CS577 HW8

William Shu, Haoyi Lu

Our algorithm:

findBestMinCut(G(E, V, f)):

for each e in E:

f’(e) = 10E(f(e) + 1/(10E))

subroutine with efficient max-flow algorithm (finding max-flow)

get residual graph G’ of the max-flow

bfs(G’) 🡪 return vertexes K which is reachable to s in graph G’

for each e in E:

if e’s start included in K but end not included in K

CUT.add(e)

Return CUT

Running time:

Add penalty factor step: O(E)

Max-flow algorithm: O(EV)

Get residual graph: O(E)

Bfs(G’): O(E)

Check edges: O(E)

Total: O(EV)

We are using method of regulation. We add additional penalty term to the cost function to encourage a solution with the smallest number. Then we assume penalty factor is a, we get new flow of f(e) 🡪 f(e) + a (in this assumption, the min-cut might change). The value of a is crucial because an improper penalty factor will make the min-cut overflow. (Like the graph shown above, the second graph is 5 which is too large). Then the problem become: finding a penalty factor which is smaller enough to make min-cut not overflow.

If we multiply each edge with a positive integer, then min-cut will not change. If we multiply each edge with a positive integer k and the total number of flow is n. The sum of is 𝑘𝑛. If there is a min-cut in which the sum of flow in the edges is smaller than 𝑘𝑛,

the sum of flow in the edges in this min-cut in the previous graph will less than 𝑛.

Based on our assumption, and the sum of flow is n, which means that there won’t be a min-cut in the new graph which isn’t a min-cut in the previous graph.