```
% Achyuth Nandikotkur
% V00975928
% ECE-559B
% November 8th, 2021

% Question 2

clear;
clc;

global returns qvector searchrewards waitrewards startState
 actionsAtHigh actionsAtLow stepsize epsilon;

% To store average returns
% high - low
qvaluehighsearch = [0];
qvaluehighwait = [0];

qvaluelowsearch = [0];
qvaluelowwait = [0];
qvaluelowrecharge = [0];

Steps = 5000;

stepsize = 0.05;
epsilon = 0.1;

searchrewards = [3, 4, 5, 6];
waitrewards = [0, 1, 2];

pisearchhigh = 0.5;
piwaithigh = 0.5;
pisearchlow = 0.5;
piwaitlow = 0.25;
pirechargelow = 0.25;



loop = 1;
counterhigh = 0;
counterlow = 0;
% returns{0} high
% returns{1} low
returns = [0 0];

for outerloop = 1: Steps
    % selecting initial state as high = 1 or low = 2 with equal
 probability
    sequence = cell(1, 2);

    % Since SARSA is single step, generating a single step episode
    for k1 = 1:2
```

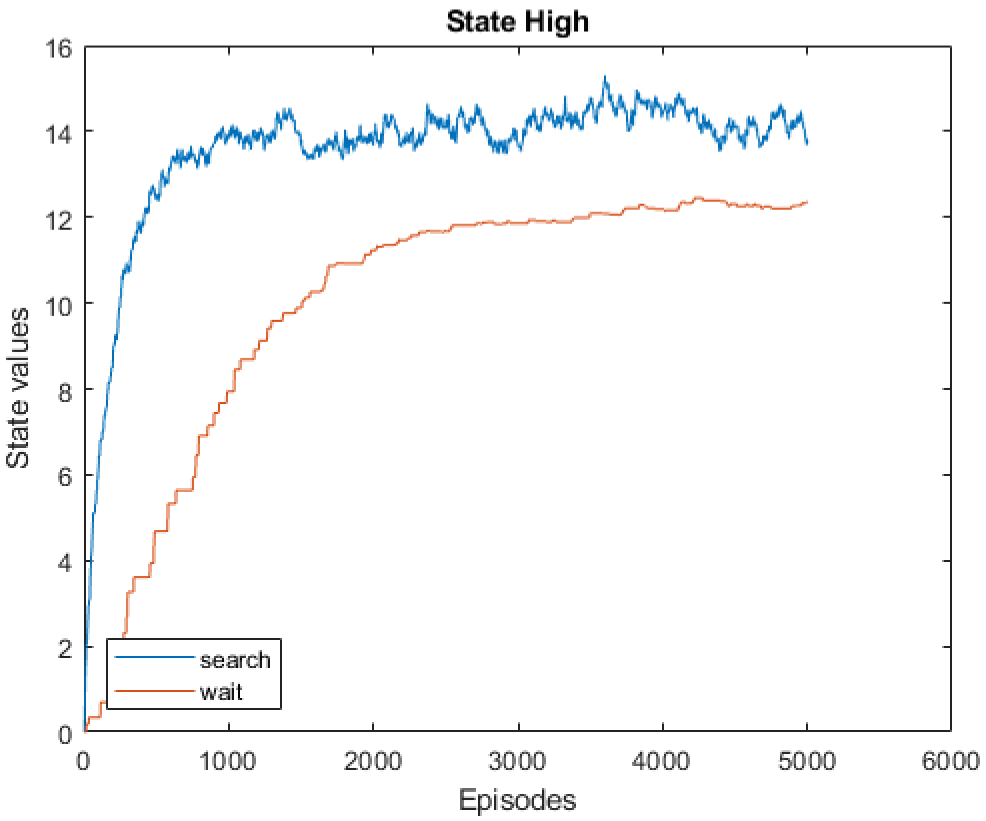
```
           sequence{k1}.state = 0;
           sequence{k1}.action = 0;
           sequence{k1}.reward = 0;
   end

   % Selecting initial state randomly
   sequence{1}.state = randsample([1, 2], 1, true, [0.5, 0.5]);

   % Selecting initial action
   epsgreedy = rand;

   % Greedily
   if(epsgreedy <= (1 - epsilon))
       % If in state high
       if(sequence{1}.state == 1)
           % choosing greedily
           [maxValuedActions, I] = max([qvaluehighsearch(end),
qvaluehighwait(end)]);

           % Tie breaking between different same max valued actions
           sameValueActions = find([qvaluehighsearch(end),
qvaluehighwait(end)] == maxValuedActions);
           r = randi(length(sameValueActions));
           sequence{1}.action = sameValueActions(r);
       else
           % choosing greedily
           [maxValuedActions, I] = max([qvaluelowsearch(end),
qvaluelowwait(end), qvaluelowrecharge(end)]);

           % Tie breaking between different same max valued actions
           sameValueActions = find([qvaluelowsearch(end),
qvaluelowwait(end), qvaluelowrecharge(end)] == maxValuedActions);
           r = randi(length(sameValueActions));
           sequence{1}.action = sameValueActions(r);
       end
   else
       % Randomly with epsilon probability

       if(sequence{1}.state == 1)
           % if initial state is high, select search or wait randomly
           sequence{1}.action = randsample([1, 2], 1);
       else
           % if initial state is low, select search or wait or
recharge randomly
           sequence{1}.action = randsample([1, 2, 3], 1);
       end
   end


   % Determining reward and next state
   if(sequence{1}.state == 1)
       % action can be search = 1, wait = 2;
       if sequence{1}.action == 1
```

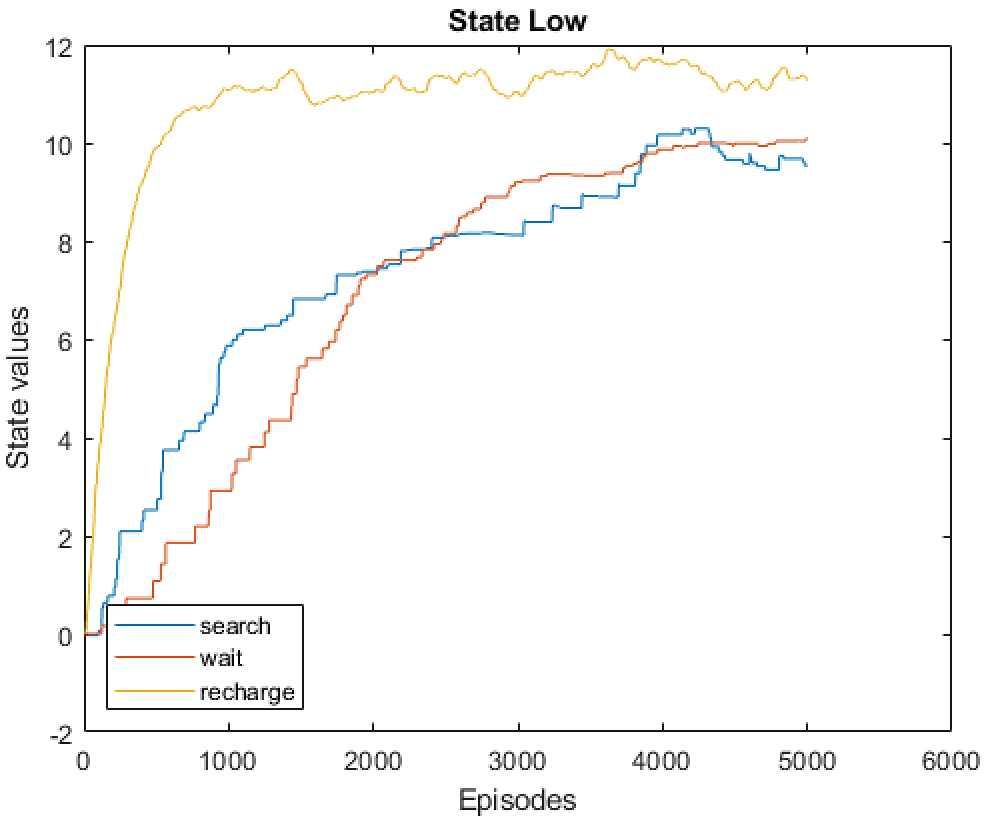
```
          sequence{1+1}.state = randsample([1, 2],1, true, [0.25,
 0.75]);
          sequence{1}.reward =  randsample(searchrewards,1, true,
 [1/4, 1/4, 1/4, 1/4]);
        else
          sequence{1}.reward =  randsample(waitrewards,1, true, [1/3,
 1/3, 1/3]);
          sequence{1+1}.state = 1;
        end
    else
        % if in state low
        % action can be search = 1, wait = 2; recharge = 3;
        if sequence{1}.action == 1
          sequence{1+1}.state = randsample([2, 1], 1, true, [0.25,
 0.75]);
          if(sequence{1+1}.state == 2)
              sequence{1}.reward =  randsample(searchrewards,1, true,
 [1/4, 1/4, 1/4, 1/4]);
          else
              sequence{1}.reward = -3;
          end
        elseif(sequence{1}.action == 2)
          sequence{1}.reward =  randsample(waitrewards,1, true, [1/3,
 1/3, 1/3]);
          sequence{1+1}.state = 2;
        else
           sequence{1}.reward = 0;
           sequence{1+1}.state = 1;
        end
    end

% MAX SARSA - CHOOSING NEXT ACTION WITH EPISLON GREEDY
    if(sequence{1+1}.state == 1)
        % choosing greedily
        [maxValuedActions, I] = max([qvaluehighsearch(end),
 qvaluehighwait(end)]);

        % Tie breaking between different same max valued actions
        sameValueActions = find([qvaluehighsearch(end),
 qvaluehighwait(end)] == maxValuedActions);
    else
        % choosing greedily
        [maxValuedActions, I] = max([qvaluelowsearch(end),
 qvaluelowwait(end), qvaluelowrecharge(end)]);

        % Tie breaking between different same max valued actions
        sameValueActions = find([qvaluelowsearch(end),
 qvaluelowwait(end), qvaluelowrecharge(end)] == maxValuedActions);
    end

    r = randi(length(sameValueActions));

    sequence{1+1}.action = sameValueActions(r);
```

```
% MAX SARSA - CHOOSING NEXT ACTION MAX
    if(sequence{1}.state == 1)
        % Choosing to search now
        if(sequence{1}.action == 1)
            if(sequence{2}.state == 1)
                % AT HIGH - can search and wait
                if(sequence{2}.action == 1)
                    temp = qvaluehighsearch(end) + stepsize
 * (sequence{1}.reward + 0.8 * (qvaluehighsearch(end)) -
 qvaluehighsearch(end));
                else
                    temp = qvaluehighsearch(end) + stepsize
 * (sequence{1}.reward + 0.8 * (qvaluehighwait(end)) -
 qvaluehighsearch(end));
                end
            else
                % AT LOW - can search, wait and recharge
                if(sequence{2}.action == 1)
                    temp = qvaluehighsearch(end) + stepsize
 * (sequence{1}.reward + 0.8 * (qvaluelowsearch(end)) -
 qvaluehighsearch(end));
                elseif(sequence{2}.action == 2)
                    temp = qvaluehighsearch(end) + stepsize
 * (sequence{1}.reward + 0.8 * (qvaluelowwait(end)) -
 qvaluehighsearch(end));
                else
                    temp = qvaluehighsearch(end) + stepsize
 * (sequence{1}.reward + 0.8 * (qvaluelowrecharge(end)) -
 qvaluehighsearch(end));
                end
            end
            qvaluehighsearch = [qvaluehighsearch; temp];
            qvaluehighwait = [qvaluehighwait; qvaluehighwait(end)];
        % Choosing to wait now
        elseif(sequence{1}.action == 2)
            % AT HIGH - can search and wait
            if(sequence{2}.action == 1)
                temp = qvaluehighwait(end) + stepsize *
 (sequence{1}.reward + 0.8 * (qvaluehighsearch(end)) -
 qvaluehighwait(end));
            else
                temp = qvaluehighwait(end) + stepsize
 * (sequence{1}.reward + 0.8 * (qvaluehighwait(end)) -
 qvaluehighwait(end));
            end
             qvaluehighwait = [qvaluehighwait; temp];
             qvaluehighsearch = [qvaluehighsearch;
 qvaluehighsearch(end)];
        end

        qvaluelowsearch = [qvaluelowsearch; qvaluelowsearch(end)];
        qvaluelowwait = [qvaluelowwait; qvaluelowwait(end)];
        qvaluelowrecharge = [qvaluelowrecharge;
 qvaluelowrecharge(end)];
```

```
   else
       % Choosing to search at LOW.
       if(sequence{1}.action == 1)
           if(sequence{2}.state == 1)
               % AT HIGH - can search and wait
               if(sequence{2}.action == 1)
                   temp = qvaluelowsearch(end) + stepsize
* (sequence{1}.reward + 0.8 * (qvaluehighsearch(end)) -
qvaluelowsearch(end));
               else
                   temp = qvaluelowsearch(end) + stepsize
* (sequence{1}.reward + 0.8 * (qvaluehighwait(end)) -
qvaluelowsearch(end));
               end
           else
               % AT LOW - can search, wait and recharge
               if(sequence{2}.action == 1)
                   temp = qvaluelowsearch(end) + stepsize
* (sequence{1}.reward + 0.8 * (qvaluelowsearch(end)) -
qvaluelowsearch(end));
               elseif(sequence{2}.action == 2)
                   temp = qvaluelowsearch(end) + stepsize
* (sequence{1}.reward + 0.8 * (qvaluelowwait(end)) -
qvaluelowsearch(end));
               else
                   temp = qvaluelowsearch(end) + stepsize
* (sequence{1}.reward + 0.8 * (qvaluelowrecharge(end)) -
qvaluelowsearch(end));
               end
           end
           qvaluelowsearch = [qvaluelowsearch; temp];
           qvaluelowwait = [qvaluelowwait; qvaluelowwait(end)];
           qvaluelowrecharge = [qvaluelowrecharge;
qvaluelowrecharge(end)];

       % Choosing to wait
       elseif(sequence{1}.action == 2)
           % LOW - WAIT - LOW
           % AT LOW - can search, wait and recharge
           if(sequence{2}.action == 1)
               temp = qvaluelowwait(end) + stepsize *
(sequence{1}.reward + 0.8 * (qvaluelowsearch(end)) -
qvaluelowwait(end));
           elseif(sequence{2}.action == 2)
               temp = qvaluelowwait(end) + stepsize
* (sequence{1}.reward + 0.8 * (qvaluelowwait(end)) -
qvaluelowwait(end));
           else
               temp = qvaluelowwait(end) + stepsize *
(sequence{1}.reward + 0.8 * (qvaluelowrecharge(end)) -
qvaluelowwait(end));
           end
           qvaluelowwait = [qvaluelowwait; temp];
           qvaluelowsearch = [qvaluelowsearch; qvaluelowsearch(end)];
```

```
            qvaluelowrecharge = [qvaluelowrecharge;
 qvaluelowrecharge(end)];

        % Choosing to recharge at low
        else
            % LOW - RECHARGE - HIGH
            if(sequence{2}.action == 1)
                temp = qvaluelowrecharge(end) + stepsize
 * (sequence{1}.reward + 0.8 * (qvaluehighsearch(end)) -
 qvaluelowrecharge(end));
            else
                temp = qvaluelowrecharge(end) + stepsize
 * (sequence{1}.reward + 0.8 * (qvaluehighwait(end)) -
 qvaluelowrecharge(end));
            end
            qvaluelowrecharge = [qvaluelowrecharge; temp];
            qvaluelowsearch = [qvaluelowsearch; qvaluelowsearch(end)];
            qvaluelowwait = [qvaluelowwait; qvaluelowwait(end)];
        end
        qvaluehighsearch = [qvaluehighsearch; qvaluehighsearch(end)];
        qvaluehighwait = [qvaluehighwait; qvaluehighwait(end)];
    end
end
% celldisp(sequence);

t1=1:length(qvaluehighsearch);
t2=1:length(qvaluehighwait);
t3=1:length(qvaluelowsearch);
t4=1:length(qvaluelowwait);
t5=1:length(qvaluelowrecharge);

figure(1)
plot(t1, qvaluehighsearch, t2,qvaluehighwait);
xlabel('Episodes')
ylabel('State values')
legend({'search','wait'},'Location','southwest')
title('State High');

figure(2)
plot(1:length(qvaluelowsearch), qvaluelowsearch,
 1:length(qvaluelowwait), qvaluelowwait, 1:length(qvaluelowrecharge),
 qvaluelowrecharge);
xlabel('Episodes')
ylabel('State values')
legend({'search','wait', 'recharge'},'Location','southwest')
title('State Low');
```


State High
State values
Episodes
16
14
12
10
8
6
4
2
0
0
1000
2000
3000
4000
5000
6000
search
wait


State Low
State values
Episodes
12
10
8
6
4
2
0
-2
0
1000
2000
3000
4000
5000
6000
search
wait
recharge

Published with MATLAB® R2021a