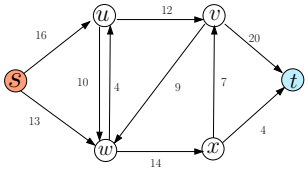
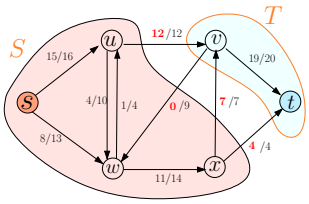
Network Flow III - Applications

4/5/05

1 Previous Lecture



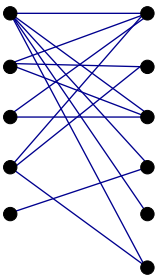
flow network



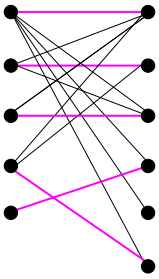
valid flow

Theorem 1.1 *Given a network flow with n vertices and m edges, one can compute a maximum flow in G in $O(n * m^2)$ time.*

2 Maximum Bipartite Matching

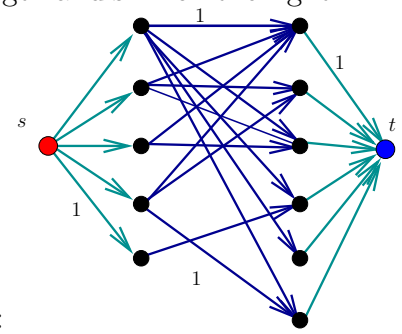


And a maximum matching in this graph is



Theorem 2.1 *One can compute maximum bipartite matching using network flows.*

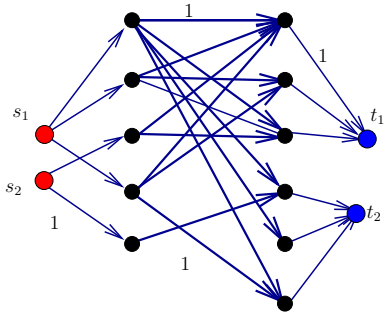
Proof: We create a new graph, with new source on the length and sink on the right. Direct



all edges from left to right, and set their capacity to 1. See: ■

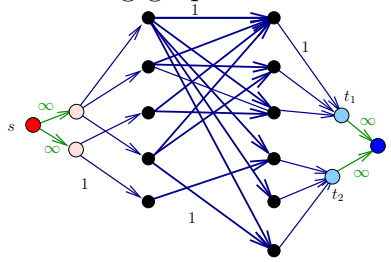
3 Multiple Sources and Sinks

Question 3.1 *Given several sources and sinks, how can we compute maximum flow on such a network?*



Idea: Create a super source, that send all this flow to the old sources, and similarly create a super sink.

Resulting graph:



4 Edge disjoint paths

Question 4.1 *Given a directed/undirected graph G, two vertices s and t, and a parameter k, find k paths from s to t in G, such that they do not share an edge.*

How to solve the problem?

Idea: Convert G (assume directed graph) into a network flow graph H, such that every edge has capacity 1. Find the maximum flow in G, we claim that the maximum flow in the network H, is equal to the number of edge disjoint paths in G.

Lemma 4.2 *If there are k edge disjoint paths in G between s and t, then the maximum flow in H is at least k.*

Proof: Just push flow 1 along each such path. ∎

Lemma 4.3 *Let f be a $0-1$ flow in a network H with flow μ. Then there are μ edge disjoint paths between s and t in H.*

Proof: By induction on the number of edges in H with flow 1 on them. If $\mu = 0$ then there is nothing to prove.

Otherwise, start traveling the graph H from s traveling only along edges with flow 1 on them. We mark such an edge as used, and do not allow one to travel on such an edge again. There are two possibilities:

(i) We reached the target vertex t. In this case, we take this path, add it to the set of output paths, and reduce the flow along the edges of the generated path π to 0. Let H' be the resulting flow network and $f\,'$ the resulting flow. $|f\,'| = \mu - 1$, H' has less edges, and by induction, it has $\mu - 1$ edge disjoint paths in H' between s and t. Together with π this forms μ such paths.

(ii) We visit a vertex v for the second time. In this case, our traversal contains a cycle C, of edges in H that have flow 1 on them. We set the flow along the edges of C to 0 and use induction on the remaining graph (since it has less edges with flow 1 on them). ∎

Since the graph is simple, there are $< n$ edges that leave s. As such, the maximum flow in H is n. Thus, applying the Ford-Fulkerson algorithm, takes $O(mn)$ time. The extraction of the paths can also be done in linear time (verify!). As such, we get:

Theorem 4.4 *Given a directed graph G, and two vertices s and t, one can compute the maximum number of edge disjoint path between s and t in $O(mn)$ time.*

As a consequence we get the following cute result:

Lemma 4.5 *In a directed graph G with nodes s and t the maximum number of edge disjoint $s-t$ paths is equal to the minimum number of edges whose removal separates s from t.*

Proof: Let U be a collection of edge-disjoint paths from s to t in G. If we remove a set F of edges from G and separate s from t, then it must be that every path in U uses at least one edge of F. Thus, the number of edge-disjoint paths is bounded by the number of edges needed to be removed to separate s and t.

As for the other direction, the set F of edges form a cut in G between s and t. to see that, let S be the set of all vertices in G that are reachable from s without using an edge of F. Clearly, if F is minimal then it must be all the edges of the cut (S, T). In particular, the minimum F is the minimum cut between s and t in G, which is by the max-flow min-cut theorem, also the maximum flow in the graph G (where every edge has capacity 1).

But then, by the previous theorem, there are $|F|$ edge disjoint paths in G (since $|F|$ is the amount of the maximum flow). ∎

4.1 Undirected graph

We would like to solve the s-t disjoint path problem for an undirected graph. The natural approach is to duplicate every edge in the undirected graph G, and get a directed graph H. Next apply the algorithm above to H.

The problem is, that in the flow we compute for H (where every edge has capacity 1), is that the computed flow f would both use the edge $u \to v$ and the edge $v \to u$. The observation is however, that in such case, we can remove both edges from the flow f. Both conservation of flow, and the maximum flow are preserved, and as such, if we repeatedly remove those double used edges from the flow, the resulting flow f' is still of the same amount of flow. Next, we extract the edge disjoint paths from the graph, and the resulting paths are now edge disjoint in the original graph.

Lemma 4.6 *There are k edge-disjoint paths in an undirected graph G from s to t if and only if the maximum value of an $s-t$ flow in the directed version H of G is at least k. Furthermore, the Ford-Fulkerson algorithm can be used to find the maximum set of disjoint s-t paths in G in $O(mn)$ time.*

5 Circulations with demands

We next modify and extend the network flow problem. Let $G = (V, E)$ be a directed graph with capacities on the edges. Each vertex v has a demand d_v:

- $d_v > 0$ - sink requiring d_v flow into this node.
- $d_v < 0$ - source with $-d_v$ units of flow leaving it.
- $d_v = 0$ - regular node.

Let S denote all the source vertices and T denote all the target vertices.

Definition 5.1 A *circulation* with demands $\{d_v\}$ is a function f that assigns nonnegative real values to the edges of G, such that:

- **Capacity condition**: $\forall e \in E \;\; f(e) \leq c(e)$.
- **Conservation condition**: $\forall v \in V \;\; f^{in}(v) - f^{out}(v) = d_v$.

$f^{in}(v)$ - flow into v, $f^{out}(v)$ - flow out of v.

Problem 5.2 Is there a circulation that comply with the demand requirements?

See Figure 1 for example.

Lemma 5.3 *If there is a feasible circulation with demands $\{d_v\}$, then $\sum_v d_v = 0$.*

Proof: Since it is a circulation, we have that $d_v = f^{in}(v) - f^{out}(v)$. Summing over all vertices: $\sum_v d_v = \sum_v f^{in}(v) - \sum_v f^{out}(v)$. The flow on every edge is summed twice, one with positive sign, one with negative sign. As such, $\sum_v f^{in}(v) - \sum_v f^{out}(v) = 0$, which implies the claim. ∎

For feasible solution: $D = \sum_{v, d_v > 0} d_v = \sum_{v, d_v < 0} -d_v$.

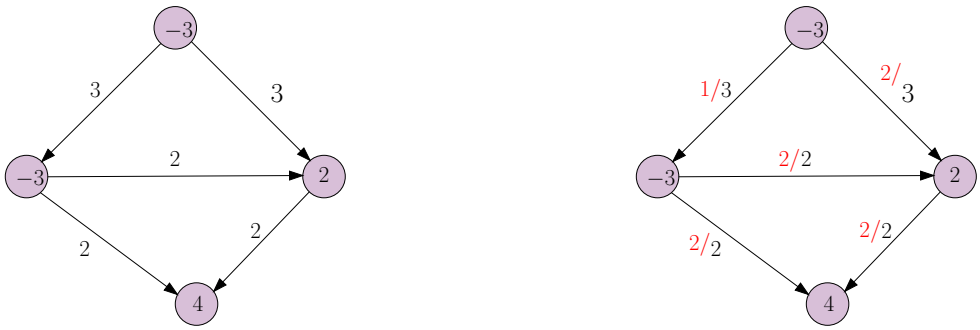


Figure 1: (a) A circulation, and (b) valid flow that fulfill the circulation.

5.1 The algorithm

- $G = (V, E)$ - input flow network with demands on vertices.
- Check that $D = \sum_{v, d_v > 0} d_v = \sum_{v, d_v < 0} -d_v$.
- Create a new super source s, and connect it to all the vertices v with $d_v < 0$. Set the capacity of the edge $s \to v$ to be $-d_v$.
- Create a new super target t. Connect to it all the vertices u with $d_u > 0$. Set capacity on the new edge $u \to t$ to be d_u.
- We have a usual network flow network H. Compute maximum flow on H from s to t. If it is equal to D, then there is a valid circulation, and it is the flow restricted to the original graph. Otherwise, there is no valid circulation.

Theorem 5.4 *There is a feasible circulation with demands $\{d_v\}$ in G and and only if the maximum s-t flow in H has value D. If all capacities and demands in G are integers, and there is a feasible circulation, then there is a feasible circulation that is integer valued.*

6 Circulations with demands and lower bounds

Assume that in addition to specifying circulation and demands on a network G, we also specify for each edge a lower bound on how much flow should be on each edge. Namely, for every edge $e \in E(G)$, we specify $\ell(e) \leq c(e)$, which is a lower bound to how much flow must be on this edge. As before we assume all numbers are integers.

We need now to compute a flow f that fill all the demands on the vertices, and that for any edge e, we have $\ell(e) \leq f(e) \leq c(e)$. The question is how to compute such a flow?

Let use start from the most naive flow, which transfer on every edge, exactly its lower bound. This is a valid flow as far as capacities and lower bounds, but of course, it might violate the demands. Formally, let $f_0(e) = \ell(e)$, for all $e \in E(G)$. Note that f_0 does not even satisfy the the conservation rule:

$$L_v = f_0^{in}(v) - f_0^{out}(v) = \sum_{e \text{ into } v} \ell(e) - \sum_{e \text{ out of } v} \ell(e).$$

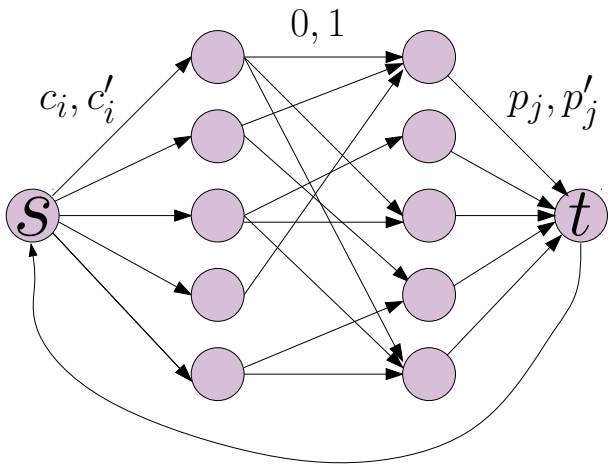


Figure 2: Find if there is a valid circulation in this graph.

If $L_v = d_v$, then we are happy, since this flow satisfies the required demand. Otherwise, there is imbalance at v, and we need to fix it.

Formally, we set a new demand $d'_v = d_v - L_v$ for every node v, and the capacity of every edge e to be $c'(e) = c(e) - \ell(e)$. Let G' denote the new network with those capacities and demands (note, that the lower bounds had "disappeared"). If we can find a circulation f' on G' that satisfies the new demands, then clearly, the flow $f = f_0 + f'$, is a legal circulation, it satisfies the demands and the lower bounds.

But finding such a circulation, is something we already know how to do, using the algorithm of Theorem 5.4. Thus, it follows that we can compute a circulation with lower bounds.

Lemma 6.1 *There is a feasible circulation in G if and only if there is a feasible circulation in G'. If all demands, capacities, and lower bounds in G are integers, and there is a feasible circulation, then there is a feasible circulation that is integer valued.*

Proof: Let f' be a circulation in G'. Let $f(e) = f_0(e) + f'(e)$. Clearly, f satisfies the capacity condition in G, and the lower bounds. Furthermore,

$$f^{in}(v) - f^{out}(v) = \sum_{e \text{ into } v} (\ell(e) + f'(e)) - \sum_{e \text{ out of } v} (\ell(e) + f'(e)) = L_v + (d_v - L_v) = d_v.$$

As such f satisfies the demand conditions on G.

Similarly, let f be a valid circulation in G. Then it is easy to check that $f'(e) = f(e) - \ell(e)$ is a valid circulation for G'. ■

7 Application - survey design

- Want to design a survey.
- Each consumer answer question on subset of products.

- Every consumer can be asked only products they bought.
- Each consumer i asked about a number of products between c_i and c'_i.
- Each product j, must have between p_j and p'_j consumers opinions.

Idea: Reduce to circulation in a graph. See Figure 2 for the resulting graph.