CS535 Fall 2022 HW3 Sample Solutions

1. First reduce the network as in what we do in the TS feasibility test, then construct the residual network. Start from v in the residual network to compute the set of all reachable vertices by graph traversal in linear time, and that is the minimal set U containing v that satisfies $c\big(\delta^{in}(U)\big) = b(U)$.

2. Suppose $a = (u, v)$.

   If the flow on arc a was not tight before we change the capacity then there is no need to change f.

   If a was tight, we first reduce the flow on a by 1. In the flow network we search for an $s - u$ path P_1 with positive flow on it. Reduce the flow on P_1 by 1 and adjust the residual graph correspondingly. Then we search in the flow network a $v - t$ path P_2 with positive flow and do the same. Finally we search for an augmenting path in the residual network and push 1 unit of flow, if such a path can be found. All 3 searches can be done by a linear time traversal.

3. First mark all the unsaturated edges non-essential. We then compute the strongly connected components in the residual graph in linear time. Let U be any strongly connected component, mark all arcs $a(u, v), u, v \in U$ non-essential, for there exists a path from v to u, forming a circuit with $a(u, v)$. Pushing flow on this circuit in the residual network yields a new max flow where arc $a(u, v)$ is not saturated. The marking can also be done in linear time.

   All the edges that are not marked non-essential are essential edges.

4. Create a source s and a sink t.

   Create a node u_j $\forall j \in J$ representing each job. For each job node u_j, create an arc (s, u_j) with capacity p_j representing its requirement.

   Sort all the beginning and due days into a list L of at most $2|J|$ time intervals. For each machine i and each time interval L_k, create a node v_{ik}, and connect it to sink t with capacity equal to the width of the interval.

   For each job j and machine i, create arcs $(u_j, v_{ik}), \forall k$ with unlimited capacity.

   Now a feasible schedule exists if and only if we can create a flow in this network with amount $\sum_j p_j$.

5. Let p be the source node. Create a sink t. For all subordinate node $s \in S$, create an arc (s, t) with infinity capacity. Now finding a min-cut in this network gives the minimum effort required to cut p off.

6. For each max flow f, get a rounded integer flow f_1. In the space of all feasible flows, extend f on the direction of $\overrightarrow{(f_1, f)}$ until you hit a feasibility constraint $0 \le f(a) \le c(a)$ for some arc a. This gives a new flow f_2. It is clear that f is a convex combination of f_1 and f_2 with $f = wf_1 + (1-w)f_2, 0 < w < 1$. Since both f and f_1 are max flow, f_2 is also a max flow. Since all constraints are integer, f_2 has at least one fewer fractional flow edge than f. Make f_2 your new f and repeat this process to decompose f into f_1, which is integer, and f_2. Repeat until all edges in the latest f_2 are integer.

   In each iteration, f_1 is one integer max flow of the final decomposition with weight w. Each rounding can be done in $O(nm)$ and each f_2 can be computed in $O(m)$. Since each iteration eliminates at least 1 fractional edge, the number of iterations is $O(m)$.