HW9

Keerthana Golla

November 2023

Question-1

Algorithm: Update Maximum Flow After Decreasing Edge Capacity Input:

1. Flow network $G = (V, E)$ with non negative integer capacities.
2. Edge $(u, v) \in E$ with $f(u, v) = c(u, v)$ (capacity is reduced by 1).

Output: New maximum flow in the modified network G'.

Procedure:

Initialization:

- Create G', the flow network with the reduced capacity.
- Initialize the flow f in G.

Update Flow:

- Decrease the flow value on edge (u, v) by 1: $f(u, v) = f(u, v) - 1$.
- This results in an excess flow at node u and a deficit at node v, both with 1 unit.

Push Excess Flow:

- Use BFS or DFS to find a path from u (excess) to t in the residual graph of G'.
- If a path is found, push the excess flow along this path. If not, push the flow from u back to s and v to t .

Return Updated Flow:

- After adjusting the flows, return the modified flow as the new maximum flow in G'.

Proof of Correctness:

1. Validity: Flow in G' respects capacities and maintains flow conservation since we only adjust flow along valid paths.
2. Optimality: Given the unique minimum cut in G, decreasing the capacity of edge (u, v) by 1 necessitates a decrease of 1 in the maximum flow. The algorithm precisely accomplishes this.

   This change creates an imbalance between the in-flows and out-flows at nodes u and v respectively. To rectify this, we examine the residual graph and seek to find a path from t to v and another path from u to s, which can be accomplished in $O(|V| + |E|)$ time. Subsequently, we decrease the flow on all edges of these two paths from t to v and from u to s. This procedure resolves the imbalance of in-flow and out-flow at nodes u and v. However, it's important to note that the maximum flow in the graph is decreased by one.Since given, that we have already found the optimal flow on graph G and that this flow is an integral flow (i.e., for all(a, b) E, $f(a, b)$ is an integer). Call this flow f and uses edge (u, v) to its full capacity, i.e., $f(u, v) = c(u, v)$.so if $c(u, v)$is decreased then unique min cut will be decreased by 1 hence it's important to note that the maximum flow in the graph is decreased by one.

Proof of Runtime:

1. Flow Update: Updating the flow on edge (u, v) takes $O(1)$ time.
2. Search for Path: Using BFS/DFS to find a path in the residual graph is $O(|V| + |E|)$.And also for making the graph have a valid flow.
3. Total Time: The overall runtime is dominated by the BFS/DFS search, making the algorithm $O(|V| + |E|)$.

Therefore, the algorithm efficiently finds the new maximum flow in G' after decreasing the capacity of edge (u, v) by 1 in $O(|V| + |E|)$ time, as required.

Question-2

To prove that for any directed graph G = (V, E) with non-negative capacities c : E → R+, there always exists a maximum flow f* whose support has no directed cycle, we can use the concept of augmenting paths and proof by contradiction.

We'll prove this by showing that if there is a directed cycle in the support of a flow f, then it is possible to find an augmenting path that increases the flow without violating capacity constraints. This implies that a flow with a directed cycle in its support is not a maximum flow, and we can iteratively find augmenting paths until we reach a maximum flow without directed cycles.

Proof:

1. Start with an initial flow f that satisfies capacity constraints but may contain directed cycles in its support.

2. If there is no directed cycle in the support of f, we already have a flow that meets the desired criteria (no directed cycles in the support).

3. If there is a directed cycle in the support of f, then we can find an augmenting path within this cycle. Consider any edge $e = (u, v)$ in the directed cycle. Since $f(e) > 0$, we can find an augmenting path P from s to t that includes edge e, bypassing the cycle.

4. We can increase the flow along path P by the minimum capacity along the path, which is determined by the minimum capacity of any edge on the path. Let δ be the minimum capacity along path P.

5. Update the flow f by increasing the flow along path P by δ and decreasing the flow along the cycle by δ (specifically, subtract δ from all edges in the cycle). This maintains flow conservation at all nodes.

6. The new flow f' obtained after this augmentation has a greater value than f because we increased the flow along path P.

7. Repeat steps 2 to 6 until there are no more directed cycles in the support of the flow.

8. The final flow f^* obtained using this process is a maximum flow and has no directed cycles in its support because we ensured that every augmenting path bypassed any cycles.

Therefore, for any directed graph G with non-negative capacities, there always exists a maximum flow f^* whose support has no directed cycle. This is proven by iteratively finding augmenting paths to eliminate directed cycles from the support of the flow, and it shows that if there is a directed cycle in the support of a flow f, you can find an augmenting path that increases the flow without violating capacity constraints, ultimately leading to a maximum flow with no directed cycles in its support.

In conclusion : Consider any directed cycle in the support of f*. Since this is a directed cycle, it means there is a path from the source s to the sink t that goes through this cycle. Since the flow is conserved, the amount of flow entering the cycle from s must be equal to the amount of flow leaving the cycle towards t. If there were a cycle in the support of f*, it would imply that you can keep sending flow around the cycle indefinitely without reaching the sink t. This would mean that f* is not a maximum flow, which is a contradiction.

Hence, a maximum flow f* must not have a directed cycle in its support. This completes the proof.

Additional note ,

In the context of maximum flow problems, self-loops or cycles that involve intermediate nodes (s-u-v-s) do not contribute to increasing the flow from the source (s) to the sink (t). These cycles don't violate flow conservation within themselves, but they don't affect the overall flow from s to t. To ensure the efficient operation of maximum flow algorithms, incoming edges to the source and outgoing edges from the sink are typically removed to eliminate such self-loops or cycles. This removal simplifies the flow network and prevents unintended behaviors - just like how professor explained the flight problem and 2 people keeps on travelling in loop , to avoid such situations we remove any incoming edges to Source and outgoing edge from sink .