30 Dynamic Programming for Path Design

Given the transition costs in red, what are the maximum and minimum costs to get from node 1 to node 11? This situation is encountered when planning paths for autonomous agents moving through a complex environment, e.g., a wheeled robot in a building.

Solution: The minimum cost is 16 (path [1,6,9,11] or [1,2,8,9,11]) and the maximum value is 28 (path [1,4,5,6,7,9,11]!). The attached code uses value iteration to find these in two and five iterations, respectively.

```matlab
clear all;

ch = input('Find minimum (0) or maximum (1): '); % Value iteration solution of deterministic dynamic programming.
if ~ch,
    init = 1e6; % look for minimum; % The program looks complicated only because I cover both
    % big initial guesses for costs to go % minimization and maximization in the same program!
else,
    init = 1e-6; % look for maximum; % small initial guesses for value to go
end;

% interconnect matrix: row is the node (first is starting
% point) and column is the set of nodes pointed to. Note
% that the ending node is not included because it points to
% nowhere.
I = [[2 6 4]  % node 1 (start) points to nodes 2,6,4
     [3 8 5]  % node 2 points to nodes 3,8,5. And so on...
     [8 6 NaN] % node 3
     [5 7 NaN] % node 4
     [6 7 10] % node 5
     [8 9 7] % node 6
     [10 9 NaN] % node 7
     [10 9 NaN] % node 8
     [11 NaN NaN] % node 9
     [11 NaN NaN]]; % node 10

% cost per link - these go with the interconnects in A. Note
% that the entries with direct connection to the end node are NaN,
% because we will enforce the link cost in ctg (below) explicitly
C = [[3 7 5]  % The cost is 3 to move between nodes 1 and 2,
      [2 5 4]  % and 7 to move between nodes 1 and 6, etc.
      [4 5 NaN] % node 2
      [3 5 NaN] % node 3
      [4 4 7] % node 4
      [4 5 4] % node 5
      [4 8 NaN] % node 6
      [8 4 NaN] % node 7
      [NaN NaN NaN] % node 8
      [NaN NaN NaN]]; % node 10

% initial guess of cost-to-go (or value-to-go) at each node
tg = [[[NaN]
       [init]  % node 1
       [init]  % node 2
       [init]  % node 3
       [init]  % node 4
       [init]  % node 5
       [init]  % node 6
       [init]  % node 7
       [init]  % node 8
       [4]  % node 9 (points directly to end, node 11)
       [3]]]; % node 10 (points directly to end, node 11)

w = size(I,2); % width of interconnect matrix

disp(sprintf('%g ',tg));  % list the first cost-to-go or
% value-to-go

for k = 1:5,  % carry out a fixed number of iterations

  % cycle through the nodes one by one. Note that we don't
  % need to recompute tg for nodes that point to the end
  for i = 1:sum(~isnan(C(:,1))),

    % We'll look for the minimum estimated cost-to-go
    % (or maximum estimated value-to-go) across
    % the possible nodes pointed to
    if ~ch,
      dummy = 1e6;  % initialize to be huge
    else,
      dummy = 1e-6;  % initialize to be tiny
    end;

    % look at all the nodes pointed to from node i
    for j = 1:w,
      if ~isnan(I(i,j)), % consider only true entries in I

        test = tg(I(i,j)) + C(i,j) ;

        if ~ch,  % look for minimum
          if test < dummy,
            dummy = test ;
          end;
        else,  % look for maximum
          if test > dummy,
            dummy = test ;
          end;
        end;

      end;  % "true entries"
    end;  % j: nodes pointed to
  end;  % i: nodes
  tg(i) = dummy ;
end;  % i: nodes

disp(sprintf('%g',tg));

end;  % k: iteration