Assignment 2

Akshay Khadse
Roll Number: 153079011

Project Structure:

```
153079011/
|--> data/           original  MDP files and solutions
|--> generated/      contains generated MDP Files
|--> results/        results from the trials on generated files
|--> generate.py     script to generate new MDP instances
|--> get_results.sh  script to perform trials of different algorithms
|--> planner.py      implementation of required algorithms
|--> planner.sh
```

Generation of MDP Files:

- `generate.py` script is used to generate new randomised instances of MDP. Each such file generated is placed in `generated/` directory
- Number of states and number of files to be generated is specified near the topmost part
- Following are the requirements for an MDP to be valid
  - There must be at least one transition from each state
  - Sum of transition matrix entries (between 0 and 1) from a state and a particular should be 1
  - Rewards can be negative as well as positive
- From line 44 onwards, first a different random seed is generated for each file to be generated. Then the transition and reward function arrays of appropriate sizes are initialized with all zeros.
- For each initial state and each action, a sequence of 0s and 1s is generated such that it has at least one entry as 1 and length of this sequence is equal to number of states. This is done to ensure that there is at least one transition from each state.
- For obtaining valid transitions for given state and action, a random vector of length equal to total states is generated such that each entry is between 0 and 1. This vector is then element wise multiplied with the sequence of 0s and 1s generated from previous step, to ensure that there are some non reachable states. Then, this vector is divided by its sum to make the sum of all elements equal to 1.
- For each state and action pair, rewards are generated by subtracting 1 from each value of a random vector whose entries lie between 0 and 2. This is multiplied by the sequence of 1s and 0s to ensure there are no rewards for non reachable states.
- From line 11 to 44 is the function to write the file according to arrays generated.

- Gamma was generated as a random sample from uniform distribution between 0 and 1 and written upto 2 digits in MDP file.
- Snippets from `_randDense` function of MDP Toolbox documentation was used for above implementation [reference 1].

Comparison:

100 sample MDPs with 50 states and 2 actions were generated by using above logic and each algorithm was applied for each MDP and the number of iterations observed were averaged for each algorithms across all the MDPs. Following are the results observed:

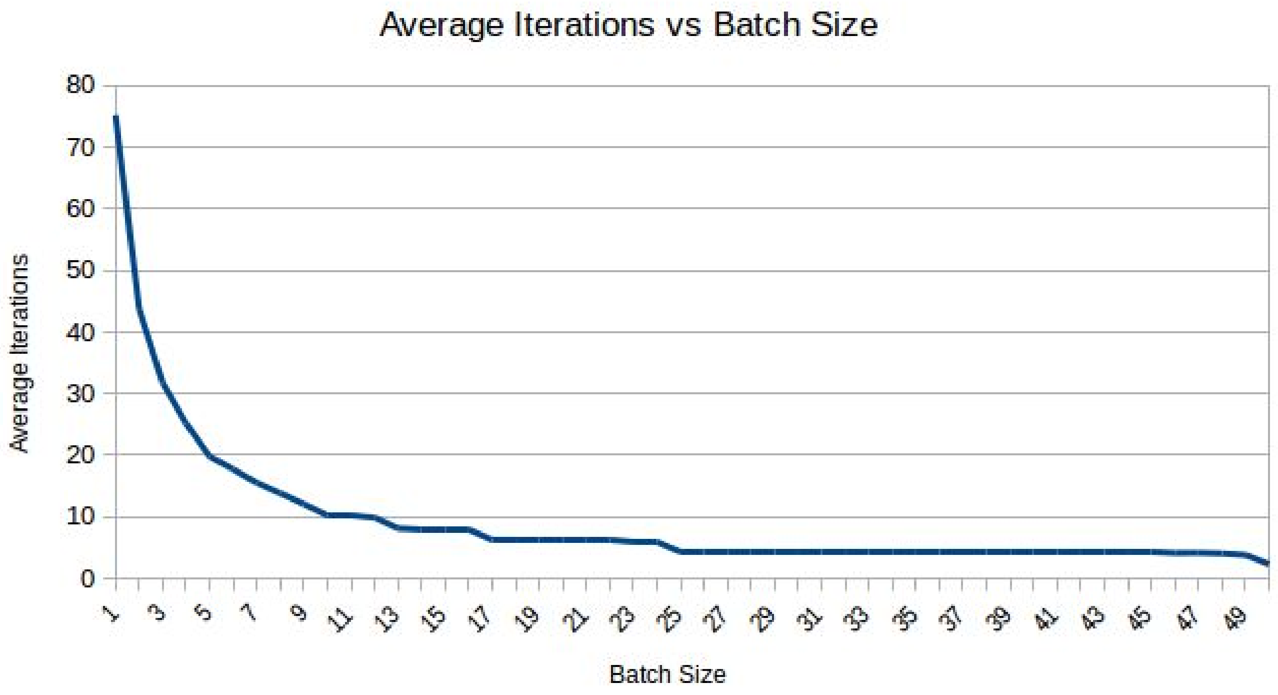
Algorithm	Average Iterations	Max	Min
Howard's PI	2.31	3	2
Randomised PI (trial 0)	4.68	7	3
Randomised PI (trial 1)	6.73	9	3
Randomised PI (trial 2)	4.76	8	2
Batch Switching PI (batch size 50)	2.31	3	2
Batch Switching PI (batch size 40)	4.17	6	4
Batch Switching PI (batch size 30)	4.19	5	4
Batch Switching PI (batch size 25)	4.2	5	4
Batch Switching PI (batch size 20)	6.2	7	6
Batch Switching PI (batch size 15)	8.06	10	7
Batch Switching PI (batch size 10)	10.09	11	10
Batch Switching PI (batch size 5)	19.78	21	18
Batch Switching PI (batch size 1)	75.26	83	68

Howard's policy iteration takes least number of iterations 2.31 on average. This is same as running a Batch switching PI with a batch size of 50, which asserts that its implementation is correct. However, this number increases as the batch size decreases. More trials were done to get an idea of this trend.

Randomised policy iteration is taking 4.68 iterations on an average. However, this average can change for different trials. The maximum of 7 iterations and minimum 3 iterations were required to solve MDPs in the `generated` folder using this algorithm.

Effect of batch size was observed by running BSPI on each MDP with batch size varying from 1 to 50 in steps of 1 and the number of iterations was averaged over all the MDPs for each batch size used. These results can be found in `results/batchsize.csv` where each row represents a MDP file and each column represents the number of iterations.

Following is a graph of results:



Observations:

- Howard’s PI always takes at most 3 iterations for 2 action MDP problem. Here, minimum of 2 iterations were required.
- Average iterations for Randomised PI were observed to be 4.68 in first trial 6.73 in second and 4.76 in third. Thus, it can be seen that the maximum and minimum number of iterations required in these cases is also different for the same MDP problems. This is due to the random seed for each application of the algorithm through `get_results.sh`
- For Batch Switching PI, it can be observed from the above graph that with lowest possible batch size it takes 75.26 iterations to solve a 50 state 2 action MDP. This number then drops quickly as batch size is increased and reaches below 10 for batch size 13 and more. The rate of decrease then does not change as much. On further increment in batch size by 37, minimum is reached at 2.31 iterations when batch size is 50. This trend in agreement figure 2(b) of reference 2.

References:

1. MDP Toolbox Documentation http://pymdptoolbox.readthedocs.io/en/latest/_modules/mdptoolbox/example.html#rand
2. Batch-Switching Policy Iteration, Shivaram Kalyanakrishnan, Utkarsh Mall, and Ritish Goyal, IJCAI 2016, pp. 3147-3153