Week 4

1 Network Flow

Chapter 7 of the book is about optimisation problems on networks. Section 7.1 gives a quick introduction to the definitions of graph theory. In fact I hope these are already known to everyone, but I will briefly recall them whenever necessary. Please alert immediately if I use a term that you don't know. All terms in graph theory are usually simple and indeed very intuitive. They can be explained in a few seconds. So don't hesitate to ask!

We are given a *network* $G = (\mathcal{N}, \mathcal{A})$. This is a *directed graph*. A (undirected) *graph* is an object defined by a set of *vertices* or *nodes* and a set of *edges*, which are pairs of nodes.[1] We will write an edge as a set $\{i, j\}$. In a directed graph, the edges have a direction, and are called *arcs*. The order of the two nodes in the notation is relevant: (i, j) means an arc with i its *tail* and j its *head*.

We are ready for the most important definition of this part of the course, which I will give in a mathematical formula first. We define a *flow* in a network $G = (\mathcal{N}, \mathcal{A})$ as any feasible solution of the following set of linear constraints

$$\begin{array}{l} \sum_{(i,j)\in\mathcal{A}} f_{ij} - \sum_{(j,i)\in\mathcal{A}} f_{ji} = b_i, \ \forall i \in \mathcal{N} \\ 0 \le f_{ij} \le u_{ij}. \end{array} \tag{1}$$

The interpretation of f_{ij} is indeed the size of a stream of some commodity that "flows" through the arc (i, j). We always assume that it is non-negative and usually there is a capacity on the flow dependent on the arc, here u_{ij}. $b_i > 0$ signifies that node i supplies b_i of the commodity; we call node i then a *source*. $b_i < 0$ signifies that node i demands b_i of the commodity; we call node i then a *sink*. $b_i = 0$ signifies that the node is used only as transfer point. A little thought should make it clear that there are feasible flows only if $\sum_{i\in\mathcal{N}} b_i = 0$. An example is given in the accompanying powerpoint file.

In matrix notation, a network flow is any vector $f \in \mathbb{R}^{|\mathcal{A}|}$ satisfying

$$\begin{array}{l} Af = b, \\ \underline{0} \le f \le u. \end{array}$$

The matrix A is called the *node-arc incidence matrix*; a node is *incident* to an arc (and vice versa) if it is either head or tail of the arc. The rows of A correspond to nodes and the columns to arcs. The column corresponding to arc (i, j) has 1 in row i, the tail of the arc, and -1 in row j, the head of the arc, and all its other entries are 0.

[1] I will draw figures at the blackboard. Unfortunately, I am so bad in making figures that it costs me too much time to insert them in these written notes.

Let us consider the optimisation problem. Let c_{ij} be the cost of sending one unit of flow through arc (i, j).

$$\begin{array}{ll} \min & c^T f \\ \text{s.t.} & Af = b \\ & 0 \leq f \leq u. \end{array} \tag{2}$$

Again, for an example I refer to the ppt-file. This is the *min-cost flow problem* or capacitated transhipment problem. A wide variety of optimisation problems can be modelled as network flow problem: the shortest path problem, the maximum flow problem, the transportation problem, the assignment problem etc.

Any f in the null-space of A, i.e., any

$$\{f \in \mathbb{R}^{|\mathcal{A}|} \mid Af = \underline{0}\}$$

is called a *circulation*. Notice that a circulation is not just a feasible solution of (2) with $b = \underline{0}$, since it may have negative coefficients and coefficients greater than the capacities.

1.1 Primal algorithms

We will present two primal algorithms, the simplex method and the negative cost cycle algorithm. In each iteration of any (primal) algorithm for problem (7) we have a feasible flow, and investigate if we can improve it.

Suppose we have feasible flow f. Let us argue that we can change this flow over any undirected cycle of the network G, i.e., any set of edges that form a cycle if we disregard directions. Let C be such a cycle in the underlying undirected graph and choose any direction for this cycle, clockwise or anti-clockwise (drawn at the blackboard). Then some arcs of the directed graph appear on the cycle in forward direction, call them the set F^C, and some in backward direction, call them the set B^C. Set

$$\begin{array}{ll} h^C_{ij} = 1, & \text{if } (i,j) \in F^C, \\ h^C_{ij} = -1, & \text{if } (i,j) \in B^C, \\ h^C_{ij} = 0, & \text{if } (i,j) \notin C. \end{array} \tag{3}$$

All is illustrated in the example in the ppt-file. It is easy to see that h^C satisfies $Ah^C = 0$ (shown in a figure on the blackboard). It is called a *simple circulation*.

Thus, given a solution f, $f + \theta h^C$ is also a solution for any $\theta \in \mathbb{R}$, provided that $0 \leq f + \theta h^C \leq u$; i.e., provided that

$$\begin{array}{ll} 0 \leq f_{ij} + \theta \leq u_{ij} & \text{if } (i,j) \in F^C, \\ 0 \leq f_{ij} - \theta \leq u_{ij} & \text{if } (i,j) \in B^C, \end{array}$$

i.e.,

$$\theta^* = \min \left\{ \min_{(i,j)\in F^C} \{u_{ij} - f_{ij}\}, \min_{(i,j)\in B^C} f_{ij} \right\}. \tag{4}$$

Given some feasible flow our algorithms will only be interested in flow exchanges that reduce total costs. Changing f with θh^C will change the cost with

$$\theta c^T h^C = \theta \left(\sum_{(i,j)\in F^C} c_{ij} - \sum_{(i,j)\in B^C} c_{ij} \right). \tag{5}$$

We call a cycle a negative cost cycle if

$$\left(\sum_{(i,j)\in F^C} c_{ij} - \sum_{(i,j)\in B^C} c_{ij} \right) < 0 \tag{6}$$

Thus we get a better solution if we have a negative cost cycle. Clearly we only strictly decrease total costs if in (4) $\theta^* > 0$, i.e. non of the forward edges on C has $f_{ij} = u_{ij}$ and non of the backward arcs on C has $f_{ij} = 0$. Given a feasible flow f, such a cycle is called an *unsaturated* cycle.

Both the simplex algorithm and the negative cost cycle algorithm of Section 7.4 in [B&T] are variations on the theme to find in each iteration a negative cost cycle C and change the present feasible flow f to $f + \theta^* h^C$. The simplex method cannot always avoid to have $\theta^* = 0$, because of degeneracy, but will always terminate in an optimal solution. The other algorithm only makes the changes on unsaturated cycles but may not terminate. Both algorithms, if they terminate, terminate with a feasible flow for which there is no negative cost cycle, which is indeed optimal, as we will show.

1.1.1 The Simplex Method for Network Flow

Let us denote $|\mathcal{N}| = n$ and $|\mathcal{A}| = m$. First notice that, since $\sum_{i\in\mathcal{N}} b_i = 0$, the rows of A must be linearly dependent. Satisfying the equalities for $n-1$ of the nodes will automatically satisfy the remaining equality. Hence $\text{rank}(A) \leq n-1$. Thus the problem remains the same if we delete the last restriction from it. Let us denote the matrix A with the last row deleted $\tilde{A}$.

Assume that G is connected, without loss of generality, otherwise we could split the problem into two or more separate problems. To facilitate the exposition of the simplex algorithm let us also assume that in (2) u is infinite, making the flow problem uncapacitated.

$$\begin{array}{ll} \min & c^T f \\ \text{s.t.} & Af = b \\ & f \geq 0. \end{array} \tag{7}$$

For the characterisation of basic (feasible) solutions we need the definition of a *tree*. It is an undirected concept. In a graph a tree is a connected subgraph without (undirected) cycles. (A subgraph of a directed graph without directed cycles is called a *directed acyclic subgraph*, a concept we do not use in this part of the course.) A tree is a *spanning tree* of a graph if it contains all nodes. A basic and extremely crucial result in graph theory says that any tree on n vertices has exactly $n-1$ edges. The following theorem characterises a basic solution, not necessarily feasible.

Theorem 1.1 (7.4 in [B&T]) *A flow is a basic solution of (7) if and only if there exists a tree T of the underlying undirected graph of G, such that $f_{ij} = 0$ for all $\{i,j\} \notin T$.*

PROOF. ($\Leftarrow$.) Let us take any spanning tree T of G. Let n be the number of the node corresponding to the last row of A. Choosing the columns of $\tilde{A}$ corresponding to the edges of T gives a square matrix B. We will show that B is non-singular, which proves the theorem since it implies that $Bf = b$ has a unique solution.

Choose n, the node corresponding to the last row of A, as the root of T and give an artificial direction on the edges in T from each node towards the root. Thus every node has only one outgoing arc in this artificial direction. Renumber nodes such that on the unique path in T from each node to the root-node n numbers are increasing. Renumber arcs by their tail-node in the artificial direction. This renumbering of the nodes effects the row- and column order of B but not its singularity or non-singularity and it makes B a lower triangular matrix with elements 1 or -1 on the diagonal. Hence, $\det(B) \in \{1,-1\}$.

($\Rightarrow$.) If there is a basic solution with a cycle C in its support, then there is a simple circulation h^C such that for ϵ small enough $f + \epsilon h^C$ has the same active constraints as f. Thus f is not a unique solution of its active constraints, hence no basic solution. □

Thus basic feasible solutions of the network flow polyhedron correspond to trees in the underlying undirected graph. This may be a crucial insight to prove the Hirsch bound for flow polytopes (assuming capacities on all arcs makes the polyhedron bounded but does not change the property).

The proof also gives rise to two interesting observations:

Remark 1. Apart from proving non-singularity, $\det(B) \in \{1,-1\}$ shows also, by application of Cramer's rule that $Bf = b$ has an integer solution for all integral vectors b. The theorem then implies that the integer flow problem can simply be solved by its LP-relaxation.

There is another proof of this fact: The matrix A is *totally unimodular*, meaning that every square submatrix of A has determinant $+1$, 0, or -1. There is a very nice induction proof of total unimodularity of A.

Remark 2. If G is connected then $\text{rank}(A) = n - 1$, since B is a rank $n-1$ submatrix of A.

The guide for improvement in the simplex method was the reduced cost corresponding to a basic feasible solution. They are particularly nicely characterised here. Suppose that we have a bfs with T the corresponding tree. Consider any non-basic variable f_{kl} and its arc (k, ℓ). This arc together with the arcs in T creates a unique cycle C. Direct C such that (k, ℓ) is a forward arc on C, i.e., $(k, \ell) \in F^C$. As argued before, increasing one unit of flow on (k, ℓ) changes the cost as given in (5) by $c^T h^C = \left(\sum_{(i,j) \in F^C} c_{ij} - \sum_{(i,j) \in B^C} c_{ij} \right)$. Hence, the reduced cost is (cf. (6))

$$\bar{c}_{k\ell} = \left(\sum_{(i,j) \in F^C} c_{ij} - \sum_{(i,j) \in B^C} c_{ij} \right).$$

Thus simplex stops as soon as there are no negative cost cycles left.

The reduced costs can be found in a more efficient way than going through all $O(m)$ cycles corresponding to the non-basic variables. Namely, through exploiting the simple structure of the dual problem of (7) without capacities

$$\begin{array}{rl} \max & p^T b \\ \text{s.t.} & p^T A \leq c^T \end{array} \tag{8}$$

The column of A corresponding to arc (i, j) has 1 in coordinate i and -1 in coordinate j and 0 everywhere else. Thus, the reduced cost for each arc (i, j) is

$$\bar{c}_{ij} = c_{ij} - (p_i - p_j), \ \forall (i, j) \in \mathcal{A}.$$

From the above it is easy to see, that if p is a feasible dual solution, i.e. $\bar{c}_{ij} \geq 0$, $\forall (i, j) \in \mathcal{A}$, then $p + \delta \underline{1}$ is also feasible. Moreover, $p^T b + \delta \underline{1}^T b = p^T b$ since $\sum_i b_i = 0$. Thus, we may assume that in any dual feasible solution $p_n = 0$. Corresponding to a primal bfs, the values of the p_i's are easily computed from the reduced cost of the basic variables being 0:

$$\bar{c}_{ij} = 0 = c_{ij} - (p_i - p_j), \ \forall (i, j) \in T. \tag{9}$$

Starting from $p_n = 0$ and going through the $n-1$ equalities of (9) in a smart way sets the values of p in $O(n)$ operations. Checking reduced costs on negativity takes $O(m)$ and updating f in case a negative cost cycle exists takes $O(n)$. The total time per iteration is therefore $O(m)$ which compares favourably to the $O(mn)$ for general LP's.

1.1.2 The Negative Cost Cycle Algorithm

Let us now consider the algorithm that makes changes only on unsaturated cycles. We turn back to the capacitated problem (7). Given a flow f to find

such a cycle is done through building a *residual graph* R_f with node set $\mathcal{N}$ and arc set $\mathcal{A}_f$. (Those of you who have studied the Max-Flow problem in some previous course will recognize this graph.) Given an arc (i,j) with flow f_{ij},

- if $f_{ij} < u_{ij}$, then $(i,j) \in \mathcal{A}_f$ with capacity $u_{ij} - f_{ij}$ and cost c_{ij};
- if $f_{ij} > 0$, then $(j,i) \in \mathcal{A}_f$ with capacity f_{ij} with cost $-c_{ij}$.

It is easy to see that
1) any *directed* cycle in R_f is unsaturated, and
2) any negative cost *directed* cycle in R_f corresponds to an unsaturated negative cost *undirected* cycle of G and vice versa, and that arcs on the cycle in R_f with negative cost correspond to backward arcs on the cycle in G and arcs on the cycle in R_f with positive cost correspond to forward arcs on the cycle in G.
3) The total cost of a *directed* cycle in R_f is the total cost (sum of costs on forward arcs minus sum of costs on backward arcs) of the undirected cycle of G.

Finding a negative cost *directed* cycle in a directed graph can be done in polynomial time.

The algorithm terminates if there is no negative cost cycle in the residual graph.

Theorem 1.2 (7.6 in [B&T]) *A feasible flow f is optimal if and only if there is no unsaturated negative cost cycle.*

PROOF. $\Rightarrow$ is trivial. For $\Leftarrow$ suppose that f is not an optimal flow. Then there is an optimal flow f^* and $f^* - f = f'$ has $c^T f' < 0$. Since $Af = Af^* = b$, f' must be a circulation, i.e., $Af' = 0$. This corresponds to a *non-negative* circulation $\tilde{f} \geq 0$ in R_f. Notice that the support of this circulation may not be a single cycle. Thus we need to argue that a simple negative cost cycle exists in the residual network.

I adopt the bad habit from the book to insert a lemma in the middle of the proof of another theorem.

Lemma 1.1 *Let $f \geq 0$ be a (non-negative) circulation. Then there exist circulations $f^1, \ldots, f^k$, involving only forward arcs and positive scalars $a_1, \ldots, a_k$, such that $f = \sum_{i=1}^{k} a_i f^i$. Furthermore, if f is integral then a_i can be chosen integral, for all i.*

PROOF. Starting in some arc with $f_{ij} > 0$, because of flow conservation, there must exist an arc (j,k) with $f_{jk} > 0$, etc., until we reach a node that we have met before. Take this cycle C^1 (having only forward arcs) with its simple circulation $h^C = f^1$ and distract from the flow on each arc of C^1 an amount of flow equal to $a_1 = \min_{(i,j)\in C^1} f_{ij}$. At least one arc will get flow 0 and the remaining flow is still a non-negative circulation. Repeat this until all flow has become 0. $\square$

Therefore $\tilde{f} = \sum_i a_i \tilde{f}^i$, with each $\tilde{f}^i$ a simple circulation only on forward arcs in the residual graph. Since $\tilde{f}$ has negative cost, at least one of the $\tilde{f}^i$ must have negative cost. Thus the original graph has an unsaturated negative cost cycle. □

Notice that if b and u are integral vectors and if we start with an integral feasible flow, then in each iteration θ^* will be integral and hence, the flow will remain integral. This shows that in this case $\theta^* \geq 1$ and hence termination of the algorithm (Theorem 7.7 in [B&T]). It also proves that given integral b and u there exists an integral optimal solution to the problem.

We continue with showing how to obtain a first basic feasible solution. This goes through solving the so-called MAX FLOW problem, a very famous problem in combinatorial optimisation.

Material of Week 4 from [B& T]

I have skipped Chapter 5.

Chapter 7, 7.1 – 7.4

Exercises of Week 4

7.2, 7.14, 7.17

Next time

The rest of Chapter 7