0.5
Estimate
0
200
400
600
800
1000
Round
Random
Epsilon-greedy 0.00
Epsilon-greedy 0.10
Epsilon-greedy 0.20
Softmax 1.00
Softmax 0.10

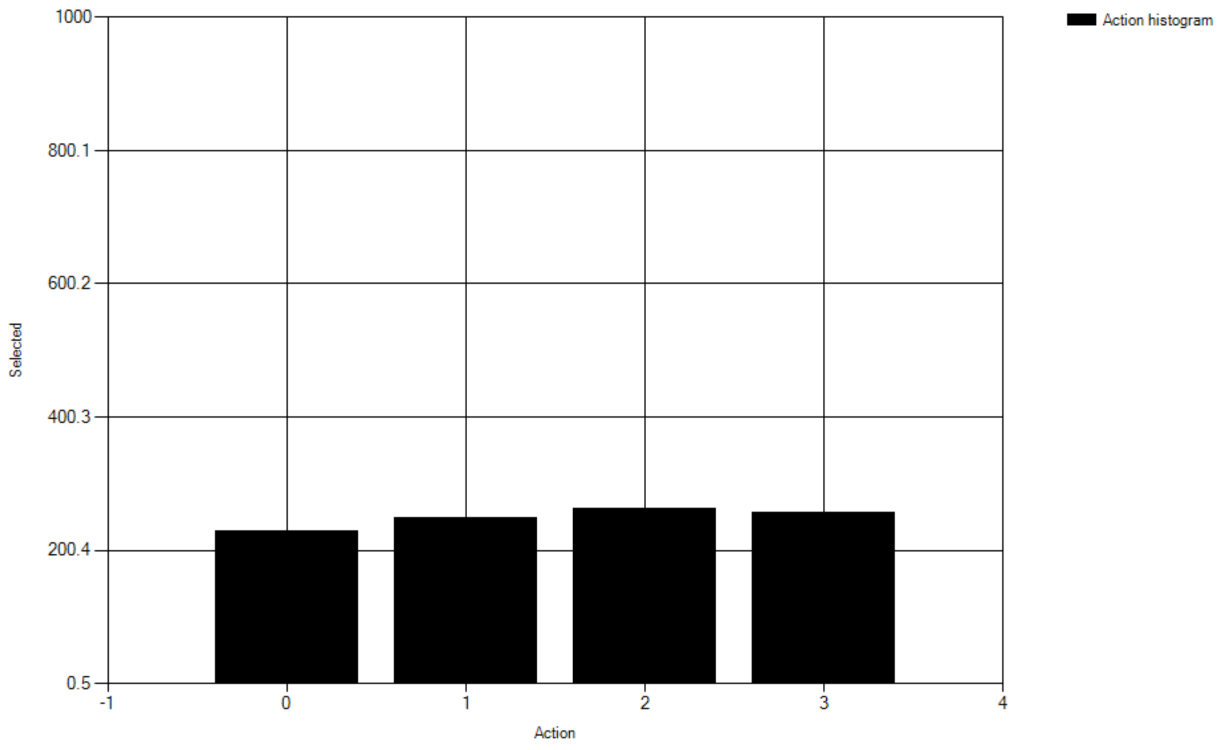
Based on research of these 4 arms estimations we can establish a table of ranks of the action selection algorithms.

Algorithm rank\Arm number	1	2	3	4
1	Epsilon-greedy 0.1	Softmax 0.1	Softmax 0.1	Random
2	Softmax 0.1	Softmax 1.0	Softmax 1.0	Epsilon-greedy 0.0
3	Softmax 1.0	Epsilon-greedy 0.2	Epsilon-greedy 0.2	Softmax 1.0
4	Epsilon-greedy 0.0	Epsilon-greedy 0.1	Epsilon-greedy 0.1	Softmax 0.1
5	Epsilon-greedy 0.2	Epsilon-greedy 0.0	Random	Epsilon-greedy 0.1
6	Random	Random	Epsilon-greedy 0.0	Epsilon-greedy 0.2

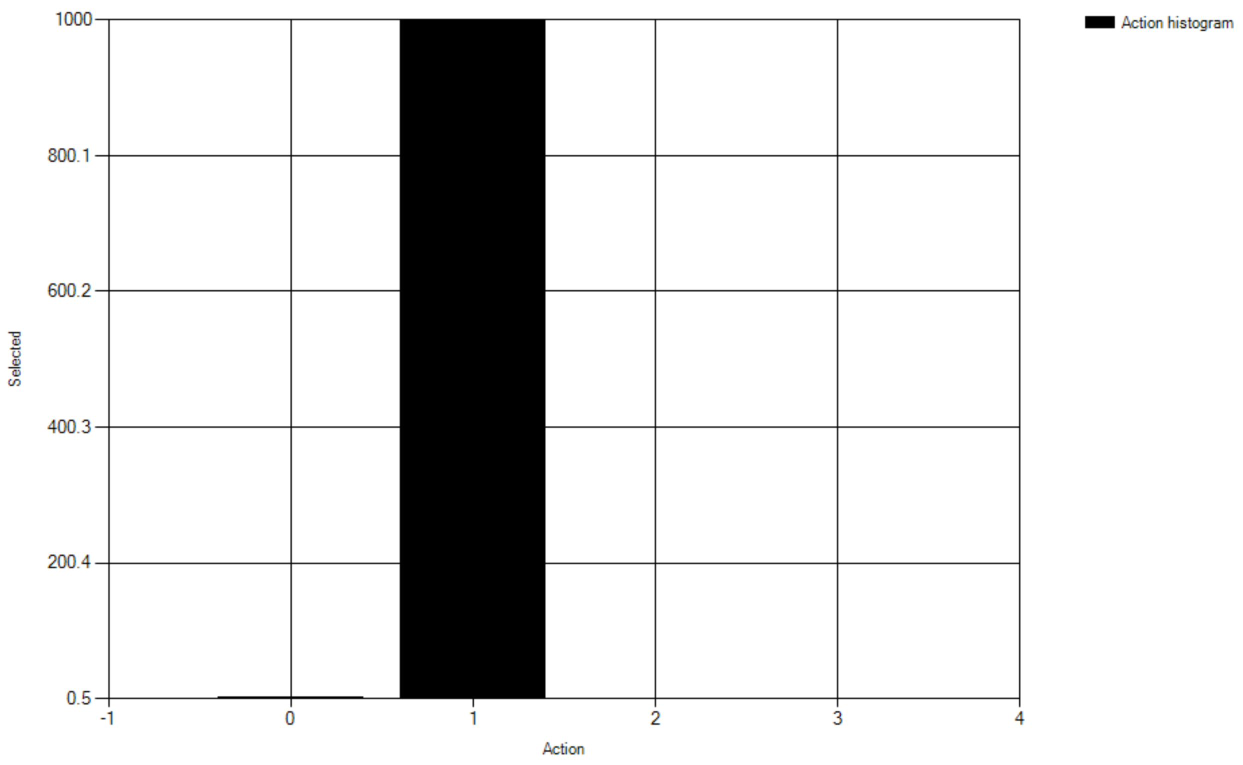
From this table we can infer that in average (in a general case) softmax 0.1 proved to be a more precise action selection algorithm once again.

“Softmax 0.1” and “Softmax 1.0” tend to achieve a stable estimation faster than the others.

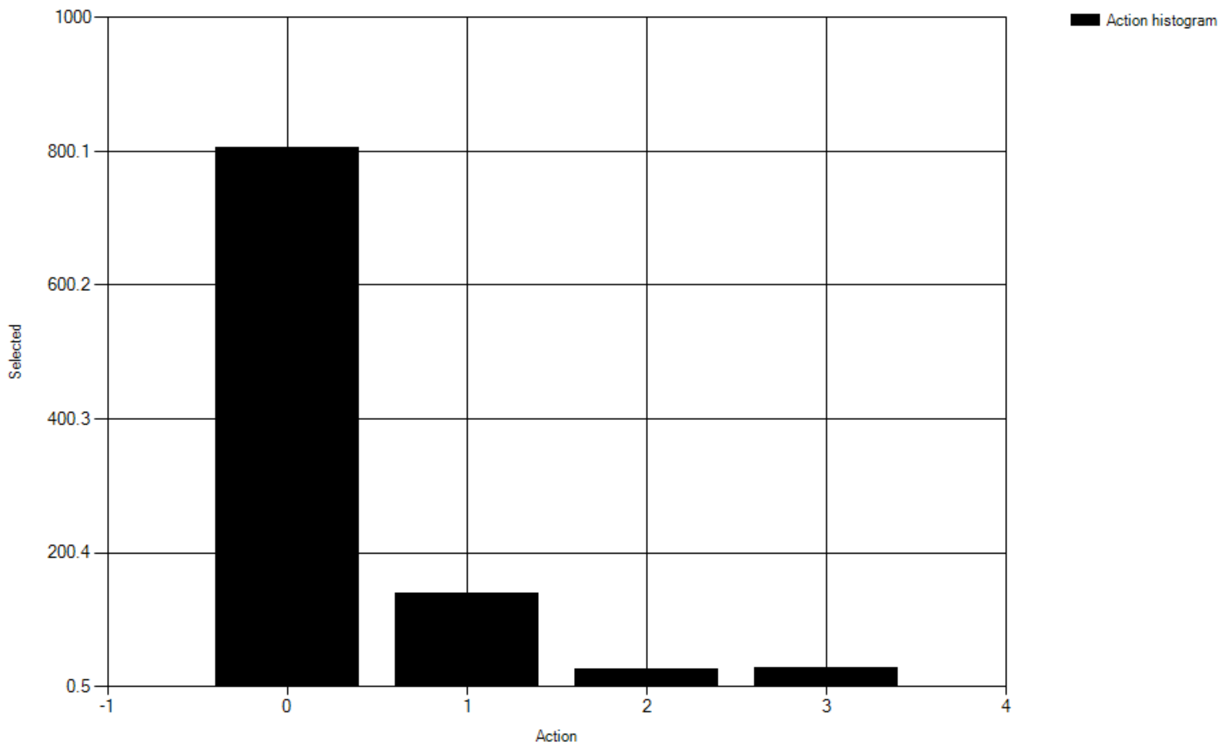
Random



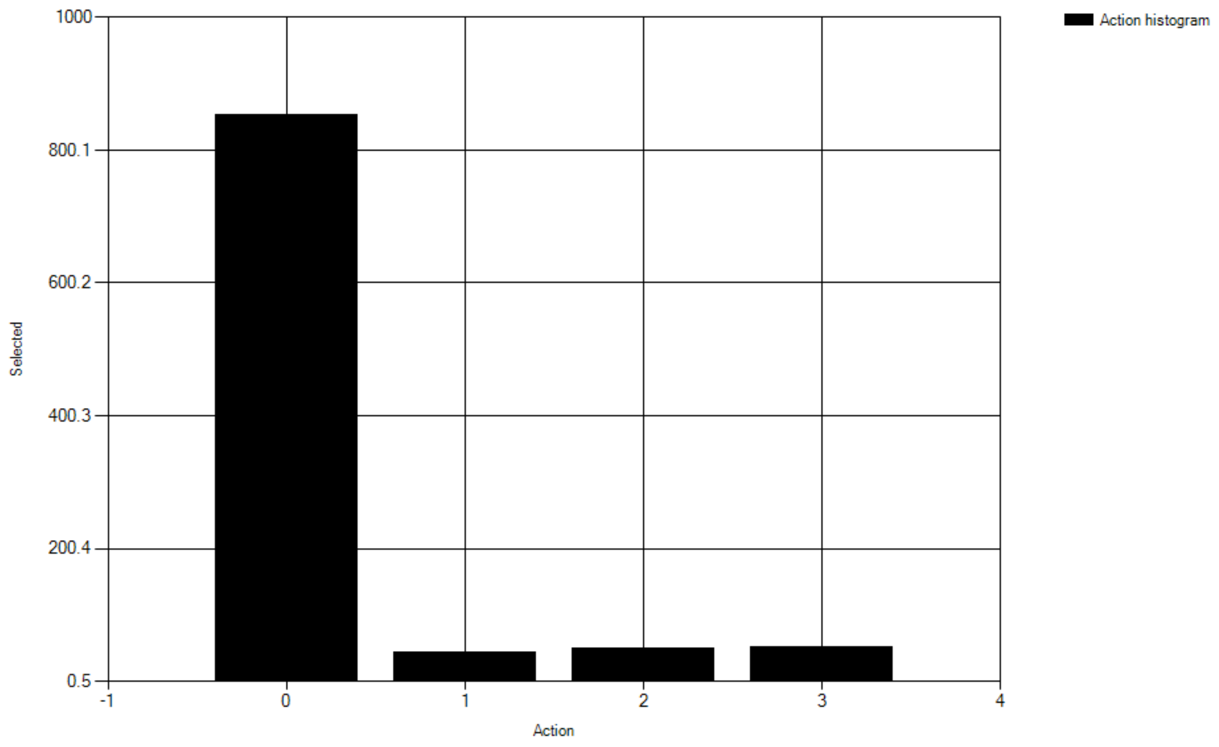
Greedy 0.0



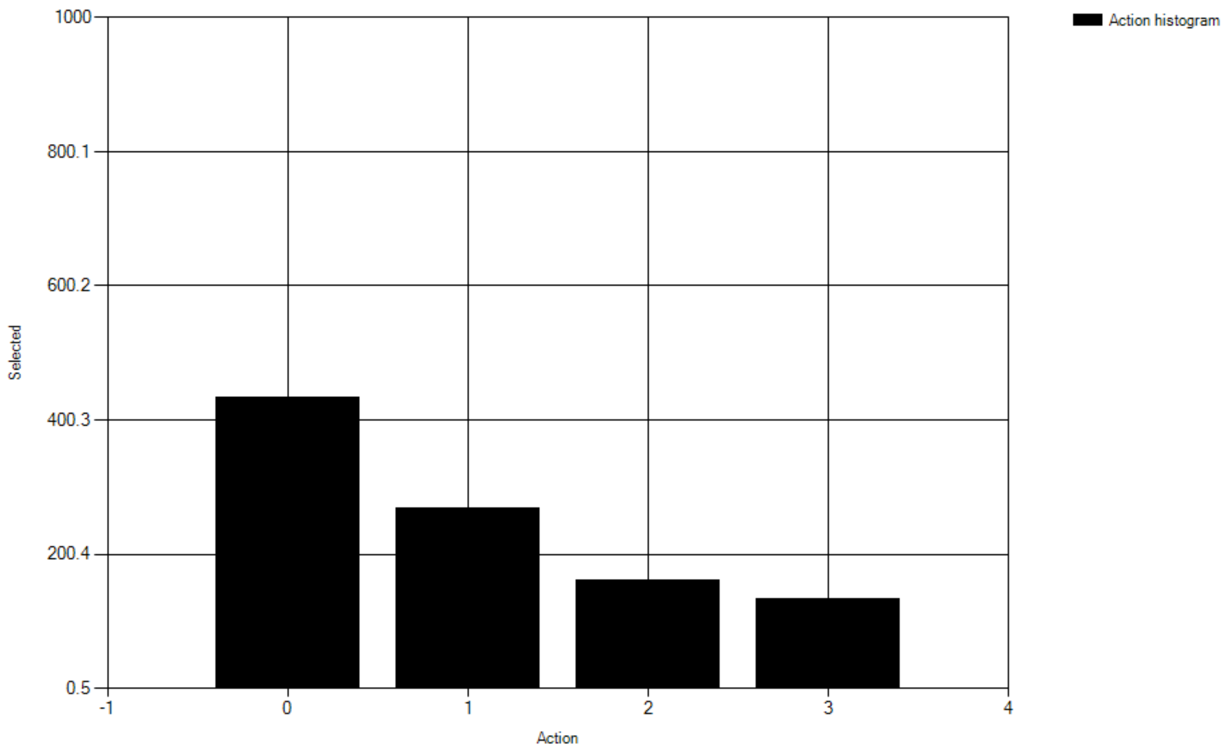
Greedy 0.1



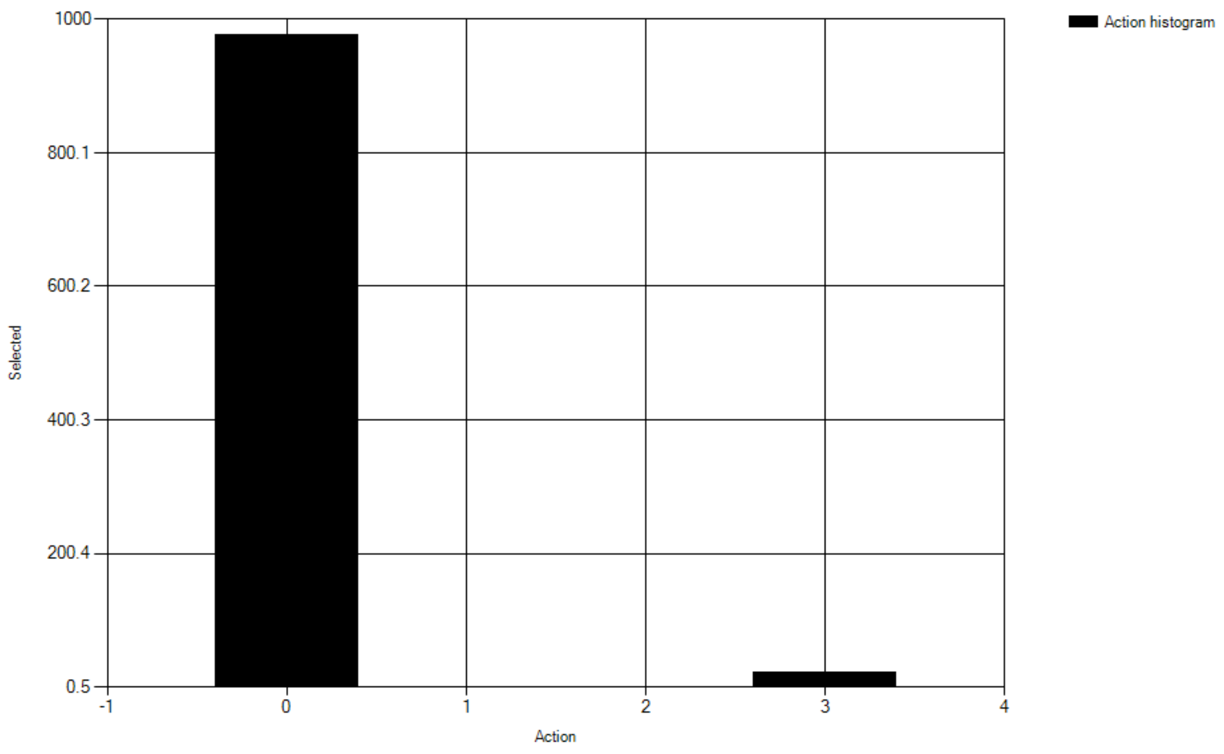
Greedy 0.2



Softmax 1.0



Softmax 0.1



The conclusions concerning the distribution of action selection histograms remain the same.

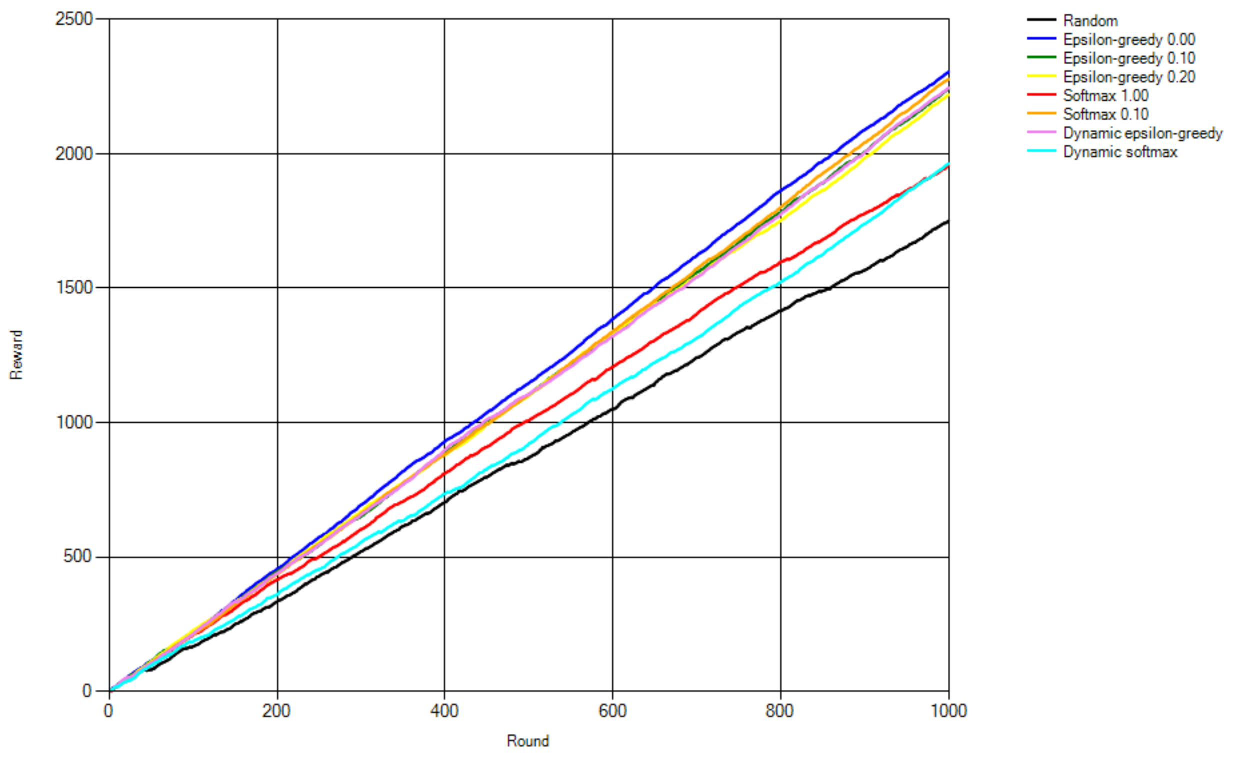
In comparison to exercise 1 we can state:

1. The precision of learning has fallen.
2. The speed of learning has fallen (it takes more time to arrive at steady state).
3. Especially it is obvious if we analyze arm №4 estimation. It is related to the fact that for that arm deviation is much larger than mean.

Exercise 3

Two additional dynamic selection algorithms were added.

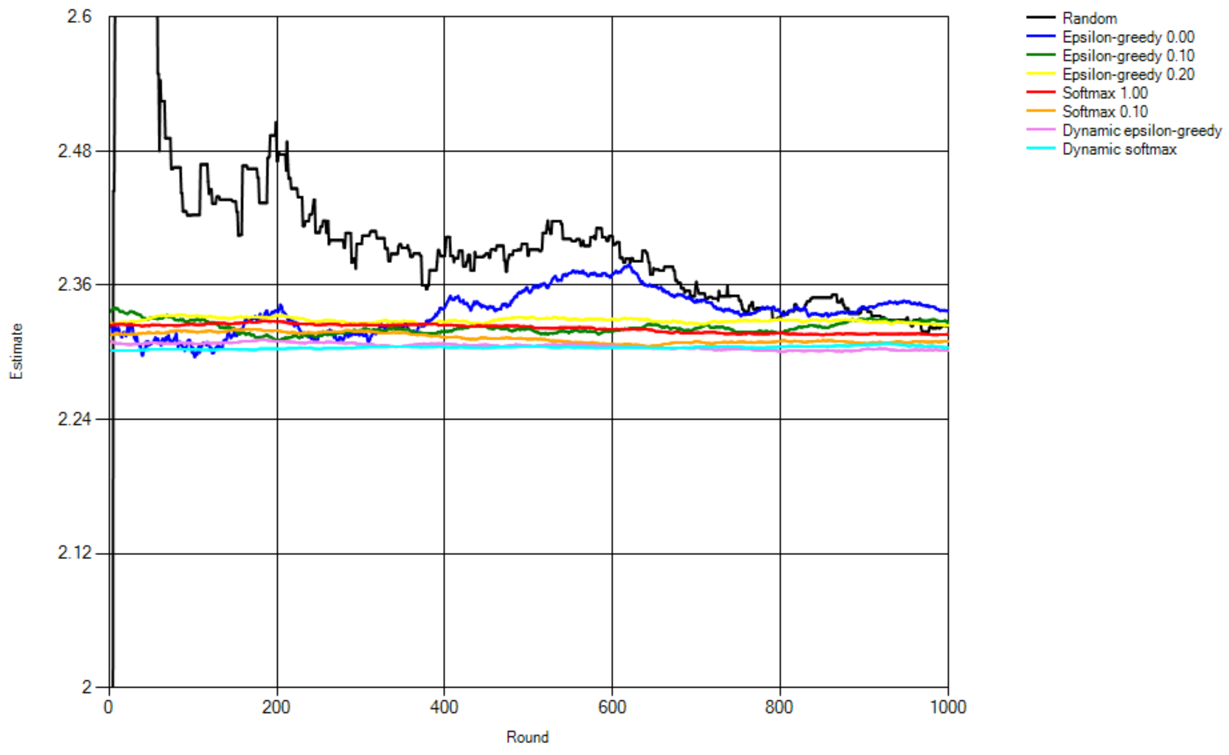
Reward per round



The algorithms achieve rewards in following descending order:

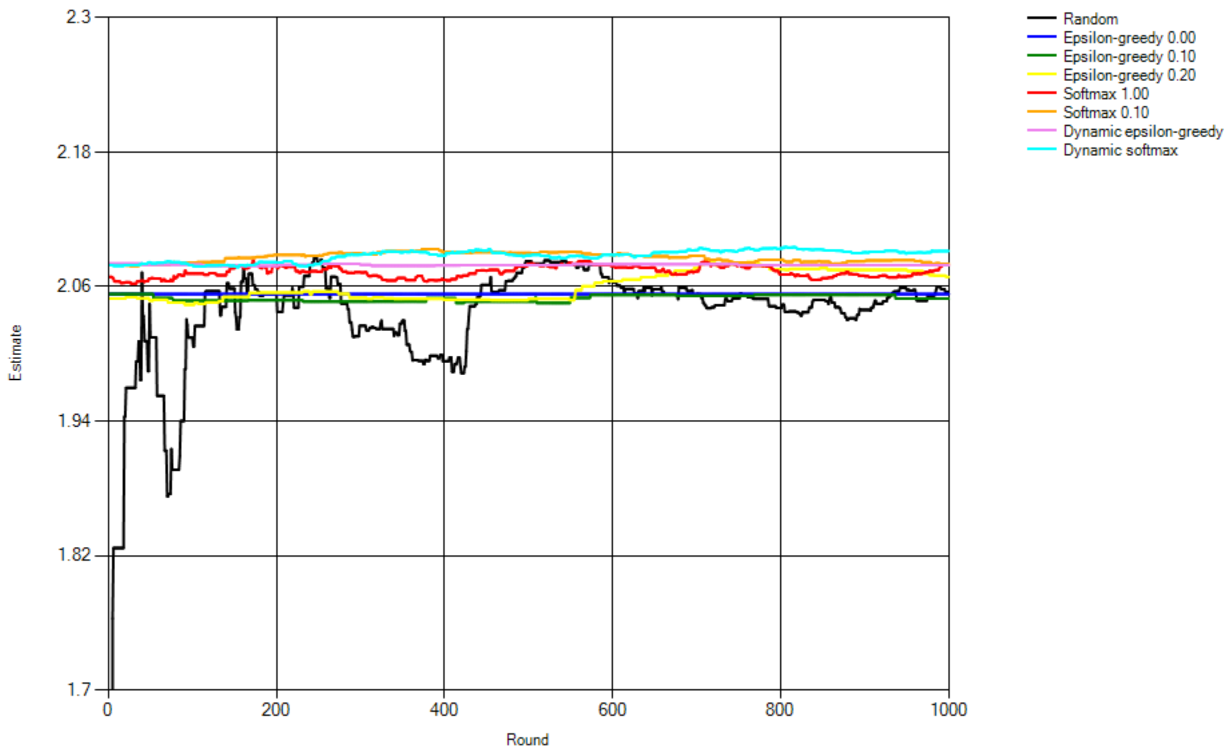
1. Epsilon greedy 0.0
2. Softmax 0.1
3. Dynamic epsilon-greedy
4. Epsilon-greedy 0.1
5. Epsilon-greedy 0.2
6. Dynamic softmax
7. Softmax 1.0
8. Random

Arm 1



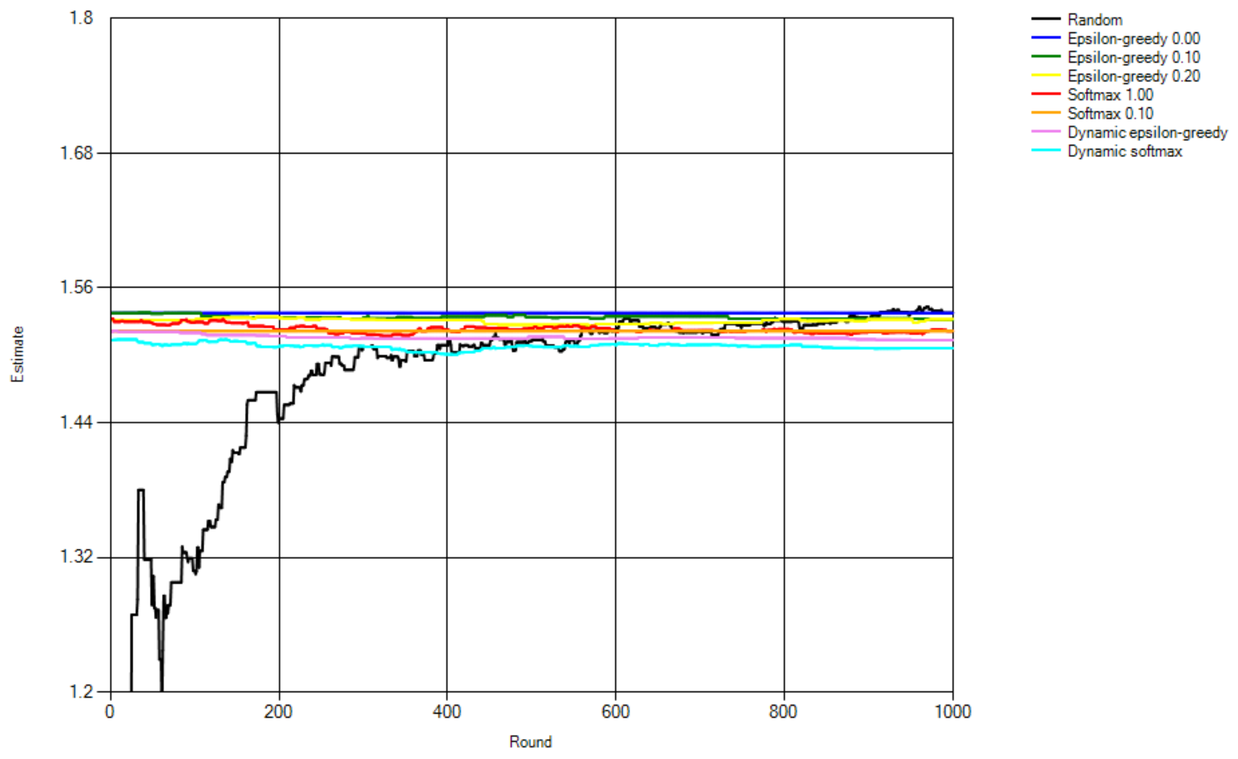
2.6
2.48
2.36
2.24
2.12
2
0
200
400
600
800
1000
Round
Estimate
Random
Epsilon-greedy 0.00
Epsilon-greedy 0.10
Epsilon-greedy 0.20
Softmax 1.00
Softmax 0.10
Dynamic epsilon-greedy
Dynamic softmax

Arm 2



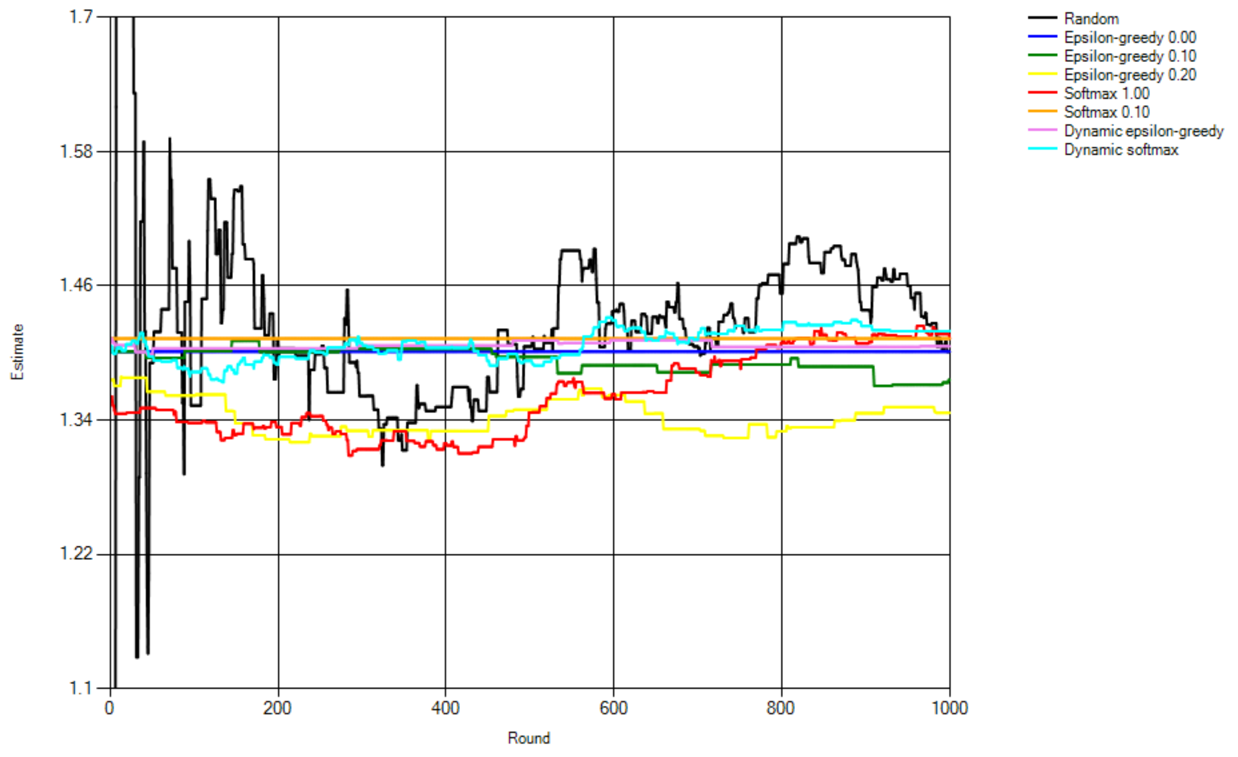
2.3
2.18
2.06
1.94
1.82
1.7
0
200
400
600
800
1000
Round
Estimate
Random
Epsilon-greedy 0.00
Epsilon-greedy 0.10
Epsilon-greedy 0.20
Softmax 1.00
Softmax 0.10
Dynamic epsilon-greedy
Dynamic softmax

Arm 3



1.8
1.68
1.56
1.44
1.32
1.2
Estimate
0
200
400
600
800
1000
Round
Random
Epsilon-greedy 0.00
Epsilon-greedy 0.10
Epsilon-greedy 0.20
Softmax 1.00
Softmax 0.10
Dynamic epsilon-greedy
Dynamic softmax

Arm 4



1.7
1.58
1.46
1.34
1.22
1.1
Estimate
0
200
400
600
800
1000
Round
Random
Epsilon-greedy 0.00
Epsilon-greedy 0.10
Epsilon-greedy 0.20
Softmax 1.00
Softmax 0.10
Dynamic epsilon-greedy
Dynamic softmax

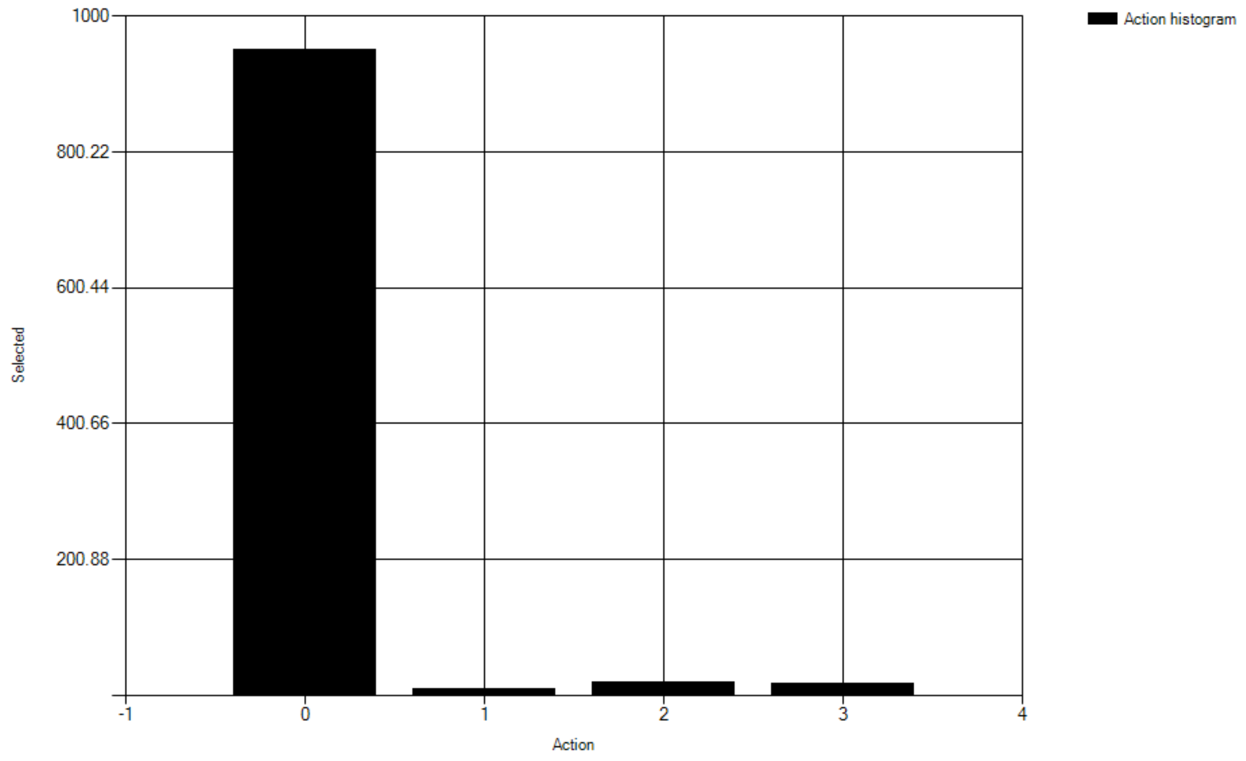
Based on research of these 4 arms estimations we can establish a table of ranks of the action selection algorithms.

Algorithm rank\Arm number	1	2	3	4
1	Dynamic softmax	Dynamic softmax	Dynamic softmax	Epsilon-greedy 0.2
2	Dynamic greedy	Softmax 0.1	Dynamic greedy	Epsilon-greedy 0.1
3	Softmax 0.1	Dynamic greedy	Softmax 0.1	Epsilon-greedy 0.0
4	Softmax 1.0	Softmax 1.0	Softmax 1.0	Random
5	Random	Epsilon-greedy 0.2	Epsilon-greedy 0.1	Dynamic greedy
6	Epsilon-greedy 0.2	Random	Epsilon-greedy 0.2	Softmax 0.1
7	Epsilon-greedy 0.1	Epsilon-greedy 0.0	Epsilon-greedy 0.0	Softmax 1.0
8	Epsilon-greedy 0.0	Epsilon-greedy 0.1	Random	Dynamic softmax

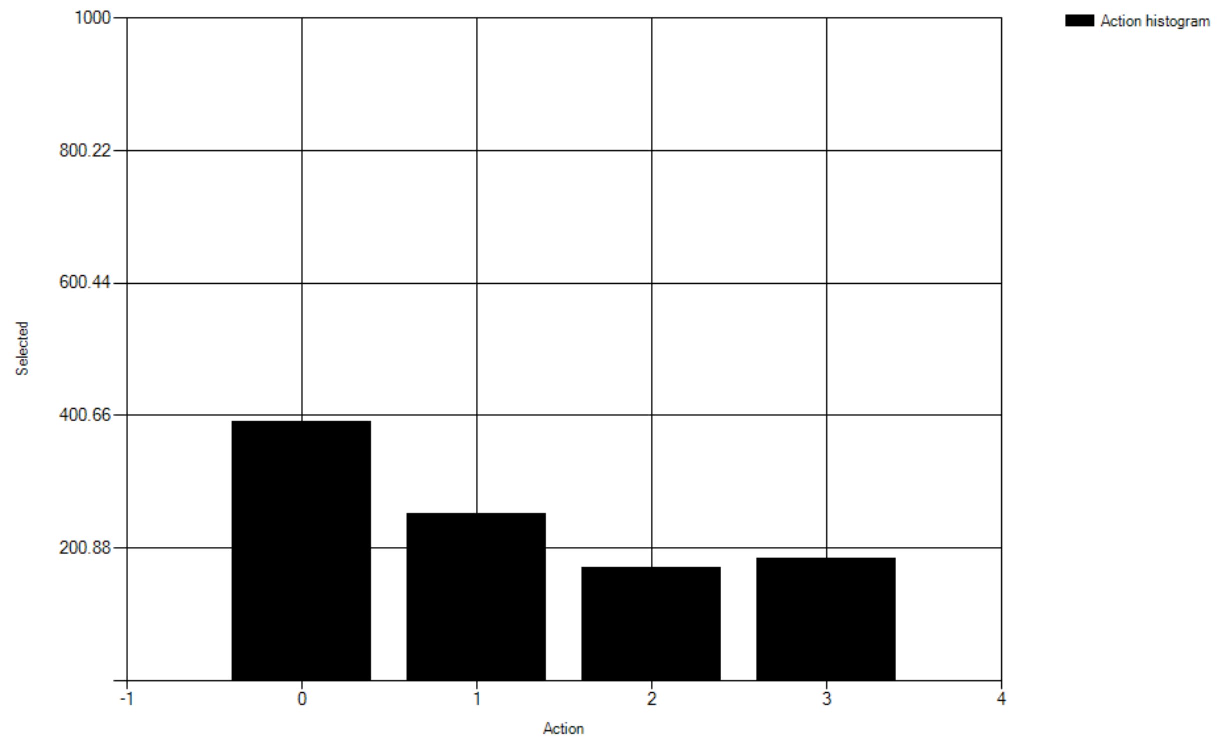
Despite the 2 new algorithms haven't proved to be the most profitable, they happened to be the most precise in terms of action estimation.

"Dynamic softmax" and "Dynamic greedy" tend to achieve a stable estimation faster than the others.

Dynamic epsilon-greedy



Dynamic softmax



In the action histograms we can see:

1. Dynamic epsilon-greedy is mostly meant for exploiting.
2. Dynamic softmax has minor exploiting and major exploration.

Stochastic Reward Game

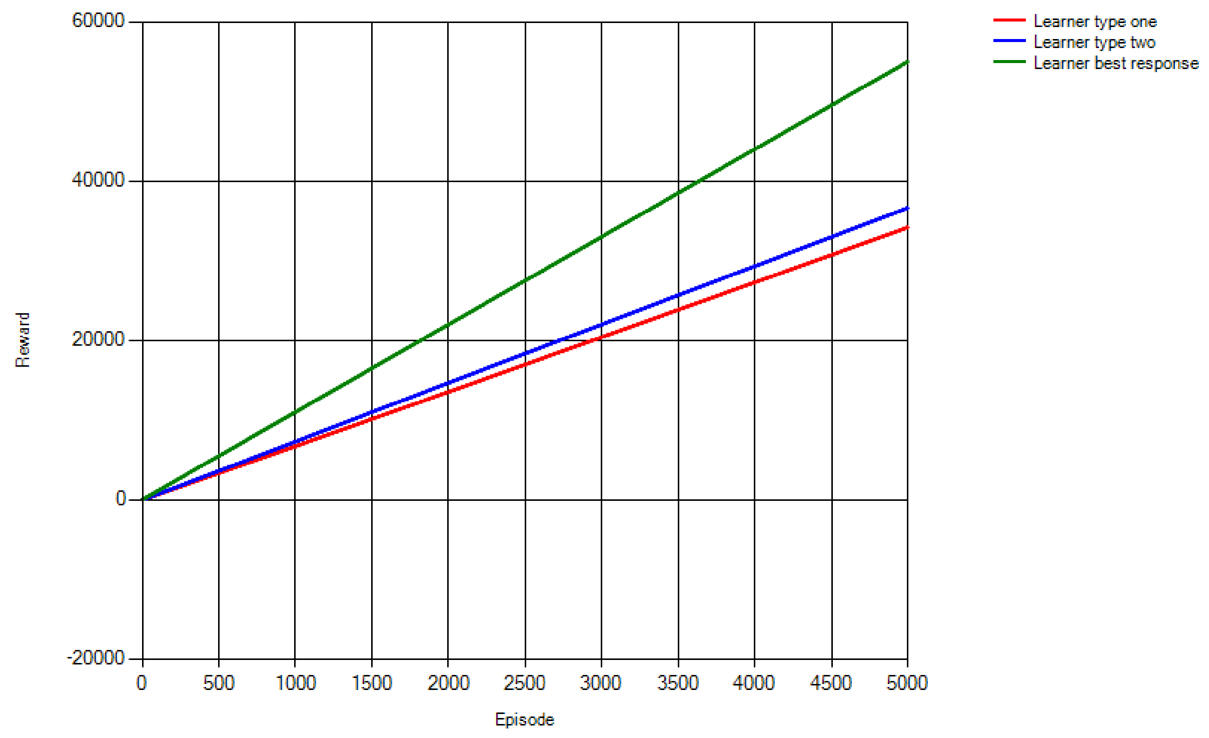
Two selection algorithms were implemented:

1. Simple Boltzmann action selection.
2. Optimistic Boltzmann action selection.
3. Best response selector (form question 2).

The simulation application has been developed. Testing was carried out for temperature parameter = 1. All plots were averaged over 100 iterations.

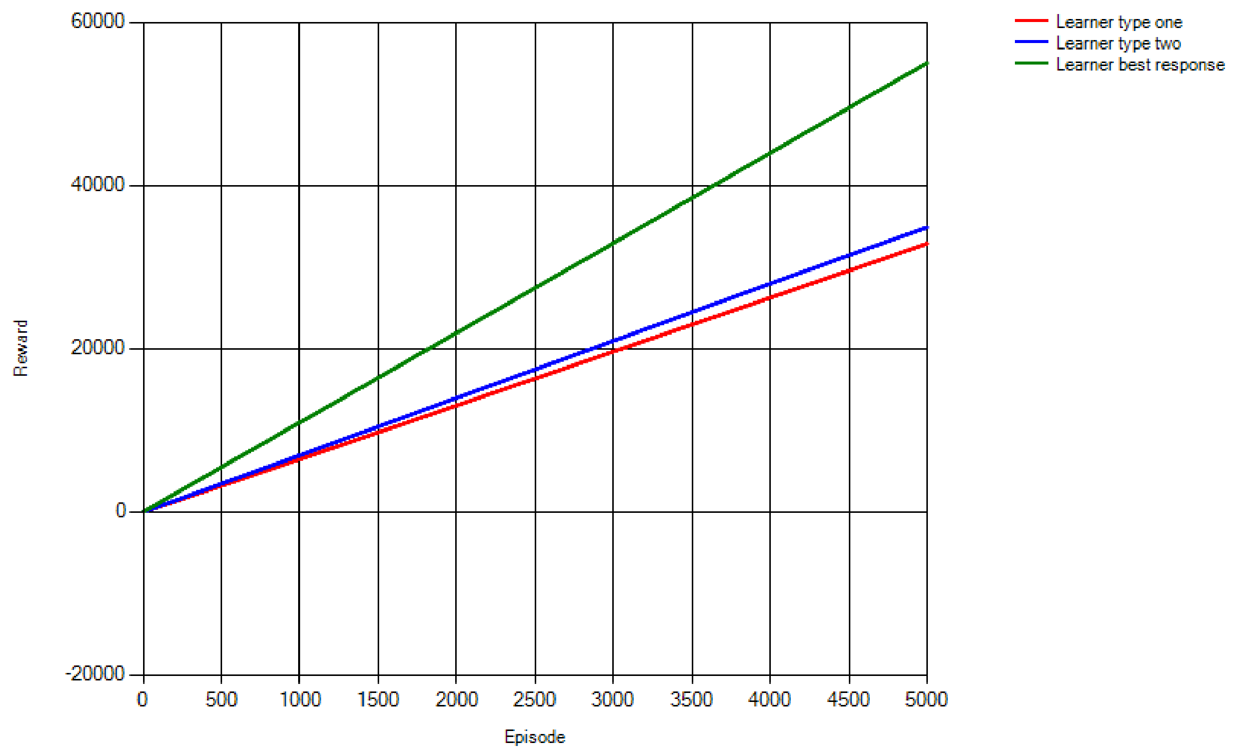
Following results were obtained:

Case 1



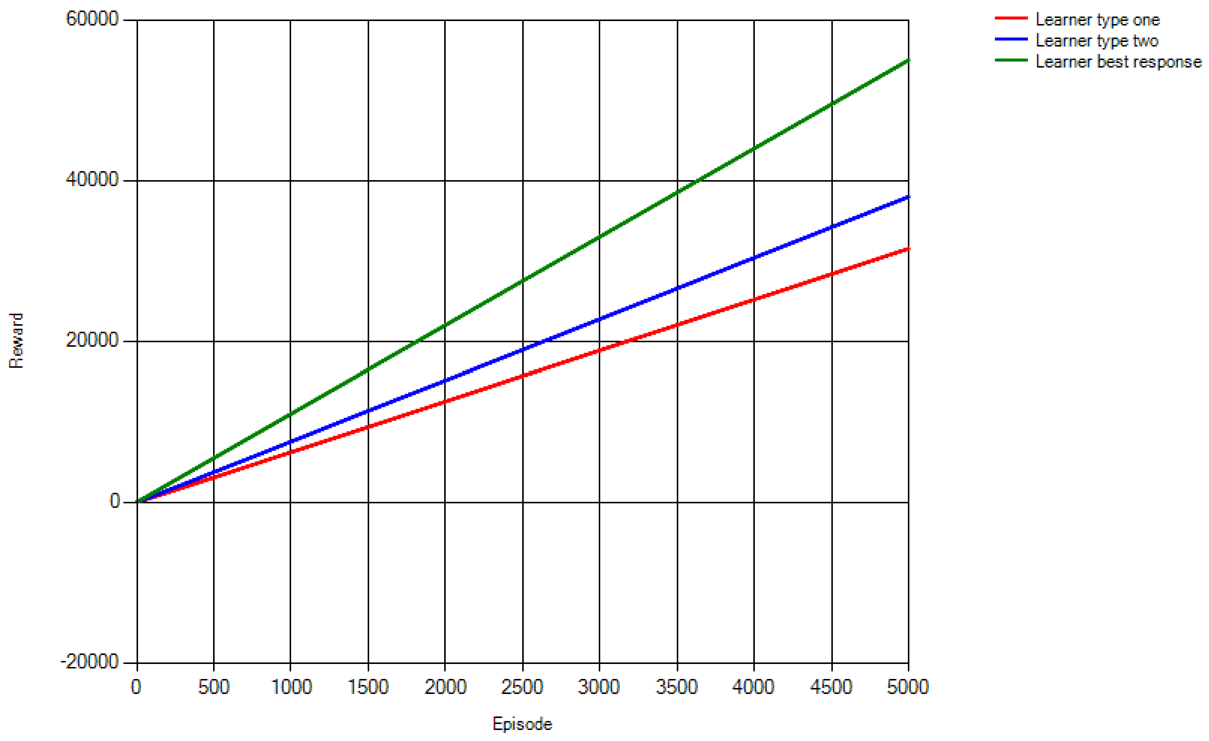
Learner type one
Learner type two
Learner best response
Reward
Episode
60000
40000
20000
0
-20000
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

Case 2



Learner type one
Learner type two
Learner best response
Reward
Episode
60000
40000
20000
0
-20000
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

Case 3



In all three cases results are roughly same and are represented by quasi-linear functions:

1. Best response.
2. Optimistic Boltzmann.
3. Simple Boltzmann.

Independent learning has not proved to be a more profitable selection strategy for the specified temperature parameter.

The best response strategy is the most efficient. However we assume that it is the most efficient because of the initial action table set-up. The most profitable action for this table is in cell [0; 0]. In the developed implementation of the algorithm if we encounter several equal estimated actions then we choose the first one which corresponds to the [0; 0] cell. Having a different action table set-up (e.g. where the most profitable action is in [1;1]) we may have a case when the learner does not do any exploration at all and exploits the first profitable action which may happen not the best one.

To conclude we can state that the effectiveness of learning strongly depends on the specified parameters of learning and actions set-up.