Network flows revisited

Combinatorial Optimization – Group K

Class 22

Spring 2023

The maximum flow problem

Previously, we defined the maximum network flow problem: given a directed graph G, source and target nodes $s, t \in V(G)$ and a capacity function $c\colon E \to \mathbb{R}^+$, we look for an assignment $f\colon E \to \mathbb{R}^+$ of flow values to the arcs such that capacity constraints hold for each arc e, flow preservation constraints hold for each $v \in V(G) \setminus \{s,t\}$ and the overall flow value m_f is maximized.

Reconsidering the above problem in light of linear programming, it is straightforward that each maximum flow problem is nothing but a linear program: a variable $x(e) = f(e)$ is assigned to each arc e, capacity and flow preservation constraints impose linear inequalities and equations on the variables, respectively, and a linear objective function (a signed sum of certain variables) is to be maximized. In more detail, assuming that all input data (that is, G, s, t, and c) are given, the maximum flow problem can be described by the following LP problem.

$$
\begin{array}{lllcl}
 & \text{max: } \sum\left\{x(e)\colon e \text{ leaves } s\right\} - \sum\left\{x(e)\colon e \text{ enters } s\right\} & & & \\
 & \text{subject to} & & & \\
(1) & \forall v \in V(G)\setminus\{s,t\}\colon & \sum\left\{x(e)\colon e \text{ leaves } v\right\} - \sum\left\{x(e)\colon e \text{ enters } v\right\} & = & 0 \\
(2) & \qquad \forall e \in E(G)\colon & x(e) & \le & c(e) \\
(3) & \qquad \forall e \in E(G)\colon & x(e) & \ge & 0
\end{array}
$$

Consequently, each maximum flow problem could also be solved by any LP solver. Although the augmenting path algorithm provides an even more efficient approach, viewing network flow problems as linear programs is not in vain: this way we can handle more complex types of network flow problems not solvable by (the previously seen version of) the augmenting path algorithm.

We mention that viewing the maximum flow problem as a linear programming problem also offers the possibility of proving the Ford–Fulkerson theorem by applying the duality theorem on it. This is indeed doable, but we omit the details here due to lack of space and time.

Incidence matrix

Although we skip the details of applying the duality theorem on the maximum flow problem, it is still worth taking just the very first step towards this direction: considering the matrix form of the above LP problem, since this will be relied on later.

To this end, let $V(G) = \{v_1, v_2, \ldots, v_{n-2}, v_{n-1} = s, v_n = t\}$ and $E(G) = \{e_1, e_2, \ldots, e_m\}$. Then we can collect all variables in the column vector x such that $x_j = x(e_j)$ holds for all $1 \le j \le m$. The constraints $0 \le x(e) \le c(e)$ are obviously very simple ones, the corresponding rows of the coefficient matrix only contain a single non-zero entry (a 1 or a -1). Consequently, it is the system of linear equations marked as (1) above that yields the more complex part of the coefficient matrix. Since every equation here obviously contributes two rows to the matrix form which are negatives of each other, let us consider the system obtained from (1) with $=$ signs replaced by $\le$. Then every vertex from v_1 to v_{n-2} contributes a row to the coefficient matrix; furthermore, the coefficient of every variable is 1 or -1 or 0 in every inequality. In more detail, in the row corresponding to the vertex v_i, the coefficient corresponding to the variable $x_j = x(e_j)$ is 1 if e_j leaves v_i; the coefficient is -1 if e_j enters v_i; and it is 0 if e_j is not incident to v_i. This gives rise to the following definition.

Definition. Assume that G is a loopless directed graph with vertex set $V(G) = \{v_1, v_2, \ldots, v_n\}$ and arc set $E(G) = \{e_1, e_2, \ldots, e_m\}$. The *incidence matrix* $B(G)$ of G is an $n \times m$ matrix such that for every $1 \leq i \leq n$ and $1 \leq j \leq m$

$$\big[B(G)\big]_{i,j} = \begin{cases} 1 & \text{if } e_j \text{ leaves } v_i; \\ -1 & \text{if } e_j \text{ enters } v_i; \\ 0 & \text{if } e_j \text{ is not incident to } v_i. \end{cases}$$

The minimum cost flow problem

If we think of the network given by G, $s, t \in V(G)$ and $c\colon E \to \mathbb{R}^+$ as a road network, it is a natural assumption that carrying a unit of flow along an arc e has a fixed nonnegative cost $k(e)$. (Think of fuel costs, highway toll, wage of the driver, etc.) It is also a natural assumption that there is a required amount of flow M to be carried from s to t (where M is also part of the input). Then it is just sensible to look for a flow f of value m_f at least M with minimum total cost $\sum_{e \in E(G)} k(e)f(e)$. This problem, the *minimum cost flow problem*, comes up in many real-life applications.

It is an obvious observation that the minimum cost flow problem is again nothing but a linear programming problem. In particular, the LP formulation of the maximum flow problem we gave above can easily be modified to correspond to the minimum cost flow problem.

$$\begin{array}{llrcl} & \text{min: } \sum_{e \in E(G)} k(e)x(e) & & & \\ & \text{subject to} & & & \\ (1) & \forall v \in V(G) \setminus \{s,t\}\colon & \sum \big\{x(e)\colon e \text{ leaves } v\big\} - \sum \big\{x(e)\colon e \text{ enters } v\big\} & = & 0 \\ (2) & & \sum \big\{x(e)\colon e \text{ leaves s}\big\} - \sum \big\{x(e)\colon e \text{ enters } s\big\} & \geq & M \\ (3) & \forall e \in E(G)\colon & x(e) & \leq & c(e) \\ (4) & \forall e \in E(G)\colon & x(e) & \geq & 0 \end{array}$$

Since the minimum cost flow problem is a special linear programming problem, it can be solved efficiently with any LP solver. However, similarly to the case of the maximum flow problem, there also exist even more efficient algorithms for the minimum cost flow problem that do not rely on linear programming.

The multicommodity flow problem

In many practical applications of network flows, the same network is used for carrying not just one, but many different commodities. Each type of commodity has its own respective source node and target node, however, all commodities contribute jointly to the load of each arc.

For a precise formulation of the *k-commodity flow problem*, assume that a directed graph G is given together with k pairs of vertices $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$ and a capacity function $c\colon E \to \mathbb{R}^+$. A solution of the problem consists of assigning k flow values $x_1(e), x_2(e), \ldots, x_k(e)$ to each arc e such that flow preservation constraints $\sum \big\{x_i(e)\colon e \text{ leaves } v\big\} = \sum \big\{x_i(e)\colon e \text{ enters } v\big\}$ hold for each $1 \leq i \leq k$ and $v \in V(G) \setminus \{s_i, t_i\}$. As to the objective function, many equivalent formulations exist – let's accept that the sum of the values of the k flows $\sum_{i=1}^{k} m_{x_i}$ is maximized, where $m_{x_i} = \sum \big\{x_i(e)\colon e \text{ leaves } s_i\big\} - \sum \big\{x_i(e)\colon \text{e enters } s_i\big\}$. All in all, the LP formulation of the multicommodity flow problem is the following.

$$\begin{array}{llrcl} & \text{max: } \sum_{i=1}^{k} \Big(\sum \big\{x_i(e)\colon e \text{ leaves } s_i\big\} - \sum \big\{x_i(e)\colon e \text{ enters } s_i\big\}\Big) & & & \\ & \text{subject to} & & & \\ (1) & \forall i \in \{1,2,\ldots,k\},\ \forall v \in V(G) \setminus \{s,t\}\colon & \sum \big\{x(e)\colon e \text{ leaves } v\big\} - \sum \big\{x(e)\colon e \text{ enters } v\big\} & = & 0 \\ (2) & \forall e \in E(G)\colon & x_1(e) + x_2(e) + \ldots + x_k(e) & \leq & c(e) \\ (3) & \forall i \in \{1,2,\ldots,k\},\ \forall e \in E(G)\colon & x(e) & \geq & 0 \end{array}$$

Being a linear programming problem, the multicommodity flow problem is again solvable efficiently (even in polynomial time). However, as opposed to the previously mentioned flow problems, no algorithm is known for the k-commodity flow problem that avoids linear programming if $k \geq 2$. On the other hand, there exist algorithms within LP theory that exploit the specialities of the multicommodity flow problem and provide better running times than general LP solvers.