LAB 4

Kumar Ayush - 2015eeb1060

Sumit Singh - 2015csb1036

Course Instructor - **Dr. Narayanan C. Krishnan**

Indian Institute of Technology, Ropar

1. Shakey the Robot

How to run

- Language: **Python 3.6.4 (64 bit)**
- Run: **$ python csl302_l4.py <*input_file*>**
- Example: **$ python csl302_l4.py 1.txt**
- Note: **Don't** delete *action.txt* file. Keep it in the **same folder** as *csl302_l4.py* file
- *action.txt* describes the actions that our agent can take. One slight change we did in one of the actions is as described below.

Method

Implemented in Python. Used three different classes for world, action and propositions. Implemented forward planner using both BFS and DFS. Goal stack planner was implemented using recursion. Also goal state was considered as a *subset* of a current state and not the *exact* given state.

Note: Slight change in following actions that we made:

Turnon/turnoff(l, x) to **Turnon/Turnoff(x, b)**: (Pre and post conditions are exactly same). x is the room and b is the box.

Assessments and Measures

Testcase #1 (f):

- 5 Actions
- 15 Nodes Expanded

- 0.015626907348632812 seconds

Testcase #2 (g):

- 5 Actions
- 5 Nodes Expanded
- 0.015604496002197266 seconds

Testcase #3 (f):

- 5 Actions
- 90 Nodes Expanded
- 0.031250715255737305 seconds

Testcase #4 (g):

- 5 Actions
- 5 Nodes Expanded
- 0.015644311904907227 seconds

Testcase #5 (f):

- 6 Actions
- 88 Nodes Expanded
- 0.03126645088195801 seconds

Testcase #6 (g):

- 8 Actions
- 8 Nodes Expanded
- 0.0 seconds

Testcase #7 (f):

- 8 Actions
- 146 Nodes Expanded
- 0.031252145767211914 seconds

Testcase #8 (g):

- 10 Actions
- 10 Nodes Expanded
- 0.015625 seconds

Testcase #9 (f):

- 10 Actions
- 523 Nodes Expanded
- 0.1718747615814209 seconds

Testcase #10 (g):

- 12 Actions
- 12 Nodes Expanded
- 0.015627145767211914 seconds

*f: forward planner, g: goal stack planner

Observation

One of the apparent difference we see that the number of nodes expanded is **significantly larger** in case of *forward planner* than in case of *goal stack planner*. Reason: Since goal stack planner directly takes the path from goal to initial state, it expands only the *relevant nodes* while forward planner keeps expanding *every node* until it reaches a solution. In case of forward planner, nodes expanded increases *exponentially* as evident in test case #9.
One of advantage we see in forward planner is the number of actions it took is **slightly less** than that in case of goal stack planner. Reason: Forward planner is

implemented using BFS so it will always have least number of actions required to reach the goal. In goal stack planner, due to its nature of algorithm, it might undo some goals in search of another goal. So, the results are not always optimum but is generally close to optimum.
One more advantage goal stack planner have over forward planner is the time taken to reach a plan. Since time taken is directly proportional to nodes expanded, goal stack planner takes almost **half as time** as forward planner to reach a valid plan.

2. Probabilistic Inference using Bayesian Networks

How to run

- Language: **Python 3.6.4 (64 bit)**
- Run: **$ python chowliu.py**
- Note: Keep data.txt and data_test.txt in the **same folder** as *chowliu.py*

Method

Implemented in Python. Gibbs samples generated by varying the number of samples and iterations per sample.

Assessments and Measures

Table 1: Marginal error and population estimate using gibbs samples

Samples	Iteration	correct population est.	wrong population est.	marginal error
100	100	1	0	0.01274731707
100	120	1	1	0.01232439024
100	150	1	0	0.01178308943
100	200	1	1	0.01096715447
100	400	0	1	0.01170243902
100	500	1	1	0.01108813008
150	100	1	0	0.01316856369
150	200	3	0	0.01120791328
200	100	3	1	0.01087252033
200	150	1	0	0.01245121951
200	200	2	1	0.0109100813
200	1000	14	0	0.000455678899
300	120	2	0	0.009878807588
3000	100	26	13	0.003683922038

Table 2: Likelihood of sample generated

Samples	Iteration	Likelihood error
100	100	0.00013830264
100	150	0.0001387593236
200	150	0.0001376538546
200	100	0.0001389151855
200	200	0.0001382664001

Observation

We immediately observe that increasing the number of iterations directly impact the marginal probability error that we calculated. Though it was also somewhat increase the number of correct population estimate but it's not by much, especially when the number of samples is quite less, although it's more likely to generate less incorrect test sample. One anomaly is when number of iterations is 1000. It's because the samples generated are entirely at random so in some cases, it might match the test sample.
Increasing the number of samples generated impacted marginal error much less than it did on number of similar samples generated compared to the test data. With increase in number of samples it is more likely to generate a test data. Large number of samples (>1000) with large number of iterations(> 500) generate much more accurate distribution samples and even generate samples matching with test data with higher probability.
Likelihood error is the difference between the probability of occurrence of test sample in test data and gibbs sampling generated data. It generally decreases with increase number of samples generated and have a minor impact when change in iterations when number of iterations are large(> 500).