# Tutorial 5 Strategy for Project 3

Two key issues need to be addressed:

1. How are rewards modelled?
2. Transitions between states and assignment of probabilities to these transitions

## Modelling rewards

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | R | B |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | T |  |  |  |
| C |  |  |  | D |

1 2 3 4 5

5

4

3

2

1

Attach a Boolean valued flag to a customer at customer generation time.

E(A) = 1 if request comes from location A.

=0 otherwise

Track customer type (regular or premium)

T(A) = 1.5 if E(A) = 1 and customer is premium

= 1.0 if E(A) = 1 and customer is regular

The reward for any given customer, say A is now given by:

R(A) = E(A)\*20\*T(A)

Similarly, rewards can be assigned to other customers B, C and D.

## Modelling transitions

First, we will define a blocked position as one of:

1. A wall
2. A position that is reached by a move opposite (at 180 degrees) to the intended movement. For example, if the intended movement is “left” from P, then Q is a blocked position where Q is right of P.

TransProb(s,a)

{

s’ = T(s,a) // T is a function that returns the state s’ when action a is taken from s.

if s’ is a pickup location and s’ is the intended state

{

p = 0.9

}

else if s’ is not a pickup location and is not a blocked position and s’ is the intended state

{

p = 0.8

}

else if s’ is a blocked position

{

p = 0.0

}

else if s’ is not a blocked position and s’ is not the intended state

{

p = (1-0.8\*neighbor\_pickup(s)\*9/8)/(n\_dest(s)-1)

// where neighbor\_pickup is a function that returns 1 if a premium passenger

// pickup is in the neighborhood of s, otherwise returns 0

// ndest\_(s) is another function that returns the number of (non-blocked)

// destinations from state s

}

}

TransProb=p

return

Once the rewards and probabilities are available from any given state with every possible action then we can apply the Value Iteration function discussed in the lectures

ValueIteration(

{

max\_iterations=500 // set value as needed, may need to decrease to 100

U0(s)=0 for all states s

k=0

repeat

{

set rewards as needed

k=k+1

+

// r(s)=-0.5, the live-in reward

} until for all states s