The Maximum Flow Problem

The definitions of capacitated network, s-t flow, value of a flow, and maximum flow are set out on pages 338-340 of Section 7.1 of KT(Chapter 7 available in MyCourses). So are the definition of $f^{in}(S)$ and $f^{out}(S)$ for any subset S of the vertices, including a single vertex v. Note that if a distribution system has many sources or sinks, we can add an auxiliary source with an arc to each of the original sources, and an auxiliary sink with an arc from each of the orginal sinks to create a capacitated network.

The definitions of an s-t cut is set out on pages 346-348 of Section 7.2 of KT. So to is the definition of the capacaity $c(A, B)$ of an s-t cut (A, B). Those pages also contain a proof that the capacity of any s-t cut is an upper bound on the volume of any s-t flow.

In class, we saw a number of examples of capacitated networks for which the maximum value of an s-t flow was equal to the minimum capacity of an s-t cut.

It turns out that this is true for every capacitated network, and there is an efficient algorithm which given a capacitated network finds a maximum value s-t flow and a minimum capacity s-t cut.

Sometimes we want the flow values to be integer-valued. As when our flow represents passengers using an airline's flight network to get from one place to another. As we shall see, provided all the capacities are integer, our algorithm returns a maximum value$s$-t flow which is integer valued.

We will describe and analyze this algoithm in a few weeks and thereby prove:

Theorem 1 (The Max-Flow Min-Cut Theorem) *In any capacitated network, the maximum value of an s-t flow is equal to the minimum capacity of an s-t cut.*

and:

Theorem 2 *If all the capacities in a capacitated network are integer-valued then there is an s-t flow f of maximum value such that $f(e)$ is integer valued for every e. Furthermore if f has value v then we can find f together with ans-t cut of capacity v in $O(v|E(G)|)$ time.*

Letting C be the capacity of the cut $(\{s\}, V - \{s\})$, ie. the sum of the capacities of the edges out of s, we have that C is an upper bound on the

value of a maximum flow so we can replace $v|E(G)|$ here by $C|E(G)|$. We note we use m for $|E(G)|$.

For now, we show the algorithm's wide applicability by formulating a variety of real world problems as max-flow min-cut problems, so that they can be solved using this algorithm.

We began with the example of Bipartite Matching. The fact that this can be transformed into a Maximum Flow Problem is set out on pages 368-370 of KT (up to the statement of 7.18, you can ignore the last half of page 370) We remark that our transformation had one slight difference from that given in KT. We gave the edges of G capacity $n+1$ rather than capacity 1. This did not change the possible feasible flows. Since the total capacity of the arcs into a vertex$x$ of X is at most one, the flow conservation constraint for X tells us that the total flow on the edges of G leaving x is at most one, even if we increase their capacities.

We then applied the Max-Flow Min-Cut Theorem to deduce the results about the structure of bipartite graphs without perfect matchings set out on pages 371-373 of KT. Because of our different capacity choice we had no need to transform the cut (A, B) by adding some vertices in B.

We also defined a cover C of a bipartite graph as a set of vertices such that there is no edge with both endpoints in $V - C$. The edges of a matching M are disjoint. Thus for every matching M and cover C, each edge of M is incident to a distinct vertex of C and so $|M| \leq |C|$. We saw that the max-flow min-cut theorem iimplies that for every bipartite graph G, the maximum size of a matching is actually equal to the minimum size of a cover. To do so, we considered a minimum capacity s-t cut (A, B) in the auxiliary capacitated network we used to find the maximum matching. We noted there were no edges from $A \cap X$ to $(B \cap Y)$ and set $C = (A \cap Y) \cup (B \cap X)$. We saw $|C|$ was the capacity of (A, B) and hence equallled the size of a maximum matching.

We then discussed edge-disjoint paths in directed graphs. We saw that there were k edge disjoint paths from s to t in a digraph G if and only if there was a flow of value k in the capacitated network obtained from G by deleting the edges into s and the edges out of t and giving the remaining edges capacity 1. This allowed us to develop a polynomial time algorithm to find a maximum size set of edge disjoint paths in G using our Max-Flow Min-Cut Algorithm. This material is discussed in Section 7.6 of KT.

We discussed an alternative approach to the proof of (7.42). We built an arborescence containing the set A of nodes reachable from s in the subgraph of G with vertex set V and edges set those edges satisfying $f(e) = 1$. Now, T

must be in A as otherwise $(A, V - A)$ would be a cut of capacity zero which is impossible. So we can use the directed path from s to t in the arborescence in the proof.