**Experiment 1.**

*First run.*

In the first experiment I decided to use Q- learning algorithm to compute Q values in Q table. Below I described results after first run.

Table 1 First Run Statistics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| **#** | **Moves taken** | **Policy** | **Bank Account** | **Time (sec.)** |
| 1 | 832 | PRANDOM | -416 | 46 |
| 2 | 1100 | PRANDOM | -684 | 63 |
| 3 | 582 | PRANDOM | -166 | 35 |
| 4 | 660 | PGREEDY | -244 | 42 |
| 5 | 350 | PGREEDY | 66 | 21 |
| 6 | 458 | PGREEDY | -42 | 28 |
| 7 | 216 | PGREEDY | 200 | 13 |
| 8 | 382 | PGREEDY | 34 | 24 |
| 9 | 444 | PGREEDY | -28 | 26 |
| 10 | 370 | PGREEDY | 46 | 20 |
| 11 | 402 | PGREEDY | 14 | 22 |

How we can see from the table in the first run experiment one reached terminal state 11 times (3 times with policy PRANDOM and 8 times with policy PGREEDY). We see that policy PGREEDY better than PRANDOM what is obvious. To analyze performance on each iteration I used number of moves taken before algorithm reached his terminal state.

Figure 1

From Figure 1 we see our algorithm learns to reach terminal state in less after each iteration, but after seventh iteration number of moves stays approximately on the same level. The best iteration is number 7. During that iteration algorithm reached terminal state in 216 moves and bank account contains reward equals 200. Why after seventh iteration number of moves increased needs to be explored in the future experiments and projects.

*Second Run.*

Table 2 Second Run

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Moves taken** | **Policy** | **Bank Account** | **Time (sec.)** |
| 1 | 796 | PRANDOM | -380 | 64 |
| 2 | 842 | PRANDOM | -426 | 82 |
| 3 | 1108 | PRANDOM | -692 | 87 |
| 4 | 368 | PGREEDY | 48 | 29 |
| 5 | 388 | PGREEDY | 28 | 29 |
| 6 | 422 | PGREEDY | -6 | 32 |
| 7 | 252 | PGREEDY | 164 | 19 |
| 8 | 314 | PGREEDY | 102 | 23 |
| 9 | 272 | PGREEDY | 144 | 20 |
| 10 | 458 | PGREEDY | -42 | 36 |
| 11 | 370 | PGREEDY | 46 | 29 |

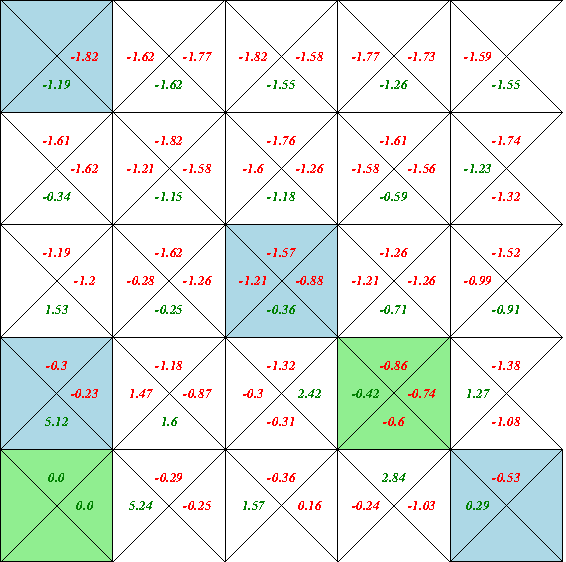
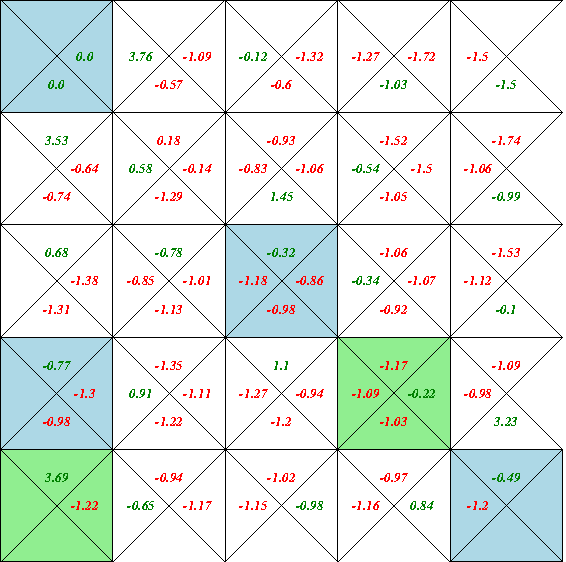
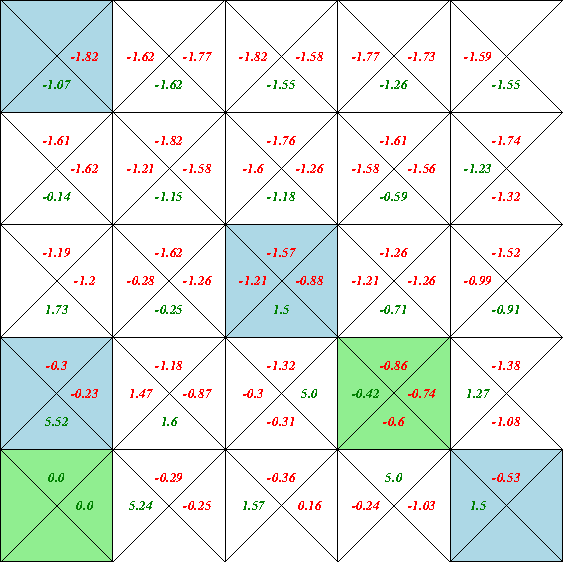
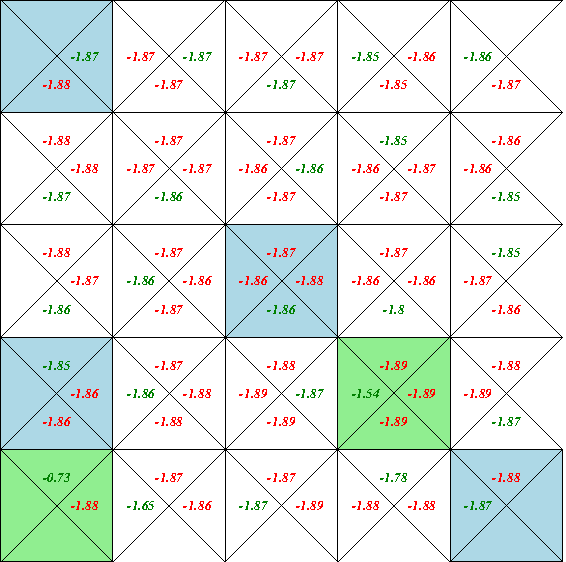
After second run algorithm reached terminal state 11 times (3 with policy PRANDOM and 8 with policy PGREEDY)

Figure 2

From Figure 2 we can notice big difference after 3 iterations, it is because we switched from PRANDOM policy to PGREEDY. Same as after first run the number of moves to reach terminal states equals approximately 400. To compare these 2 runs lets look on next figure:

We can notice that on iteration number 7 both runs reached terminal state with less moves. It may be because after 7 iteration our algorithm unlearns. After comparing these 2 runs, second runs perform a little bit better in last iterations, that’s why next I provide Q table values for second run.

First, I output Q -values in the middle of experiment (after 3000 steps) and at the end (after 6000 steps). On the next pictures you will see visual representation of Q table where numbers it’s Q values from Q table. Green number indicates maximum Q value for this state, red – other Q values for this state. Light blue boxes stand for Pick Up locations and Light green – Drop Off Locations. In my algorithm I use two Q tables. One Q table contains Q values when agent carries the block and the other when agent doesn’t carry the block. Figure 3 represents Q table after 3000 steps when agent carries the block and when agent doesn’t carry the block. On Figure 4 we can see final Q table after 6000 steps.



Q table for agent NO BLOCK after 6000 steps

Figure 3 Q table for agent WITH BLOCK after 3000 steps

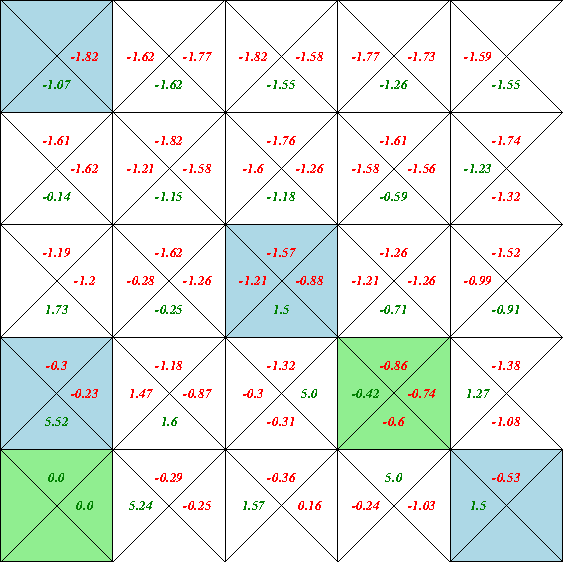
Figure 4 Q table for agent WITH BLOCK after 6000 steps

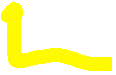
Q table for agent NO BLOCK after 3000 steps

**Results.**

How we can see in Q tables after 6 000 steps Q table creates paths to pick up locations if agent doesn’t carry the block and to drop off locations overwise. For example, if agent picks up block in location (1,1) we can see that max Q values will lead him to drop off location (1,5). With same technique we can see path if agent doesn’t carry the block to the pick-up locations

Figure 4 Q table for agent WITH BLOCK after 6000 steps





Unfortunately, when agent doesn’t have a block some paths may lead agent to the pick-up location what doesn’t contain any blocks, that’s why Q table unlearn some paths and creates some loops. But agent never get stuck in this loop because after each step reward is -1 and eventually he will get out from loop even with PGREEDY policy.

**Experiment 2**

In the experiment 2 I ran Q learning for 200 steps with PRANDOM policy and next 5800 steps with PEPLOIT policy. I reported Q tales after first Drop Off location was filled and after algorithm reached first termination state. Like in previous experiment I run experiment two times and reported results in the same manner as in Experiment 1.

After first run I observed following statistics:

Table 3 First Run

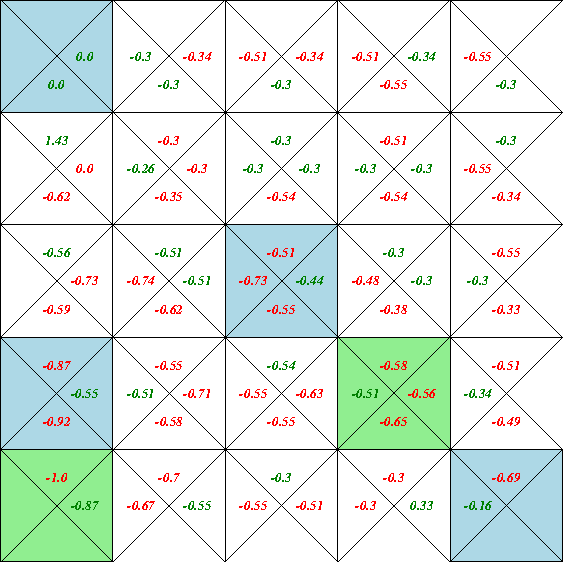
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Moves taken** | **Policy** | **Bank Account** | **Time (sec.)** |
| 1 | 506 | PEPLOIT | -90 | 158 |
| 2 | 358 | PEPLOIT | 58 | 20 |
| 3 | 386 | PEPLOIT | 30 | 21 |
| 4 | 216 | PEPLOIT | 200 | 12 |
| 5 | 210 | PEPLOIT | 206 | 11 |
| 6 | 428 | PEPLOIT | -12 | 24 |
| 7 | 376 | PEPLOIT | 40 | 21 |
| 8 | 606 | PEPLOIT | -190 | 35 |
| 9 | 812 | PEPLOIT | -396 | 47 |
| 10 | 522 | PEPLOIT | -106 | 29 |
| 11 | 618 | PEPLOIT | -202 | 34 |
| 12 | 472 | PEPLOIT | -56 | 31 |

Table 4 Second Run

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Moves taken** | **Policy** | **Bank Account** | **Time (sec.)** |
| 1 | 458 | PEPLOIT | -42 | 116 |
| 2 | 310 | PEPLOIT | 106 | 24 |
| 3 | 278 | PEPLOIT | 138 | 21 |
| 4 | 366 | PEPLOIT | 50 | 28 |
| 5 | 410 | PEPLOIT | 6 | 31 |
| 6 | 282 | PEPLOIT | 134 | 21 |
| 7 | 334 | PEPLOIT | 82 | 24 |
| 8 | 566 | PEPLOIT | -150 | 43 |
| 9 | 346 | PEPLOIT | 70 | 26 |
| 10 | 362 | PEPLOIT | 54 | 28 |
| 11 | 630 | PEPLOIT | -214 | 49 |
| 12 | 790 | PEPLOIT | -374 | 60 |
| 13 | 424 | PEPLOIT | -8 | 32 |

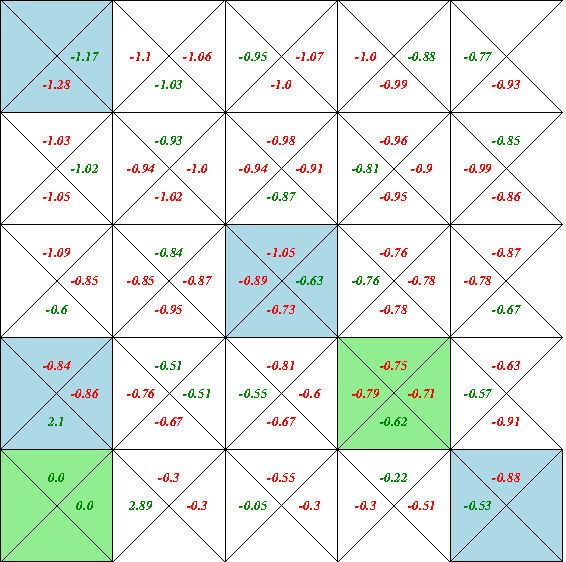
Now I compare this two runs and use demonstrate Q tables of best one.

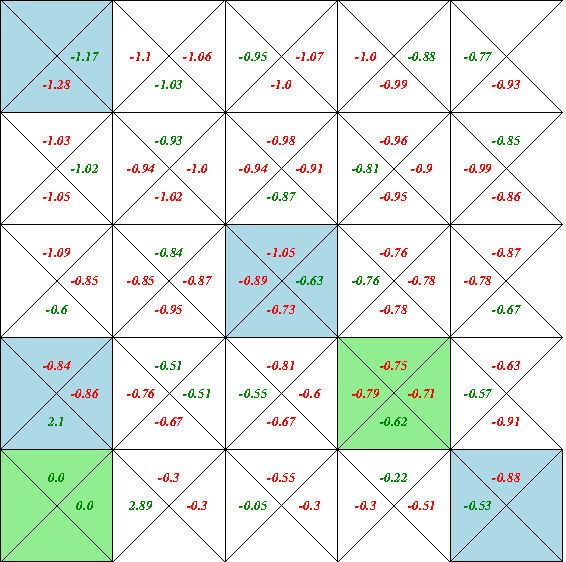
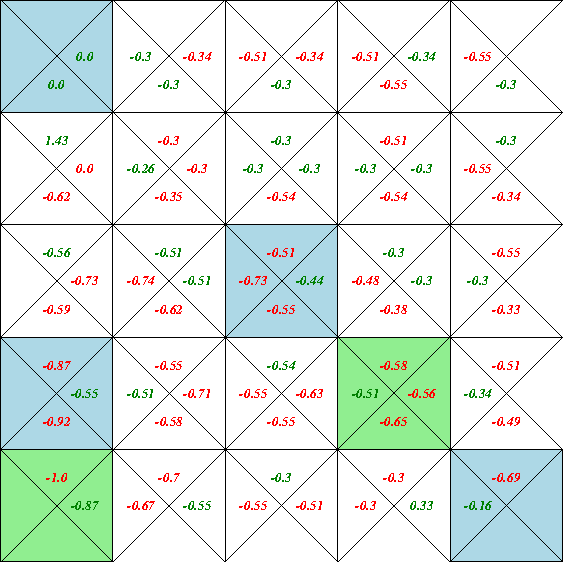
How we can see from charts and tables above, during second run algorithm reached terminal state 13 times, during first run terminal state reached 12 times. We also see that the minimum steps taken to reach terminal state was between 2-6 iterations. First run took less steps to converge during these iterations, that’s why I choose Q tables from first run to analyze. I noticed that after 5-6 iteration number of steps to reach terminal state increases, I assume it is because we overfit our Q table and unlearn paths.

In next figure we can see Q table after first Drop Off location filled.

Q table for agent NO BLOCK after first Drop Off filled

Q table for agent WITH BLOCK after first Drop Off filled



Next Q tables I reported after first terminal state was reached

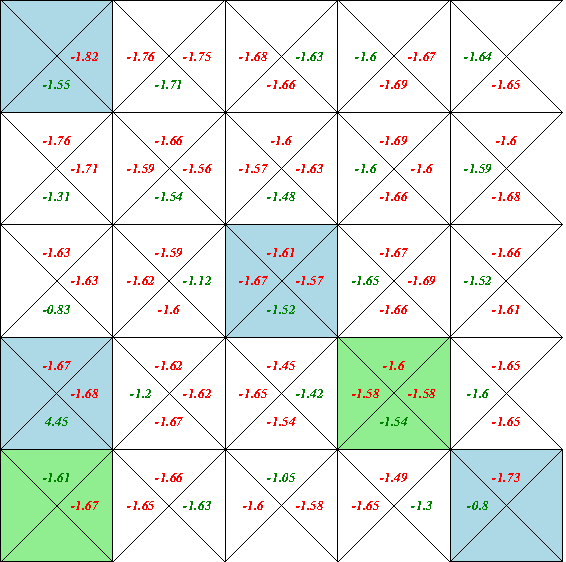
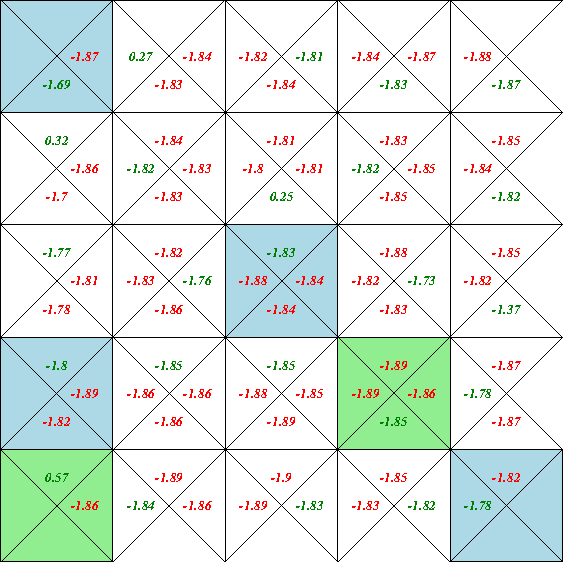
Q table for agent NO BLOCK after first termonal state reached

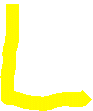
Q table for agent WITH BLOCK after first terminal reached

From this tables we can see how attractive paths start to form, but they still contain a lot of loops. Let’s see how same Q tables will look in the end of experiment.

Q table for agent NO BLOCK after 6000 steps

Q table for agent WITH BLOCK after 6000 steps





**Results.**

How we can see from Experiment 2 policy PEPLOIT allows agent to do more exploration, in result after few iterations experiment produces better results than Experiment 1.But after 7-8 iterations Q table start to unlearn attractive paths what in result increase number steps needed to reach terminal state. Also, after first Drop Off location filled its too early to say that or Q table learn anything, because how we can see form these Q tables we can’t for sure determine clear paths to Drop Off or Pick Up locations. The best result was achieved on First Run at 5 iterations (algorithm reached terminal state after 210 steps and bank account value +206)

**Experiment 3**

In this experiment I used SARSA instead of Q learning to compare these two algorithms. In next tables we can see results. I ran this experiment twice and reported both statistics and only best run Q tables.

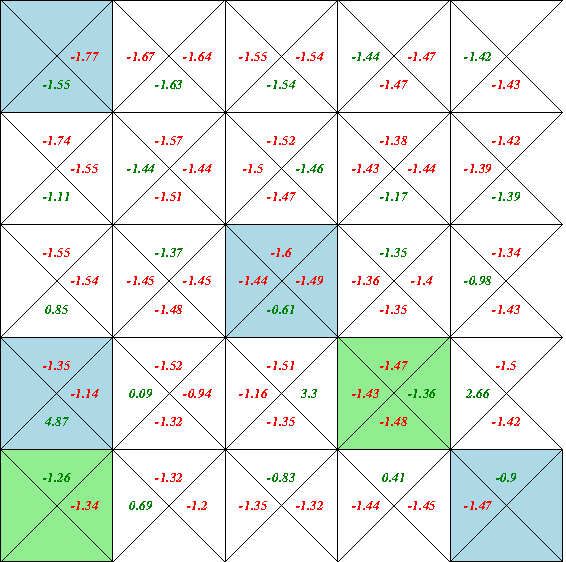
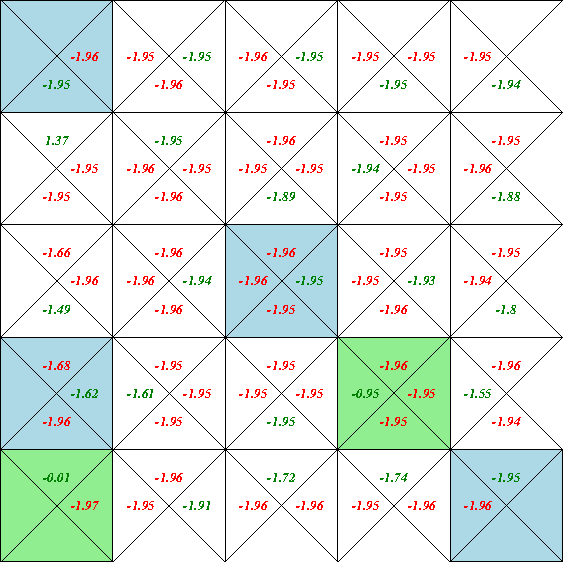
Table 5 First Run

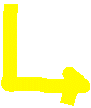
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Moves taken** | **Policy** | **Bank Account** | **Time (sec.)** |
| 1 | 494 | PEPLOIT | -78 | 37 |
| 2 | 340 | PEPLOIT | 76 | 25 |
| 3 | 440 | PEPLOIT | -24 | 33 |
| 4 | 448 | PEPLOIT | -32 | 33 |
| 5 | 410 | PEPLOIT | 6 | 30 |
| 6 | 554 | PEPLOIT | -138 | 40 |
| 7 | 658 | PEPLOIT | -242 | 49 |
| 8 | 384 | PEPLOIT | 32 | 28 |
| 9 | 600 | PEPLOIT | -184 | 43 |
| 10 | 488 | PEPLOIT | -72 | 35 |
| 11 | 618 | PEPLOIT | -202 | 47 |

Table 6 Second Run

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Moves taken** | **Policy** | **Bank Account** | **Time (sec.)** |
| 1 | 518 | PEPLOIT | -102 | 39 |
| 2 | 232 | PEPLOIT | 184 | 17 |
| 3 | 334 | PEPLOIT | 82 | 24 |
| 4 | 552 | PEPLOIT | -136 | 42 |
| 5 | 534 | PEPLOIT | -118 | 32 |
| 6 | 382 | PEPLOIT | 34 | 22 |
| 7 | 582 | PEPLOIT | -166 | 46 |
| 8 | 596 | PEPLOIT | -180 | 57 |
| 9 | 586 | PEPLOIT | -170 | 57 |
| 10 | 354 | PEPLOIT | 62 | 32 |
| 11 | 930 | PEPLOIT | -514 | 85 |

How we can see from tables in the both runs Experiment 3 reached terminal state 11 times. To compare these two runs lets look on following figure:

Second run provides better results, so I will use second run Q tables.



Q table for agent NO BLOCK after 6000 steps

Q table for agent WITH BLOCK after 6000 steps

**Results**

The Experiment 3 produces not better results than Experiment 1 or 2 but we can see good improvement from the first iteration and that result carries through all experiment. The best iteration was on second run (2 iteration it took 232 steps to reach terminal state and bank account value is 184). In conclusion, Q learning produces better results than SARSA variation.

**Comparison.**

To compare results, I created graph with statistics of best runs.

How we can see the best results showed Experiment 2, but after 5 iteration Q table start to unlearn attractive paths. Also, from chart we can notice what SARSA variation of Q learning not performing better than Q learning, but SARSA model showing good results right after 1 iteration. In conclusion best results produces Experiment 2 because of policy PEPLOIT what allows agent to do more exploration compare to PGREEDY policy.