Lecture 25: Dynamic Programming: Matlab Code

University of Southern California

Linguistics 285

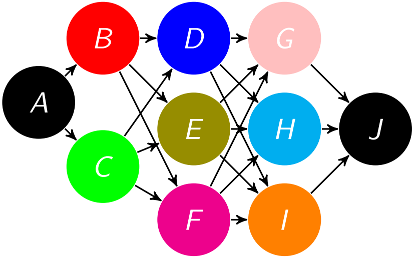
USC Linguistics

December 1, 2015

Dynamic Programming Approach

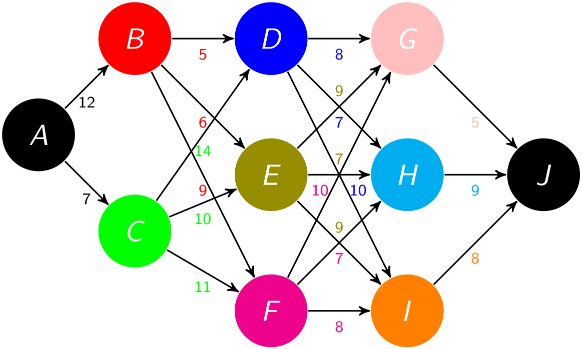
- Dynamic Programming is an alternative search strategy that is faster than Exhaustive search, slower than Greedy search, but gives the optimal solution.
- View a problem as consisting of subproblems:
  - Aim: Solve main problem
  - To achieve that aim, you need to solve some subproblems
  - To achieve the solution to these subproblems, you need to solve a set of subsubproblems
  - And so on...
- Dynamic Programming works when the subproblems have similar forms, and when the tiniest subproblems have very easy solutions.

Dynamic Programming



- Main Problem: Shortest path from A to J: V_{AJ}
- DP Thinking: Let's say I know the best paths from B to J and C to J: V_{BJ} and V_{CJ}. Then we would add V_{BJ} to the cost from A to B, V_{CJ} to the cost from A to C, compare the two, and pick the least.
- So we now have 2 subproblems V_{BJ} and V_{CJ}. If we could solve those subproblems, we could solve the main problem V_{AJ}.
- But now we can think of V_{BJ} as its own problem, and then repeat the thinking: If I knew the solutions to V_{DJ}, V_{EJ}, and V_{FJ}, then we can solve V_{BJ}. Same with V_{CJ}.
- Continue thinking in this way till we get to: V_{GJ}, V_{HJ}, V_{IJ}, which are easy to solve!

Dynamic Programming



$$V_{AJ} = min \begin{Bmatrix} 12 + V_{BJ} \\ 7 + V_{CJ} \end{Bmatrix}$$

$$V_{BJ} = min \begin{Bmatrix} 5 + V_{DJ} \\ 6 + V_{EJ} \\ 9 + V_{FJ} \end{Bmatrix}$$

$$V_{CJ} = min \begin{Bmatrix} 14 + V_{DJ} \\ 10 + V_{EJ} \\ 11 + V_{FJ} \end{Bmatrix}$$

$$V_{DJ} = min \begin{Bmatrix} 8 + V_{GJ} \\ 7 + V_{HJ} \\ 10 + V_{IJ} \end{Bmatrix}$$

$$V_{EJ} = min \begin{Bmatrix} 9 + V_{GJ} \\ 7 + V_{HJ} \\ 9 + V_{IJ} \end{Bmatrix}$$

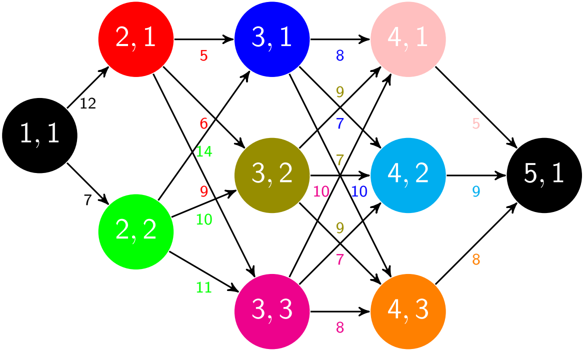
$$V_{FJ} = min \begin{Bmatrix} 10 + V_{GJ} \\ 7 + V_{HJ} \\ 8 + V_{IJ} \end{Bmatrix}$$

$$V_{GJ} = 5$$

$$V_{HJ} = 9$$

$$V_{IJ} = 8$$

Dynamic Programming solution in Matlab



$$V_{DJ} = min \begin{Bmatrix} 8 + V_{GJ} \\ 7 + V_{HJ} \\ 10 + V_{IJ} \end{Bmatrix}$$

$$V_{BJ} = min \begin{Bmatrix} 5 + V_{DJ} \\ 6 + V_{EJ} \\ 9 + V_{FJ} \end{Bmatrix}$$

$$V_{GJ} = 5$$

$$V_{AJ} = min \begin{Bmatrix} 12 + V_{BJ} \\ 7 + V_{CJ} \end{Bmatrix}$$

$$V_{EJ} = min \begin{Bmatrix} 9 + V_{GJ} \\ 7 + V_{HJ} \\ 9 + V_{IJ} \end{Bmatrix}$$

$$V_{HJ} = 9$$

$$V_{CJ} = min \begin{Bmatrix} 14 + V_{DJ} \\ 10 + V_{EJ} \\ 11 + V_{FJ} \end{Bmatrix}$$

$$V_{IJ} = 8$$

$$V_{FJ} = min \begin{Bmatrix} 10 + V_{GJ} \\ 7 + V_{HJ} \\ 8 + V_{IJ} \end{Bmatrix}$$

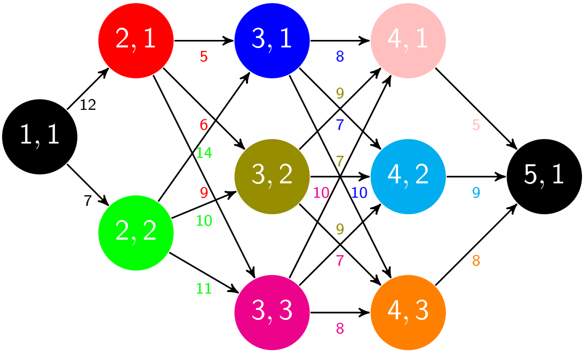
Dynamic Programming notation: distances

- cost (or distance) of going from stage 1, state 1 to stage 2, state 1
- $d(1, 1, 2, 1)$
- cost (or distance) of going from stage 1, state 1 to stage 2, state 2
- $d(1, 1, 2, 2)$
- cost (or distance) of going from stage 2, state 1 to stage 3, state 2
- $d(2, 1, 3, 2)$
- cost (or distance) of going from stage k, state i to stage k+*1*, state j
- $d(k, i, k+1, j)$

Dynamic Programming notation: Minimum cost from a node to the end

- minimum cost of going from stage 4, state 1 to end
- $V(4,1)$
- minimum cost of going from stage 3, state 2 to end
- $V(3,2)$
- Evaluating:
- $V(4,1) = 5$
- $V(3,2) = min \begin{Bmatrix} d(3,2,4,1) + V(4,1) \\ d(3,2,4,2) + V(4,2) \\ d(3,2,4,3) + V(4,3) \end{Bmatrix}$
- Generalizing:
- $V(k,i) = min_j(d(k,i,k+1,j) + V(k+1,j))$

Matlab code for DP: Defining the network




```
costs(1,1,2,1) = 12;
costs(1,1,2,2) = 7;
costs(2,1,3,1) = 5;
costs(2,1,3,2) = 6;
costs(2,1,3,3) = 9;
costs(2,2,3,1) = 14;
costs(2,2,3,2) = 10;
costs(2,2,3,3) = 11;
costs(3,1,4,1) = 8;
costs(3,1,4,2) = 7;
costs(3,1,4,3) = 10;
costs(3,2,4,1) = 9;
costs(3,2,4,2) = 7;
costs(3,2,4,3) = 9;
costs(3,3,4,1) = 10;
costs(3,3,4,2) = 7;
costs(3,3,4,3) = 8;
costs(4,1,5,1) = 5;
costs(4,2,5,1) = 9;
costs(4,3,5,1) = 8;
num_states = [1 2 3 3 1];
```

Matlab code for DP

- Using this generalized form, we can write a Matlab program, using nested loops, that will start at the end and compute $V(k, i)$ for every node recursively.
- The last one we compute will be $V(1, 1)$ which is the length of the minimum path from beginning to end.

```
d = costs;
V(5,1)=0;
for k=4:-1:1
    for i=1:num_states(k)
        for j = 1:num_states(k+1)
          path_length(j)= d(k,i,k+1,j)+V(k+1,j);
        end
        V(k,i)=min(path_length);
        clear path_length
    end
end
V(1,1)
```

Matlab code for DP

- Why is this incomplete?
- We know the minimum length path, but we don't know which states it passes through.
- Now start at the begining. Add the cost of going from stage k to each of the nodes at stage $k+1$.
- Find which total is minimal and choose the corresponding state in stage $k+1$.

```
path = 0;
index = 1;
for k=1:4
    for j=1:num_states(k+1)
         path_length(j)= d(k,index,k+1,j)+V(k+1,j);
    end
    [minval, index] = min(path_length);
    path(k) = index;
    clear path_length
end
path
```