University of Washington March 13, 2020
Department of Computer Science and Engineering
CSE 417, Winter 2020

Homework 9, Problem 4

Problem 4 (10 points):

In a standard $s-t$ Maximum-Flow Problem, we assume edges have capacities, and there is no limit on how much flow is allowed to pass through a node. In this problem, we consider the variant of the Maximum-Flow problem with node capacities.

Let $G = (V, E)$ be a directed graph, with source $s \in V$, sink $t \in V$, and nonnegative node capacities $\{c_v \geq 0\}$ for each $v \in V$. Given a flow f in this graph, the flow through a node v is defined as $f^{\text{in}}(v)$, the sum of the flows on the incoming edges to v. We say that a flow is feasible if it satisfies the usual flow-conservation constraints and the node-capacity constraints: $f^{\text{in}}(v) \leq c_v$ for all nodes.

Give a polynomial-time algorithm to find an $s-t$ maximum flow in such a node-capacitated network. Justify the correctness of your algorithm.

Answer:

algorithm:

step 1:build a new graph (G') substitute each node (v) to be two node (v_0, v_1) with an edge of capacity c_v
step 2:set the incoming edge of v link to v_0, and set outgoing edge of v link to v_1
setp 3:conduct Ford-Fulkerson algorithm to find maximum flow on the new graph G'

proof:

By doing step 1, $f^{\text{in}}(v) \leq c_v$ can be convert to $f`(v_1, v_2) \leq c_v$, since any flow flow through $v$ must flow through the new edge $v_1, v_2$, it garantees for each node, there are new edge constraint to make sure $f`(v_1, v_2) \leq c_v$, and therefore it satisfies capacity condition.
By doing step 2, the incoming edge and outgoing edge was link to the new added node, with infinity capacity as original problem. Since it remains the original structure of the graph, what can flow into and outof the node become what can flow into and out of the edge, as long as the original problem satisfies the conservation condition, the new reduced problem satisfies, hence the new graph satisfies conservation condition.
By doing so, the algorithm convert node flow constraint to edge flow constraint and hence can be solved by Ford-Fulkerson algorithm since it reduced to a common maximum flow problem. And since Ford-Fulkerson algorithm has polynomial time ($O(n^3)$ according to Kleinberg textbook, p367 7.33), and the step 1 and step 2 with linear time complexity, the overall algorithm has polynomial time.