CS 3510 Algorithms 4/9/2024

Max Flow & Min Cut

Lecturer: Jan van den Brand *Scribe(s): Ankith Thalanki*

In today's lecture we covered two seemingly different problems, *st* max-flow and *st* min-cut. We learn that while these two graph problems may appear different, they both have the same answer and certain properties they possess can be used to create an algorithm that solves both of them.

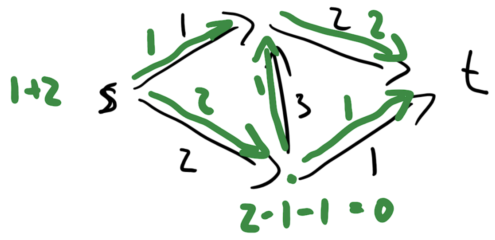
1 Max Flow

1.1 Flow

A Flow network is a weighted directed graph, where the weight of an edge in the graph represents the max capacity of each edge. The flow of the graph is a series of values, one for each edge, where $0 \leq f_e \leq w_e$ (the flow through an edge is at most the capacity), and where for every vertex v, the condition $\sum_{(u,v)\in E} f_{(u,v)} = \sum_{(v,w)\in E} f_{(v,w)}$ applies. This condition essentially states that for every vertex v the sum of all the flow entering must equal the sum of all the flow leaving. These flow networks can be used to model computer networks, traffics, pipes, etc. In all these situations it can be common to ask for the maximum flow between any two nodes, like the max amount of traffic that can go between Atlanta and Boston. This problem is formulated as the *st* max-flow in a flow network, which determines the max flow between s and t.

1.2 *st* max-flow

As shown earlier, the *st* max-flow determines the max flow between s and t in flow network, essentially just a weighted directed graph. For example, in the graph below, the max flow is 3. The capacities of each edge are represented in black, and the actual flow through each edge is in green.



How might the flow of this graph change if we remove edges? Determining what edges you can remove to reduce the *st* max-flow in a graph is known as the cut of the network.

2 Min Cut

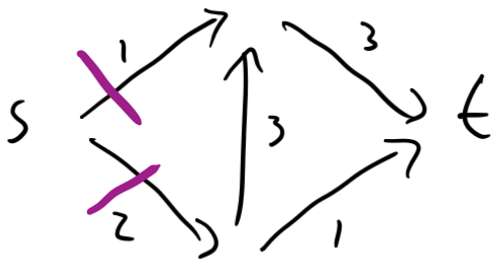
We can visualize the linear programming problem in $\mathcal{N}^x$

2.1 Cut

The cut of a flow network is in many ways completely opposite of the flow of a network. While the *st* flow of a network represents how much of some arbitrary quantity can pass from s and t, the cut of a network asks what edges of a graph need to be removed such that the *st* flow of the network is 0. We can define a cut as some set of vertices $W \subseteq V$, where W represents the vertices connected to s, while $V - W$ represents those connected to t. In more rigid terms, $s \in W$ and $t \notin W$. We define the size of cut as the total capacity of the edges actually cut to separate W from $V - W$, or $\sum_{(u,v)\in E \wedge u \in W \wedge v \notin W} w_{(u,v)}$. Notice that this means that any flow in a flow network must be less than or equal to any cut, as every unit of flow must go through one of the capacities provided by a cut edge.

2.2 *st* min-cut

The *st* min-cut asks for the smallest size of a cut that completely separates s and t, such that the flow between them is 0. For example, in the graph below, W is defined as just s, and the min-cut is represented as the total sum of the cut edges, or just 3.



If you examine the graph closely you can see that any other cut in the graph that completely separates s and t has a larger cut size. If you remember from the previous example however, the *st* max-flow of this graph also happened to be 3. This leads to another revelation, that *st* min-cut and *st* max-flow of a flow network are actually the same. We can reveal this by a proof of Contradiction. Assume that the *st* max-flow of a graph is smaller than the *st* min-cut of this graph. This means that the max-flow is not restricted by the min-cut, but by another section of the flow network which lets even less flow go through. However that means that our min-cut is not valid, as that other section with an even greater restriction would allow us to separate the network while cutting fewer edges. This would mean that max-flow has to equal min-cut (we established earlier that any flow must be $\leq$ cut).

3 Algorithm for Max Flow and Min Cut

Below we present the algorithm used for determining both the *st* min-cut and *st* max-flow of a flow network, using the property that they are equivalent.

Algorithm 1 Solving *st* MAX-FLOW AND *st* MIN-CUT.

```
procedure MaxFlow($G$)
    Initialize $f_e = 0 \; \forall e \in E$
        // Represents if an edge was flipped or not
    Initialize $r_e = 0 \; \forall e \in E$
        $G_f = G$
        Initialize max_flow= 0
        // Determines if a path still exists from $s$ to $t$.  Assumes if
        capacity/weight is 0 the edge isn't present
    while Path($G_f, s, t$) do
        // Assume Path returns edges in the path
        for $(u, v) \in textscPath(G_f, s, t)$ do
            $f_{(u,v)} += 1$
             $w_{(u,v)} := w_{(u,v)} - f_{(u,v)}$
             if $f_{(u,v)} > 0$ then
                Add edge $(v, u)$ with weight $f_{(u,v)}$ to $G'$
            end
        end
        max_flow $+ = 1$
    end
    return max_flow
```