Chapter 14

Network Flow III - Applications

By Sariel Har-Peled, December 17, 2012[1] Version: 0.1

14.1 Edge disjoint paths

14.1.1 Edge-disjoint paths in a directed graphs

Question 14.1.1. *Given a graph* G *(either directed or undirected), two vertices* s *and* t*, and a parameter* k*, the task is to compute* k *paths from* s *to* t *in* G*, such that they are* **edge disjoint***; namely, these paths do not share an edge.*

To solve this problem, we will convert G (assume G is a directed graph for the time being) into a network flow graph H, such that every edge has capacity 1. Find the maximum flow in G (between s and t). We claim that the value of the maximum flow in the network H, is equal to the number of edge disjoint paths in G.

Lemma 14.1.2. *If there are* k *edge disjoint paths in* G *between* s *and* t*, then the maximum flow in* H *is at least* k*.*

Proof: Given k such edge disjoint paths, push one unit of flow along each such path. The resulting flow is legal in h and it has value k. ■

Definition 14.1.3 (0/1-flow). A flow f is a *0/1-flow* if every edge has either no flow on it, or one unit of flow.

Lemma 14.1.4. *Let* f *be a* 0/1 *flow in a network* H *with flow value* μ*. Then there are* μ *edge disjoint paths between* s *and* t *in* H*.*

Proof: By induction on the number of edges in H that has one unit of flow assigned to them by f. If $\mu = 0$ then there is nothing to prove.

Otherwise, start traversing the graph H from s traveling only along edges with flow 1 assigned to them by f. 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. We have $|f'| = \mu - 1$, 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). The value of the flow f did not change by removing C, and as such it follows by induction that there are μ edge disjoint paths between s and t in H. ∎

Since the graph G is simple, there are at most $n = |V(H)|$ 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 by applying the algorithm in the proof of Lemma 14.1.4. As such, we get:

Theorem 14.1.5. *Given a directed graph G with n vertices and m edges, and two vertices s and t, one can compute the maximum number of edge disjoint paths between s and t in H, in $O(mn)$ time.*

As a consequence we get the following cute result:

Lemma 14.1.6. *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. Namely, $|U| \leq |F|$.

As for the other direction, let F be a set of edges thats its removal separates s and t. We claim that the set F form a cut in G between s and t. Indeed, 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, if F contains some edge which is not in (S, T) we can remove it and get a smaller separating set of edges). In particular, the smallest set F with this separating property has the same size as the minimum cut between s and t in G, which is by the max-flow mincut theorem, also the maximum flow in the graph G (where every edge has capacity 1).

But then, by Theorem 14.1.5, there are $|F|$ edge disjoint paths in G (since $|F|$ is the amount of the maximum flow). ∎

14.1.2 Edge-disjoint paths in undirected graphs

We would like to solve the s-t disjoint path problem for an undirected graph.

Problem 14.1.7. Given undirected graph G, s and t, find the maximum number of edge-disjoint paths in G between s and t.

The natural approach is to duplicate every edge in the undirected graph G, and get a (new) directed graph H. Next, apply the algorithm of Section 14.1.1 to H.

So compute for H the maximum flow f (where every edge has capacity 1). The problem is the flow f might use simultaneously the two edges $(u \to v)$ and $(v \to u)$. Observe, however, that in such case we can remove both edges from the flow f. In the resulting flow is legal and has the same value. As such, if we repeatedly remove those "double edges" from the flow f, the resulting flow f' has the same value. Next, we extract the edge disjoint paths from the graph, and the resulting paths are now edge disjoint in the original graph.

Lemma 14.1.8. *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.*

14.2 Circulations with demands

14.2.1 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 sink/target vertices.

For a concrete example of an instance of circulation with demands, see figure on the right.

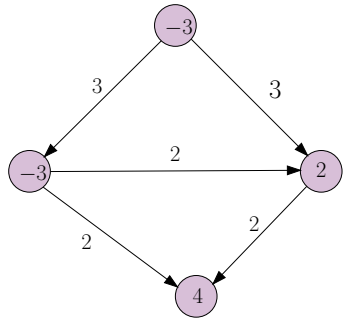


Figure 14.1: Instance of circulation with demands.

Definition 14.2.1. A ***circulation*** with demands $\{d_v\}$ is a function f that assigns nonnegative real values to the edges of G, such that:

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

Here, for a vertex v, let $f^{in}(v)$ denotes the flow into v and $f^{out}(v)$ denotes the flow out of v.

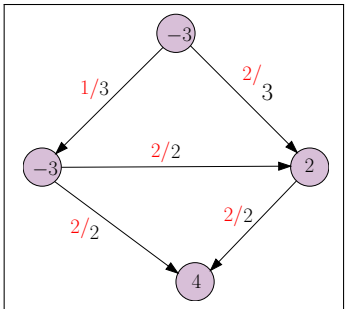


Figure 14.2: A valid circulation for the instance of Figure 14.1.

Problem 14.2.2. Is there a circulation that comply with the demand requirements?

See Figure 14.1 and Figure 14.2 for an example.

Lemma 14.2.3. *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,

$$\sum_v d_v = \sum_v f^{in}(v) - \sum_v f^{out}(v) = 0,$$

which implies the claim. ∎

In particular, this implies that there is a feasible solution only if

$$D = \sum_{v, d_v > 0} d_v = \sum_{v, d_v < 0} -d_v.$$

14.2.1.1 The algorithm for computing a circulation

The algorithm performs the following steps:

(A) $\mathsf{G} = (V, E)$ - input flow network with demands on vertices.
(B) Check that $D = \sum_{v, d_v > 0} d_v = \sum_{v, d_v < 0} -d_v$.
(C) 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 $-d_v$.
(D) 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.
(E) On the resulting network flow network H (which is a standard instance of network flow). 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 14.2.4. *There is a feasible circulation with demands $\{d_v\}$ in G if 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.*

14.3 Circulations with demands and lower bounds

Assume that in addition to specifying a 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 $e \in E(\mathsf{G})$, 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 $e \in E(\mathsf{G})$. Note that f_0 does not even satisfy the conservation rule:

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

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 14.2.4. Thus, it follows that we can compute a circulation with lower bounds.

Lemma 14.3.1. *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.

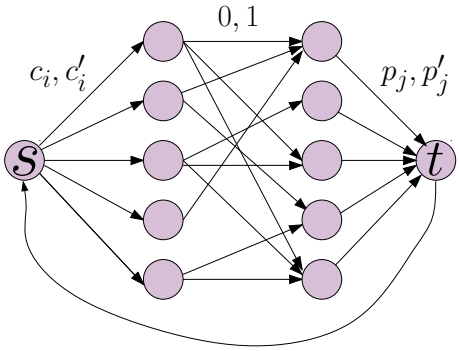
Similarly, let f be a valid circulation in G. Then it is easy to check that $f'(e) = f(e) - \ell(e)$ is a valid circulation for G'. ∎

14.4 Applications

14.4.1 Survey design

We would like to design a survey of products used by consumers (i.e., "Consumer i: what did you think of product j?"). The ith consumer agreed in advance to answer a certain number of questions in the range $[c_i, c'_i]$. Similarly, for each product j we would like to have at least p_j opinions about it, but not more than p'_j. Each consumer can be asked about a subset of the products which they consumed. In particular, we assume that we know in advance all the products each consumer used, and the above constraints. The question is how to assign questions to consumers, so that we get all the information we want to get, and every consumer is being asked a valid number of questions.

The idea of our solution is to reduce the design of the survey to the problem of computing a circulation in graph. First, we build a bipartite graph having consumers on one side, and products on the other side. Next, we insert the edge between consumer i and product j if the product was used by this consumer. The capacity of this edge is going to be 1. Intuitively, we are going to compute a flow in this network which is going to be an integer number. As such, every edge would be assigned either 0 or 1, where 1 is interpreted as asking the consumer about this product.



The next step, is to connect a source to all the consumers, where the edge $(s \to i)$ has lower bound c_i and upper bound c'_i. Similarly, we connect all the products to the destination t, where $(j \to t)$ has lower bound p_j and upper bound p'_j. We would like to compute a flow from s to t in this network that comply with the constraints. However, we only know how to compute a circulation on such a network. To overcome this, we create an edge with infinite capacity between t and s. Now, we are only looking for a valid circulation in the resulting graph G which complies with the aforementioned constraints. See figure on the right for an example of G.

Given a circulation f in G it is straightforward to interpret it as a survey design (i.e., all middle edges with flow 1 are questions to be asked in the survey). Similarly, one can verify that given a valid survey, it can be interpreted as a valid circulation in G. Thus, computing circulation in G indeed solves our problem.

We summarize:

Lemma 14.4.1. *Given n consumers and u products with their constraints $c_1, c'_1, c_2, c'_2, \ldots, c_n, c'_n$, $p_1, p'_1, \ldots, p_u, p'_u$ and a list of length m of which products where used by which consumers. An algorithm can compute a valid survey under these constraints, if such a survey exists, in time $O((n+u)m^2)$.*