# Project 2 Fall 2023 Algorithm

The algorithm is based on an extension of hill climbing (see lecture notes). In hill climbing the child node with the highest heuristic value is expanded, whereas with the modified version which we shall be using, any of the child nodes i with uphill moves (those that result in an increase in the heuristic value ) can be expanded with a certain probability. If there are k child nodes with uphill moves, then the probability of the ith child being chosen is:

--- eq(1)

The project is based on a network flow algorithm where the objective is to maximize the flow into the sink (destination) node T of the graph. There are a number of real-life use cases for network flow including an oil pipeline, water pipelines, computer networks, and so on.

For the example graph the objective is to maximize the flow into the sink node T.

Flow into T after optimization with hill climbing = 1+8+5=14

Note: optimal flow = 1+8+5=14

Flow into T before optimization = 1+3+4=8

1/5

Flow into T before optimization = 1+4+4=9

1/5

2/6

1/3

3/3

4/5

2/2

2/3

4/8

2/9

4/5

6/6

1/3

3/3

5/5

2/2

3/3

8/8

2/9

5/5

5/5

To achieve this in *general* the algorithm needs to:

1. Generate all paths P from the sink node T to the source node S (see Tutorial 4 for code, it utilizes the all\_simple\_paths\_edges function call).
2. For each path p in P compute the increase in flow () that can be accomplished in p. The value of for a path p is the *minimum* increase that can be permitted when taken across all edges of the path p. f no path p exists, (i.e., if ), then return the current solution, else for each path p create a child of the current node. # This is the successor function.
3. Select the best child (path) according to the probability expression in eq. (1).
4. For the selected path p in step 3, increase the flow for every edge in p by ()
5. Go to 1.

Note: In the above algorithm we can guarantee that the law of conservation is obeyed due to step 2.

Once the solution is obtained compare the solution with the hill climbing version supported by Networkx (see Tutorial 4) for details.

# Trace of hill climbing algorithm for returning the optimal result (flow =14 into node T)

The execution starts with the example graph (Figure 1 A, the example graph EG) at the root node.

p=t,v,s

h=2

p=t,w,v,s

h=4

p=t,w,s

h=0

p=t,w,u,s

h=0

p=t,z,u,s

h=1

p=t,z,w,s

h=0

p=t,z,w,v,s

h=1

p=t,z,w,u,s

h=0

Child node 2 is the node with the highest outcome. It is expanded next. As a result of the expansion the graph becomes:

6/6

1/3

3/3

3/5

2/2

2/3

8/8

2/9

4/5

5/5

Now expanding node 2 produces:

p=t,z,w,v,s

h=0

p=t,z,u,s

h=1

p=t,w,s

h=0

p=t,v,s

h=0

Node 11 has the highest h value and thus it is expanded. This expansion results in the following graph:

6/6

1/3

3/3

5/5

2/2

3/3

8/8

2/9

5/5

5/5

After C11 is expanded all remaining nodes have h=0, thus the algorithm terminates with a flow of 14 into node T.

Note: The execution trace of the algorithm above was targeted at the original Hill Climbing (HC) algorithm where the best node (i.e., the one with the highest h value) was expanded. In R2 and R3 of Part B you are required to use the stochastic version of HC. The stochastic version is controlled by eq(1).

Let us take an example. Suppose that 3 children C1, C2 and C3 have uphill movements in a given iteration. Suppose that the 3 children C1, C2 and C3 have h values 3,8 and 1. Then C1, C2 and C3 have probabilities of 3/12, 8/12 and 1/12 of being chosen respectively.

To implement this, simply generate a random number between 1 and 12. If the number falls in the interval [1,3] C1 is chosen, if it falls within [4,11] then C2 is chosen, else C3 is chosen.