Approximate Q learning:

We have tried to use approximate Q learning to solve our problem. Since our problem have clearly defined state, actions, world and reward functions, we thought the approximate Q learning would work in our problem like it did in AI course programming homework. With respect to the Q functions, given that whether the methods will work is unknown we choose to try the simplest function—linear function

and if linear method works fine on our problem, we may try more complex functions such as polynomial function and neural network to get better result. After Q function is decided we use the following formula to update our Q value and weights

At this point, the last thing is feature generation. Given the state and legal actions is limited, we generate seven features. The first is the number of orders agent are receiving at this state. The second is the number of orders agent rejects at this state. These two feature is designed to find a correlation between the agent’s Q value and its action about dealing with the new orders. The third feature is the number of orders agent has already received but still in the restaurant which means the agent is on the way to pick order up at a restaurant. The fourth feature is the number of orders the agent is already carrying. We hope these two feature helps relate the agent’s current order state with Q value. The fifth feature is the distance between the agent and the destination, here we use breath first search to find the nearest path since our map is quite small. By this feature, we hope to relate our Q value with the action of going back to destination which is quite important in our problem. The sixth feature is the number of orders which is still in the legal deliver time while the seventh feature is the the number of orders which is already time out. Through these two features, the status of the orders agent is carrying or received is taken into consideration. For the hyper parameters, we choose 0.1 as our learning rate, 0.8 as discounting factor and 0.5 as exploration factor since we realize the state space of our problem is huge so we hope the algorithm may explore faster to get desirable result.

In the experiment part, though our design of methods make the best use of the state and legal actions we have, the result of the experiment is not satisfying. The weights of our features keeps growing exponentially and finally reach positive or negative infinite randomly. The score of our state is slightly floating around the minimum score.

“””

change the following data to form and remove the words (iter 0

dict\_values([0.08179253966435338, 0.0030604646790041014, -4.47665398605479, 0.0, -2.2488563708755076, -0.6717629491851815, -3.804891036869607])

iter 10

dict\_values([-31527384637.342777, 1.1834620556519453e+19, -1.4418236136249996e+20, -13055379.448204614, -4.856847218996233e+20, 1.6096521153621022e+16, -1.4419845788366664e+20])

score = -120

iter 20

dict\_values([3.0834615716025046e+110, 6.106525465952827e+119, 9.726676148199692e+119, -3.013645184376389e+111, -1.0170082130155232e+120, -1.8041915041151543e+112, 9.726676298482382e+119])

score = -135

iter 30

dict\_values([-1.2195361811401935e+174, 3.4749605915164014e+185, -1.2613315901530267e+187, -8.408877267689702e+186, -4.313132484388226e+187, -5.664794679087079e+179, -2.10221926027405e+187])

score = -120

iter 40

dict\_values([-1.5748200465298788e+231, 1.9654794419707615e+230, 2.7529187176641007e+231, 1.8678620979122397e+226, -5.661366502426272e+230, 5.7686045222751186e+231, -3.0156671259900147e+231])

score = -135

iter 50

dict\_values([-1.4765748148601626e+230, 1.9649439943759436e+227, 1.8615011811415014e+229, 1.8678620979122397e+226, -8.61660725704951e+227, 3.7721618983395963e+229, -1.9087928550980204e+229])

score = -120

iter 60

dict\_values([8.49584025141372e+252, 1.0682436113166877e+256, -1.6594865937741988e+257, -2.3429353612426437e+229, -5.544588314614429e+257, -9.838323823469948e+253, -1.6585027613918518e+257])

score = -120

iter 70

dict\_values([-6.824075517742546e+278, 3.669868013385737e+275, -1.5174157614591902e+278, 5.825199525533636e+278, -2.483275136493153e+276, 2.8098354101078584e+278, 1.497948353966537e+278])

score = -120

)

“””

After we carefully investigate the action of our agent, the result show that the agent keeps walking back and forth between the destination (school) and the position next to it, which marks the end of our trial on approximate Q learning. The possible reasons for the failure we concluded are the state space is too large for the agent to learning with limited iteration given our handful computation power and the features we extracted from state action is still not good enough to drive the agent act normally. Finally, we estimate the cost of optimizing the approximate Q learning and the feasibility of the method. We decided to try other methods for solving our problem.