Module 4 Quiz Review

Network Flow

Network flow problems

A *flow network* is a directed graph with:

- A single source node $s \in V$
- A single terminus/sink node $t \in V$
- Integer capacities on each edge

Max-flow problem: What is the maximum amount of *flow* we can push through the network?

Augmenting paths

Let G_f be graph whose edge weights are *residual capacities*—the amount of additional flow we could push along an edge.

An *augmenting path* is a path from $s \to t$ such that all edges have residual capacity > 0.

"Backflow" edges Ford-Fulkerson introduces the concept of "backflow." We visualize "un-pushing" some flow across an edge by modeling backflow: a reverse edge whose capacity is the amount of flow moving across the forward edge.

Ford-Fulkerson Algorithm

Ford-Fulkerson is an algorithm for max-flow.

```
1. Init $f(u, v) = 0$ for all $(u, v) \in E$
2. While there is an augmenting path $p$ from $s$ to $t$ in $G_f$:
    1. Let $c_p$ be the minimum residual capacity along each edge in $p$
    2. Increase $f(u, v)$ by $p$
    3. Decrease $f(v, u)$ by $p$
```

Return flow out of s (or, equivalently, into t)

Time complexity Ford-Fulkerson is $O(E \times f)$ in runtime, where E is the number of edges in G and f is the final max-flow. (Pseudopolynomial runtime)

Min-Cut

A cut $C = (A, B)$ is a partition of G such that one of A, B contains s and the other contains t. In other words:

- $A \cap B = \{\}$
- $A \cup B = V$
- $s \in A, t \in B \vee s \in B, t \in A$

We care about the net flow across the cut, that is, the flow through edges $a \to b$ minus the flow through edges $b \to a$ $(a \in A, b \in B)$

Flow-value lemma

Let f be any flow and $C = (A, B)$ be any cut. Then the net flow across C is equal to f.

Intuitive explanation No matter where you place the cut, it will separate the source and sink. So all flow must cross the cut at some point.

Proof by induction We prove this by induction on the size of B.

Base case: $B = \{t\}, A = V - \{t\}$. All flow in the network has to sink into t, so the net flow across C must equal f. $\therefore NF(A, B) = f$.

Inductive hypothesis: For all B of size k or less, $NF(A, B) = f$.

Inductive case: $B' = B + \{a\}, A' = A - \{a\}$ (move some vertex a from A to B). Our network flow is still f. The flow across C' is the same as C. Why?

Moving a' across the cut:

- Subtracts flow from $a \to B$, which used to cross the cut but is now internal.
- Adds negative flow that goes from a back to nodes in A.
- Subtracts negative flow from $B \to a$, which used to cross the cut but is now internal.
- Adds negative flow that used to be internal, but now crosses the cut.

But because a is internal, the net flow through the node is zero. So we can show that $NF(C') = NF(C) + 0$. Inductive case holds, and so does the flow-value lemma.

Max-flow min-cut theorem

Claim: the max value of a s-t flow equals the min-capacity of an s-t cut.

Intuitive explanation: the min-capacity cut is the "bottleneck" that all flow must cross, so it must equal the max flow through a graph.

"Strong" max-flow min-cut theorem

Theorem statement: For some flow f, the following are equivalent conditions:

1. There exists a min cut with capacity f.
2. f is a max flow.
3. There is no augmenting path with respect to f.

Theorem proof, strong formulation Assume there exists a min-cut with capacity f, but that f is not a max flow. Then there must exist an f' that is a max flow, and $f' > f$. But then $f' > f = \mathit{Capacity}(C)$, which contradicts the weak duality. Therefore: $1 \implies 2$.

Assume f is a max flow, but there exists an augmenting path with respect to f. Then Ford-Fulkerson suggests a greater max-flow which contradicts that f is a max flow. Therefore $2 \implies 3$.

Assume there's no augmenting path for f. Therefore there exists a cut that separates nodes that are reachable from those that aren't.

By flow-value, that cut has capacity f. Therefore $3 \implies 1$.

Weak duality

Let f be any flow, $C = (A, B)$ be any cut. Then $f \leq \mathit{Capacity}(C)$.

Follows almost directly from the max-flow min-cut theorem.

Reductions

A *reduction* is a means of reducing an instance of one problem to another in such a way that a solver for the second problem can be used to solve the first.

Formally, we say a problem A is *polynomial-time reducible* to another problem B if we can convert instances of A to instances of B and solutions of B to solutions of A in polynomial time. Denote this $A \leq_p B$.

We can also interpret this to mean that A is "no harder" than B.

We say A and B are polynomial-time equivalent if $A \leq_p B$ and $B \leq_p A$.

Polynomial

Network Flow Reductions

Several problems can be solved via reduction to network flow.

Bipartite matching

Decision problem: given a bipartite undirected graph $G = (V, E)$ where V is partitioned into L, R and E connects compatible vertices $L \leftrightarrow R$, what's the maximum number of elements we can match?

Reduction: Create source/sink vertices s, t. Create an edge between s and every element of L, and between t and every element of R. Assign capacity 1 to all edges. The maximum number of matched elements = max flow through G'.

Circulation networks

Generalize flow networks beyond having a single source or single sink. Instead, there are multiple sources (with negative integer demand), and multiple sinks (with positive integer demand).

Circulation with demands is a *decision* problem rather than an optimization problem, like maximum flow. Instead of some value representing the max flow we can push through, our response is “yes” or “no” depending on whether a given circulation network is feasible.

Each node in the circulation network has an associated demand. Sources have negative demand, sinks have positive demand, and internal nodes have zero demand.

We solve the circulation network problem via a reduction to single-source/single-sink max-flow.

To reduce, create a super-source $s*$ and super-sink $t*$. Connect $s*$ to each source node with an edge of capacity equal to the demand of that source node. Connect $t*$ to each sink node with an edge of capacity equal to the demand of that sink node. Run any algorithm for max-flow, and return “Yes” if the max flow = the total demand of all sink nodes.

Circulation networks with lower bounds

Generalize circulation networks to include not only an upper bound on flow (in the form of an edge capacity), but also a lower bound.

We accomplish this via a reduction to a standard circulation network, which can then be itself reduced to a standard single-source/single-sink flow network.

Reduce as follows: 1. Force flow across each edge with a lower bound by setting its capacity to (upper bound) – (lower bound). 2. Increase the demand at the start of the edge by the lower bound. 3. Decrease the demand at the end of the edge by the lower bound.

Now have a standard circulation network, which can be reduced as noted above.

Complexity classes

P := roughly, the set of problems solvable in polynomial time. NP := roughly, the set of problems solvable in polynomial time by a nondeterministic Turing machine. (Equivalently, the set of problems whose solutions can be verified in polynomial time given a certificate.)

NP-Hard is the set of problems “at least as hard” as any problem in NP. A problem $x \in$ NP-Hard $\iff \forall y \in NP, y \leq_p x$.

NP-Complete

A problem is NP-Complete if it is both NP *and* NP-Hard.

Proving NP-Complete: 1. Prove NP - Show that the problem's solution can be verified in polynomial time given a certificate. 2. Prove NP-Hard - Typically, show that some known NP-Hard problem reduces to this problem, then appeal to transitivity.