Last update: June 8, 2022 Translated From: e-maxx.ru

Flows with demands

In a normal flow network the flow of an edge is only limited by the capacity $c(e)$ from above and by 0 from below. In this article we will discuss flow networks, where we additionally require the flow of each edge to have a certain amount, i.e. we bound the flow from below by a **demand** function $d(e)$:

$$d(e) \le f(e) \le c(e)$$

So next each edge has a minimal flow value, that we have to pass along the edge.

This is a generalization of the normal flow problem, since setting $d(e) = 0$ for all edges e gives a normal flow network. Notice, that in the normal flow network it is extremely trivial to find a valid flow, just setting $f(e) = 0$ is already a valid one. However if the flow of each edge has to satisfy a demand, than suddenly finding a valid flow is already pretty complicated.

We will consider two problems:

1. finding an arbitrary flow that satisfies all constraints
2. finding a minimal flow that satisfies all constraints

Finding an arbitrary flow

We make the following changes in the network. We add a new source s' and a new sink t', a new edge from the source s' to every other vertex, a new edge for every vertex to the sink t', and one edge from t to s. Additionally we define the new capacity function c' as:

- $c'((s', v)) = \sum_{u \in V} d((u, v))$ for each edge (s', v).
- $c'((v, t')) = \sum_{w \in V} d((v, w))$ for each edge (v, t').
- $c'((u, v)) = c((u, v)) - d((u, v))$ for each edge (u, v) in the old network.
- $c'((t, s)) = \infty$

If the new network has a saturating flow (a flow where each edge outgoing from s' is completely filled, which is equivalent to every edge incoming to t' is completely filled), then the network with demands has a valid flow, and the actual flow can be easily reconstructed from the new network. Otherwise there doesn't exist a flow that satisfies all conditions. Since a saturating flow has to be a maximum flow, it can be found by any maximum flow algorithm, like the Edmonds-Karp algorithm or the Push-relabel algorithm.

The correctness of these transformations is more difficult to understand. We can think of it in the following way: Each edge $e = (u, v)$ with $d(e) > 0$ is originally replaced by two edges: one with the capacity $d(i)$, and the other with $c(i) - d(i)$. We want to find a flow that saturates the first edge (i.e. the flow along this edge must be equal to its capacity). The second edge is less important - the flow along it can be anything, assuming that it doesn't exceed its capacity. Consider each edge that has to be saturated, and we perform the following operation: we draw the edge from the new source s' to its end v, draw the edge from its start u to the new sink t', remove the edge itself, and from the old sink t to the old source s we draw an edge of infinite capacity. By these actions we simulate the fact that this edge is saturated - from v there will be an additionally $d(e)$ flow outgoing (we simulate it with a new source that feeds the right amount of flow to v), and u will also push $d(e)$ additional flow (but instead along the old edge, this flow will go directly to the new sink t'). A flow with the value $d(e)$, that originally flowed along the path $s - \cdots - u - v - \ldots t$ can now take the new path $s' - v - \cdots - t - s - \cdots - u - t'$. The only thing that got simplified in the definition of the new network, is that if procedure created multiple edges between the same pair of vertices, then they are combined to one single edge with the summed capacity.

Minimal flow

Note that along the edge (t, s) (from the old sink to the old source) with the capacity ∞ flows the entire flow of the corresponding old network. I.e. the capacity of this edge effects the flow value of the old network. By giving this edge a sufficient large capacity (i.e. ∞), the flow of the old network is unlimited. By limiting this edge by smaller capacities, the flow value will decrease. However if we limit this edge by a too small value, than the network will not have a saturated solution, e.g. the corresponding solution for the original network will not satisfy the demand of the edges. Obviously here can use a binary search to find the lowest value with which all constraints are still satisfied. This gives the minimal flow of the original network.

Contributors:
jakobkogler (89.83%) adamant-pwn (10.17%)