CSCI-UA 472 Artificial Intelligence

Muhammad Wajahat Mirza

mwm356@nyu.edu

Homework 01

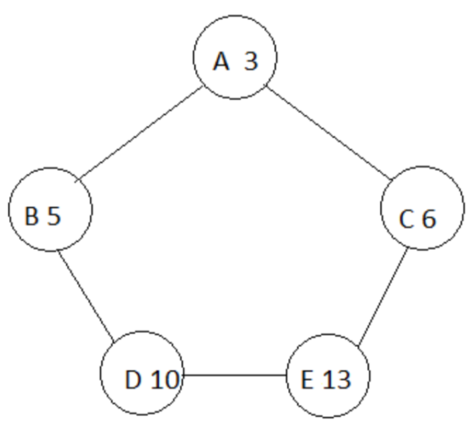
September 15, 2020

Problem Overview

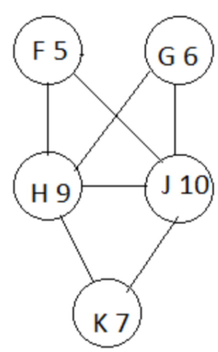
The **"Best Vertex Cover"** problem is defined as follows:
Input : An undirected graph **G**, in which each vertex is marked by a positive cost; and a total budget **T**.

Goal: A set of vertices S such that **(a)** every edge in G has at least one end in S and **(b)** the total cost of the vertices in S is at most B.

For instance, in Graph 1 below, with **T** = 20, a solution is {A,C,D}. In Graph 2, with **T** = 20, a solution is {H,J}



Graph 1



Graph 2

Problem 1

Consider the following state space for systematically solving the problem:

State: A set of vertices whose total cost is at most **T**.
Successor to state S: Add some vertex **V** to **S** such that:
(a) V is alphabetically later than the vertices in **S** (to guarantee systematic search);
(b) V covers some edge that is uncovered in **S** (otherwise it's pointless);
(c) the total cost is no greater than **T**.

Start state: The empty set.
Goal state: A state that covers all the edges.

Show the part of the state space that would be generated if you use **depth-first search (DFS)** in this state space to find the solution to **graph 2** with **T** = 23.
You should show this as a tee as shown below:

```
Root--->C1--->C1.1--->C1.1.1
        |     |       |
        |     |       |->C1.1.2
        |     |
        |     |->C1.2--->C1.2.1
        |
        |->C2--->C2.1
```

Figure 1: left-to-right typewriter format

Solution to Problem 1

In order to find the optimal solution that satisfies all three given conditions: **(a) V** is alphabetically later than the vertices in **S**; **(b) V** covers some edge that is uncovered in **S**; **(c)** the total cost is no greater than **T** where **T** = 23, we do our Depth-first Search (DFS) from Root starting with **F5**. Once DFS on **F5** is complete, we progress to **G6**, **H9** and so on until we find the solution. When we find solution that satisfies all three above mentioned conditions, we terminate DFS and state our solution.

```
Problem 1:

{Root} --> {F5} --> {F5,G6} --->{F5,G6, H9} ==> 'Adding {J10} violates cost T condition'
        ||      |           |
        ||      |           |--->{F5,G6,J10} ==> 'Adding {K7} violates cost T condition
        ||      |           |
        ||      |           |--->{F5,G6, K7} ==> 'Undiscovered edge H-J'
        ||      |
        ||      |-->{F5, H9} --->{F5,H9,K9} ==> 'Undiscovered edge G-J'
        ||      |
        ||      |-->{F5,J10}--->{F5,J10,K9} ==> 'Undiscovered edge G-H'
        ||      |
        ||      |-->{F5, K7}                ==> 'Adding any other vertex violates condition (a)'
        ||
        ||
        ||->{G6} --->{G6,H9}--->{G6,H9, K7} ==> 'Undiscovered edge F-J'
        |       |           |
        |       |           |--->{G6,J10,K7} ==> 'Undiscovered edge F-H'
        |       |
        |       |-->{G6,K7}                 ==> 'Adding any other vertex violates condition (a)'
        |
        |
        |->{H9}--->{H9,J10}                 ==> 'Solution Found! Meets All conditions i.e.'
                                                1. Systemic Search
                                                2. Covers all uncovered/undiscovered edges
                                                3. Cost < T i.e. T < 23
```

Figure 2: left-to-right typewriter format for DFS

So, solution to our problem using DFS is **{H9, J10}**.

Problem 2

Show the states generated in doing a **breadth-first search** of the state space in problem 1.

Solution to Problem 2

In order to find the optimal solution that satisfies all three given conditions: **(a) V** is alphabetically later than the vertices in **S**; **(b) V** covers some edge that is uncovered in **S**; **(c)** the total cost is no greater than **T** where **T** $= 23$, we do our Breadth-first Search (BFS) from Root starting with the shallowest depth. When we find solution that satisfies all three above mentioned conditions, we terminate BFS and state our solution.

```
Problem 2:

{At} --> {Depth 01} --> {F5} -- {G6} -- {H9} -- {J10} -- {K7}
         |
         |{Depth 02} --> {F5, G6} -- {F5, H9} -- {F5,J10} -- {F5, K7} ==> 'Undiscovered edges at each state'
         |
         |{Depth 02} --> {G6, H9} -- {G6,J10} -- {G6, K7}             ==> 'Undiscovered edges at each state'
         |
         |{Depth 02} --> {H9, J10}                ==> 'Solution found! Meets all requirements i.e.'
                                                      1. Systemic Search
                                                      2. Covers all uncovered/undiscovered edges
                                                      3. Cost < T i.e. T < 23
```

Figure 3: left-to-right typewriter format for BFS

So, solution to our problem using BFS is also **{H9, J10}**.

Problem 3

A. Construct an instance of **Best Vertex Cover** where doing a **breadth-first search** will construct 1000 (or more) times as many states as doing a **depth-first search**.

B. Construct an instance of **Best Vertex Cover** where doing a **depth-first search** will construct 1000 (or more) times as many states as doing a **breadth-first search**. (Hint: There is an instance where the correct solution contains only one vertex.)

Solution to Problem 3 A

- Structure details for the Best Cover Vertex in this case:
  - We have n vertices.
  - Apart from 1^{st} and n^{th} vertices, each vertex has two edges that connect them to their parent and child vertices respectively.
  - 1^{st} vertex edge will connect it to its child vertex while n^{th} vertex edge will connect it to its parent vertex.
  - Say our goal state or solution lies at depth $n-1$.
- DFS will be an optimal solution because:
  - DFS will explore every single child of each vertex until it reaches the depth $n-1$.
  - Hence, in this graph, DFS can reach the solution in $n-1$ states.
- BFS will not be an optimal solution because:
  - Firstly, BFS will explore all possible states at each depth.
  - With BFS, solution's depth cannot be any smaller than $[n/2]$.
  - Therefore, BFS will create $\binom{n}{j}$ number of states at each depth.

- We sum all the states at each depth.

- Hence, as directed in the question, we now have an equation:

$$1000(n-1) < \Sigma_{j=1}^{[n/2]-1} \binom{n}{j}$$

- Therefore, for $n \geq 16$ vertices in the graph, BFS will construct 1000 or more no. of states than DFS before reaching the goal state.

Solution to Problem 3 B

- Structure details for the Best Cover Vertex in this case:
  - Say we have n vertices.
  - Our goal vertex, say V_1 is at the center.
  - Each of other $n-1$ vertices will be connected to V_1 with an edge (like a star).
  - These $n-1$ vertices will only have one edge that is connected to V_1.
  - We now have our graph in a star shape, with one vertex, V_1 at center and every other vertex branching out from V_1.
- Performing BFS on this graph will be very fast because:
  - Maximum depth will be achieved by BFS at D = 1.
  - After visiting $n-1$ states (other vertices at depth = 1), goal state is reached.
- Performing DFS on this graph will be slow and creates more states because:
  - In a systemic search, DFS will first explore all the possible sets of states before reaching the goal state.
  - Thus, in this graph, it will create $2^{(n-1)}$ number of state sets.
- Hence, as directed in the question, we now have an equation:

$$1000(n-1) < 2^{(n-1)}$$

- Therefore, for $n \geq 15$ vertices in the graph, DFS will construct 1000 or more no. of states than BFS before reaching the goal state, V_1.

Problem 4

Consider the Best Vertex Cover on a graph in general with n vertices (not on the specific examples above).
What is the maximal depth D of the state space? What is the branching factor B? Give a bound on the number of states in the state space (you should be able to get a bound much smaller than B^D.)

Solution to Problem 4

- Maximal Depth D = n.
  - (D = n) if all vertices are in the State and/that cost < T.
- Branching factor B = n.
  - (B = n) if the null root vertex branch out (expand directly) to each of the n vertices in the state space.
- Bound on no. of states = 2^n which is much smaller than B^D.
  - Rather than vertices themselves, in this state space, we have sets of vertices. Thus, we have an upper bound of 2^n on the number of possible states as opposed to B^D where vertices are dealt individually (and not in sets).

Problem 5

Consider trying to solve Best Vertex Cover using hill climbing in the following state space.
State: Any set of vertices.
Neighbors of state S: Either add one vertex to **S** or delete one vertex from **S**.
Error function:
  Max(0,(Total cost of the vertices in S)-T) +
  the sum of the costs of all the uncovered edges E,
    where the cost of an edge is considered to be the cost of its cheaper end.

For example, in graph 1, suppose **S** = { D,E } and **T** = 20. **Cost(D)** = 10. **Cost(E)** = 13. The uncovered edges are A-B and A-C, both of cost 3. So **Error(S)** = Max(0,10 + 13 - 20)+3+3 = 9.

What is the sequence of states encountered doing simple hill-climbing in this space, starting from state { D,E }?

Solution to Problem 5

Key to understanding the following answer table:

- Neighbor(+/-) = Which neighbor is being added or deleted.
- State = Current sequence of States.
- Error Cost = Error(S) computed from above equation.
- Action = If curr error < prev Err, then add. Else Skip.
- Progress Bar = Shows the sequences in which states are added (in decreasing order of error cost)

Problem 5:

Error Function: Max(0,(Total cost in S)-T) + cost sum of uncovered edges E (Cheapest end).

Start State: { D,E }

Progress Bar: { D,E }

Neighbor(+/-)	State	Error Cost	Action
--	{D,E}	Max(0,23-20)+3+3 = 09	--
{B}(+)	{D,E,B}	Max(0,28-20)+3 = 11	Skip
{C}(+)	{D,E,C}	Max(0,29-20)+3 = 12	Skip
{A}(+)	{D,E,A}	Max(0,26-20) = 06	<u>Add</u>

Progress Bar: { D,E } --> {D,E,A}

Neighbor(+/-)	State	Error Cost	Action
--	{D,E}	Max(0,23-20)+3+3 = 09	--
{B}(+)	{D,E,B}	Max(0,28-20)+3 = 11	Skip
{C}(+)	{D,E,C}	Max(0,29-20)+3 = 12	Skip
{A}(+)	{D,E,A}	Max(0,26-20)+0 = 06	<u>Add</u>
{B}(+)	{D,E,A,B}	Max(0,31-20)+0 = 11	Skip
{C}(+)	{D,E,A,C}	Max(0,32-20)+0 = 12	Skip
{C}{E}(-)	{D,A}	Max(0,13-20)+6 = 06	Skip
{A}(-)	{D}	Max(0,10-20)+6+3+3=12	Skip
{D}(-)	{}	Max(0,0-20)+27 = 27	Skip
{E}(+)	{E}	Max(0,13-20)+11 = 11	Skip
{A}(+)	{E,A}	Max(0,16-20)+05 = 05	<u>Add</u>

Progress Bar: { D,E } --> {D,E,A} --> {E,A}

Neighbor(+/-)	State	Error Cost	Action
--	{D,E}	Max(0,23-20)+3+3 = 09	--
{B}(+)	{D,E,B}	Max(0,28-20)+3 = 11	Skip
{C}(+)	{D,E,C}	Max(0,29-20)+3 = 12	Skip
{A}(+)	{D,E,A}	Max(0,26-20)+0 = 06	Add
{B}(+)	{D,E,A,B}	Max(0,31-20)+0 = 11	Skip
{C}(+)	{D,E,A,C}	Max(0,32-20)+0 = 12	Skip
{C}{E}(-)	{D,A}	Max(0,13-20)+6 = 06	Skip
{A}(-)	{D}	Max(0,10-20)+6+3+3=12	Skip
{D}(-)	{}	Max(0,0-20)+27 = 27	Skip
{E}(+)	{E}	Max(0,13-20)+11 = 11	Skip
{A}(+)	{E,A}	Max(0,16-20)+05 = 05	Add
{C}(+)	{E,A,C}	Max(0,22-20)+05 = 07	Skip
{D}(+)	{E,A,D}	Max(0,26-20)+05 = 06	Skip
{B}(+)	{E,A,B}	Max(0,21-20)+00 = 01	Add

Progress Bar: { D,E } --> {D,E,A} --> {E,A} --> {E,A,B}

Neighbor(+/-)	State	Error Cost	Action
--	{D,E}	Max(0,23-20)+3+3 = 09	--
{B}(+)	{D,E,B}	Max(0,28-20)+3 = 11	Skip
{C}(+)	{D,E,C}	Max(0,29-20)+3 = 12	Skip
{A}(+)	{D,E,A}	Max(0,26-20)+0 = 06	Add
{B}(+)	{D,E,A,B}	Max(0,31-20)+0 = 11	Skip
{C}(+)	{D,E,A,C}	Max(0,32-20)+0 = 12	Skip
{C}{E}(-)	{D,A}	Max(0,13-20)+6 = 06	Skip
{A}(-)	{D}	Max(0,10-20)+6+3+3=12	Skip
{D}(-)	{}	Max(0,0-20)+27 = 27	Skip
{E}(+)	{E}	Max(0,13-20)+11 = 11	Skip
{A}(+)	{E,A}	Max(0,16-20)+05 = 05	Add
{C}(+)	{E,A,C}	Max(0,22-20)+05 = 07	Skip
{D}(+)	{E,A,D}	Max(0,26-20)+05 = 06	Skip
{B}(+)	{E,A,B}	Max(0,21-20)+00 = 01	Add
{C}(+)	{E,A,B,C}	Max(0,27-20)+00 = 07	Skip
{D}(+)	{E,A,B,D}	Max(0,31-20)+00 = 11	Skip
{A}{D}(-)	{E,B}	Max(0,18-20)+03 = 03	Skip
{A}(+)	{E,A}	Max(0,16-20)+05 = 05	Skip
{E}(-)	{A}	Max(0,03-20)+21 = 21	Skip
{B}(+)	{A,B}	Max(0,8-20)+16 = 16	Skip

Progress Bar: { D,E } --> {D,E,A} --> {E,A} --> {E,A,B}

No solution found! Min. Err (1) was achieved by {E,A,B}

End of Assignment. Thank you!