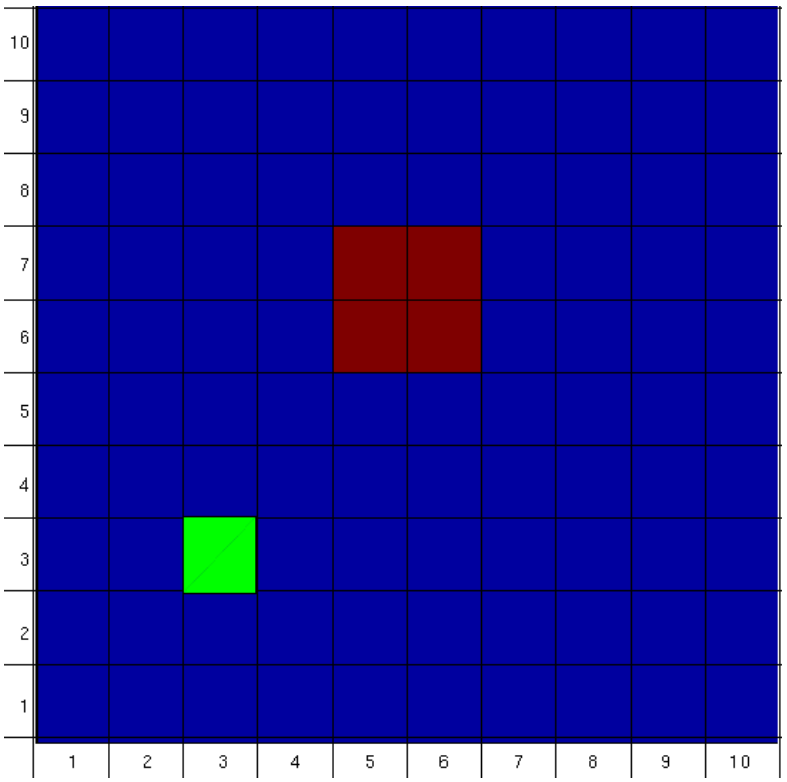
Value Iteration (Stochastic Grid World)

Value Iteration (Stochastic Grid World)

0.0/30.0 points (graded)

In this problem we will consider the optimal control problem for a stochastic version of the "Grid World" domain we examined in lecture. Again, all the Value Iteration code is provided in Drake. Your task will be to setup the transition matrices and cost functions in order to define the stochastic dynamics and optimal control problem.



The stochastic Grid World consists of a $n \times n$ grid (see figure above). We will take $n = 10$ for this example. Here, the cells are listed in (x, y) order. The cell (1,1) corresponds to the bottom left corner, the cell (10,1) corresponds to the bottom right, and the cell (10,10) corresponds to the top right. The actions that the robot can take are "up", "right", "down", and "left". The dynamics in this domain are as follows:

- If the robot is on any cell at the border of the domain and attempts to take a move that would take it out of the domain, it will remain in the same cell with probability 1. If it is on the border and attempts to move in any other direction (i.e., one that will not take it out of the domain), it will move in that direction with probability 1. So, for example, if the robot is in cell (4,10), and it takes the "up" action, it will remain in cell (4,10) with probability 1. If it takes the "right" action, it will move to cell (5,10) with probability 1.

- If the robot is on a cell that is not on the border and attempts to move in a given direction, it will successfully do so with probability p. With probability $(1-p)/3$ it will move in any of the other three directions.

- If the robot is at the goal state, the robot will remain there with probability 1 no matter what action it takes.

The goal is to get to the cell (3,3), while avoiding an obstacle that spans the four cells (5,6), (5,7),(6,6),(6,7) as shown in the picture above. In order to achieve this, we will make our cost function as follows:

- The cost of being in any of the obstacle cells and taking any action from these is 50.

- The cost of being at the goal state and taking any action is 0.

- In any other state, the cost of taking any action is 1.

(a) Take a look at the "properties" in the MarkovDecisionProcess.m file in the drake/systems/ folder. Write Matlab code to compute the transitions T and cost C that are required as inputs to this function, corresponding to the transition dynamics and cost described for the stochastic grid world above. (The states S and actions A are already provided for you below). We will take the probability p described above to be 0.75.

Some helpful tips: (1) You should test out your code on a local machine before copying it below. (2) It might be helpful to test your code out with $p = 1$ first since the solution there is easy to verify (however, be sure to change p back to 0.75 when making the submission!). (3) It might be useful to obtain solutions to the entire problem (i.e., parts (b) and (c) before submitting your code below).

```
% States and actions
n = 10;
N = n^2;
[y,x] = meshgrid(1:n,1:n);
S = [x(:)';y(:)'];
A = [1,2,3,4]; % 1:"up", 2:"right", 3:"down", 4:"left"

% Costs
```

```
C = sparse(N,4);

% Transitions
T{1} = sparse(N,N);
T{2} = sparse(N,N);
T{3} = sparse(N,N);
T{4} = sparse(N,N);
```

Unanswered

```
% States and actions
n = 10;
N = n^2;
% S = 1:N;
[y,x] = meshgrid(1:n,1:n);
S = [x(:)';y(:)'];
A = [1,2,3,4];

% Costs
C = sparse(N,4);
% Obstacles
for i = 1:n
    for j = 1:n
        % Obstacles (cost should be 50)
        if all([i,j] == [5,6]) || all([i,j] == [6,6]) || all([i,j] == [5,7]) || all([i,j] == [6,
            C((j-1)*n+i,:) = 50*ones(1,4);
        else
            % Cell is not an obstacle (cost should be 1)
            C((j-1)*n+i,:) = ones(1,4);
        end
    end
end

% Goal (3,3), cost should be 0
C(2*n+3,:) = zeros(1,4);


% Transition matrices
T{1} = sparse(N,N);
T{2} = sparse(N,N);
T{3} = sparse(N,N);
T{4} = sparse(N,N);

% Probabilities of transitions
p_correct = 0.75;
p_wrong = (1-p_correct)/3;

% Action = 1 (up)
for i = 1:N
    if any(i == [(N-n+1):N])
        % on the top border
        T{1}(i,i) = 1;
    elseif (mod(i,n) == 0)
        % right border
        T{1}(i,i+n) = 1;
    elseif any(i == [1:n])
        % bottom border
```

```
        T{1}(i,i+n) = 1;
    elseif (mod(i,n) == 1)
        % left border
        T{1}(i,i+n) = 1;
    else
    % Typical case (not on the border)
    T{1}(i,i-n) = p_wrong;
    T{1}(i,i+1) = p_wrong;
    T{1}(i,i+n) = p_correct;
    T{1}(i,i-1) = p_wrong;
    end
end

% Action = 2 (right)
for i = 1:N
    if (mod(i,n) == 0)
        % on the right border
        T{2}(i,i) = 1;
    elseif any(i == [(N-n+1):N])
        % top border
        T{2}(i,i+1) = 1;
    elseif any(i == [1:n])
        % bottom border
        T{2}(i,i+1) = 1;
    elseif (mod(i,n) == 1)
        % left border
        T{2}(i,i+1) = 1;
    else
    % Typical case (not on the border)
    T{2}(i,i-n) = p_wrong;
    T{2}(i,i+1) = p_correct;
    T{2}(i,i+n) = p_wrong;
    T{2}(i,i-1) = p_wrong;
    end
end

% Action = 3 (down)
for i = 1:N
    if any(i == [1:n])
        % on the bottom border
        T{3}(i,i) = 1;
    elseif (mod(i,n) == 0)
        % right border
        T{3}(i,i-n) = 1;
    elseif any(i == [(N-n+1):N])
        % top border
        T{3}(i,i-n) = 1;
    elseif (mod(i,n) == 1)
```

```
        % left border
        T{3}(i,i-n) = 1;
    else
    % Typical case (not on the border)
    T{3}(i,i-n) = p_correct;
    T{3}(i,i+1) = p_wrong;
    T{3}(i,i+n) = p_wrong;
    T{3}(i,i-1) = p_wrong;
    end
end

% Action = 4 (left)
for i = 1:N
    if (mod(i,n) == 1)
        % on the left border
        T{4}(i,i) = 1;
    elseif (mod(i,n) == 0)
        % right border
        T{4}(i,i-1) = 1;
    elseif any(i == [1:n])
        % bottom border
        T{4}(i,i-1) = 1;
    elseif any(i == [(N-n+1):N])
        % top border
        T{4}(i,i-1) = 1;
    else
    % Typical case (not on the border)
    T{4}(i,i-n) = p_wrong;
    T{4}(i,i+1) = p_wrong;
    T{4}(i,i+n) = p_wrong;
    T{4}(i,i-1) = p_correct;
    end
end

% Once you're at the goal, you stay there
for i = 1:N
    T{1}(2*n+3,i) = 0;
    T{2}(2*n+3,i) = 0;
    T{3}(2*n+3,i) = 0;
    T{4}(2*n+3,i) = 0;
end

T{1}(2*n+3,2*n+3) = 1;
T{2}(2*n+3,2*n+3) = 1;
T{3}(2*n+3,2*n+3) = 1;
T{4}(2*n+3,2*n+3) = 1;
```

Run Code

(b) Download the file StochasticGridWorld.m from here. This file calls the Value Iteration code in Drake and sets up the visualization for the stochastic grid world. You will not need to modify this file. All you need to do is create a function stochasticSATC() that outputs S,A,T, and C as computed in part (a) above:

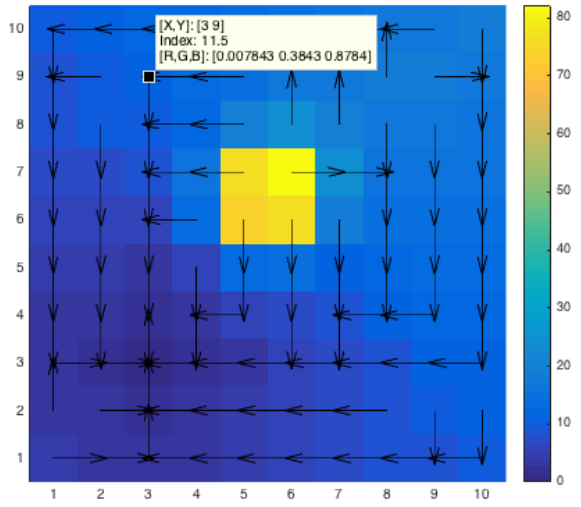
[S,A,T,C] = stochasticSATC()

Make sure to save the stochasticSATC.m file in the same folder as StochasticGridWorld.m. Now run the runValueIteration function in StochasticGridWorld.m. This will run value iteration using the transition dynamics and cost function you defined.

What is the optimal cost-to-go returned by value iteration at the cell (3,9)?

Answer: 11.5

Explanation

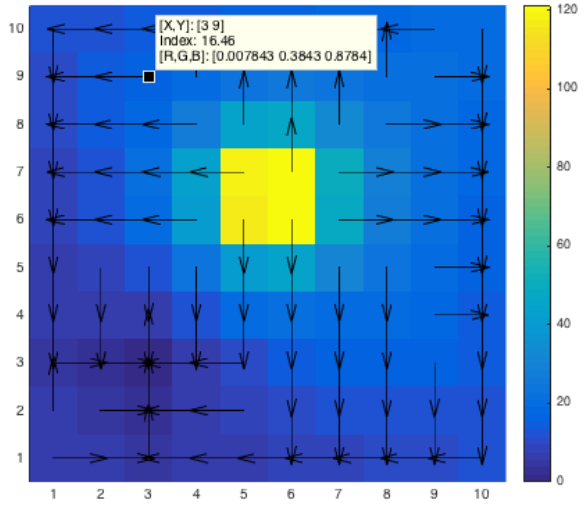


This can be read off from the plot generated by the code (the "index" field). The cost-to-go is approximately 11.5.

(c) Change the probability p to 0.5. Now what is the optimal cost-to-go returned by value iteration at the cell (3,9)?

Answer: 16.46

Explanation



This can be read off from the plot generated by the code (the "index" field). The cost-to-go is approximately 16.46.

Submit You have used 0 of 3 attempts

Answers are displayed within the problem