University Nice Côte d'Azur — Graphs
Master 2 — September-November 2019

Exercises : Flows, Linear Programming and Graphs' Applications

To be returned for **December 20th 2019**.

The goal of this homework is to learn how to compute a flow with maximum value in any network flow and to see some of its applications (and also some important properties of Linear programming).

1 Flow in graphs

A *network flow* $\mathcal{N} = (D = (V, A), s, t, c : A \to \mathbb{R}^+)$ is defined by a directed graph $D = (V, A)$, with a *source* $s \in V$, a *target* (or *sink*) $t \in V$ and a *capacity* function $c : A \to \mathbb{R}^+$ over the arcs.

A *s-t flow* in $\mathcal{N}$ is any function $f : A \to \mathbb{R}^+$ satisfying :

Capacity : $f(a) \leq c(a)$ for all $a \in A$;

Flow conservation : $\sum\limits_{w \in N^-(v)} f(wv) = \sum\limits_{w \in N^+(v)} f(vw)$ for all $v \in V \setminus \{s, t\}$.

Question 1 *Show that any network flow admits a s-t flow.*

The *value* of a flow $f : A \to \mathbb{R}^+$ is defined as $v(f) = \sum\limits_{w \in N^+(s)} f(sw)$. Let us first show that the amount of flow leaving s (which is $v(f)$ by definition) equals the amount of flow entering into t.

Question 2 *Show that $v(f) = \sum\limits_{w \in N^-(t)} f(wt)$ for any network flow $\mathcal{N} = (D, s, t, c : A \to \mathbb{R}^+)$.*

hint : sum the flow conservation constraints over all vertices of $V \setminus \{s, t\}$

The problem considered here is, given a network flow $\mathcal{N} = (D, s, t, c)$, to compute a s-t flow $f : A \to \mathbb{R}^+$ with maximum value $v(f)$. First, let us show some easy upper bound on the value of any flow in $\mathcal{N}$.

A *s-t cut* in $\mathcal{N}$ is defined as any bipartition (V_s, V_t) of V (i.e., $V_s \cup V_t = V$ and $V_s \cap V_t = \emptyset$) such that $s \in V_s$ and $t \in V_t$. The *capacity* of a s-t cut (V_s, V_t) is $\delta(V_s, V_t) = \sum\limits_{u \in V_s, v \in V_t} c(uv)$ (with the convention that, if $uv \notin A$, then $c(u, v) = 0$).

Question 3 *Let f be any s-t flow in $\mathcal{N}$ and (V_s, V_t) be any s-t cut. Show that $v(f) \leq \delta(V_s, V_t)$.*

Let v^* be the maximum value of a s-t flow in $\mathcal{N}$ and δ^* be the minimum capacity of a s-t cut. Show that $v^* \leq \delta^*$.

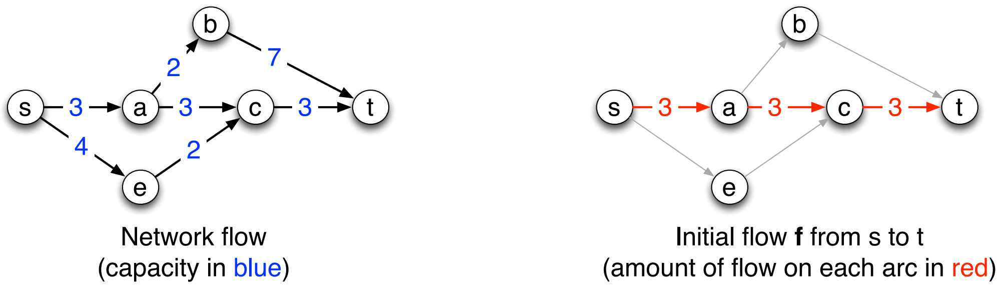
Previous question aims at showing that a minimum s-t cut in $\mathcal{N}$ is a bottleneck for a maximum s-t flow in $\mathcal{N}$. We will show a tighter relationship in what follows.

1.1 Ford-Fulkerson algorithm

Let $\mathcal{N} = (D = (V, A), s, t, c : A \to \mathbb{R}^+)$ be a flow network and let $f : A \to \mathbb{R}^+$ be a s-t flow.

The *auxiliary digraph* $\mathcal{N}_{aux}$ with respect to $(\mathcal{N}, f)$ is the digraph with auxiliary arc capacity c_{aux} defined as follows. $\mathcal{N}_{aux}$ has vertex set V and, for every $(u, v) \in V \times V$, add an arc uv with capacity $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$ in $\mathcal{N}_{aux}$.

Note that $uv \in V \times V$ may be an arc (with positive auxiliary capacity, i.e., $c_{aux}(uv) > 0$) of $\mathcal{N}_{aux}$ even if $uv \notin A$.



s
a
b
c
e
t
3
3
3
2
7
4
2
Network flow
(capacity in blue)
s
a
b
c
e
t
3
3
3
Initial flow **f** from s to t
(amount of flow on each arc in red)

FIGURE 1 – (left) Network flow $\mathcal{N}$ with arcs' capacity in blue. (right) A s-t flow f : a red number on an arc indicates the amount of flow along it. Arcs that are represented in grey have no flow.

Question 4 *Consider the network flow $\mathcal{N}$ described in Figure 1. Prove that the function $f : A \to \mathbb{R}^+$ (in red on the Figure) is a flow. Draw the auxiliary digraph $\mathcal{N}_{aux}$ with respect to $(\mathcal{N}, f)$.*

Let $\mathcal{N} = (D = (V, A), s, t, c : A \to \mathbb{R}^+)$ be a flow network, $f : A \to \mathbb{R}^+$ be a s-t flow and $\mathcal{N}_{aux}$ be the auxiliary digraph with respect to $(\mathcal{N}, f)$. Assume that there is a directed path P from s to t **in** $\mathcal{N}_{aux}$ with $\epsilon = \min\limits_{a \in A(P)} c_{aux}(a) > 0$. Let $f' : A \to \mathbb{R}$ be defined as follows :

— For every arc $a \in A \setminus A(P)$, let $f'(a) = f(a)$;
— For every arc $a \in A \cap A(P)$ with $f(a) + \epsilon \leq c(a)$, then $f'(a) = f(a) + \epsilon$;
— Else, if $a = uv \in A \cap A(P)$ and $f(a) + \epsilon > c(a)$, let $f'(a) = c(a)$ and $f'(vu) = f(vu) - (\epsilon - (c(a) - f(a)))$.

By performing the above operation, we say that f' is obtained from f by *pushing* ϵ amount of flow along the (not necessarily directed) path P in $\mathcal{N}$.

Question 5 *Prove that f' is a s-t flow in $\mathcal{N}$ with $v(f') = v(f) + \epsilon > v(f)$.*

Let us consider the following (Ford-Fulkerson) Algorithm 1.

Question 6 *Prove that if Algorithm 1 terminates, then it returns a s-t flow in $\mathcal{N}$.*

Let us consider the following pathological example described in Figure 2.

Question 7 *Consider the flow network $\mathcal{N}$ and initial flow described (in red) in Figure 2. Apply Algorithm 1 to it by, iteratively pushing flow along path (s, c, d, a, b, t), then along path (s, c, b, a, d, t), then along path (s, a, b, c, d, t) and then along path (s, a, d, c, b, t), and iteratively repeating such a sequence of pushing paths. Conclusion ?*

Algorithm 1 Ford-Fulkerson's algorithm.

```
Require: A network flow $\mathcal{N} = (D = (V, A), s, t, c : A \to \mathbb{R}^+)$ and initial $s$-$t$ flow $f_0 : A \to \mathbb{R}^+$.
Ensure: If it terminates, a $s$-$t$ flow $f' : A \to \mathbb{R}^+$ with maximum value.
1: $f \leftarrow f_0$.
2: Let $\mathcal{N}_{aux}$ be the auxiliary digraph with respect to $(\mathcal{N}, f)$.
3: while There exists a directed $s$-$t$ path $P$ in $\mathcal{N}_{aux}$ with $\epsilon = \min_{a \in A(P)} c_{aux}(a) > 0$ do
4:     Let $f'$ be obtained from $f$ by pushing $\epsilon$ amount of flow along $P$ in $\mathcal{N}$.
5:     $f \leftarrow f'$.
6:     Let $\mathcal{N}_{aux}$ be the auxiliary digraph with respect to $(\mathcal{N}, f)$.
7: return $f$
```

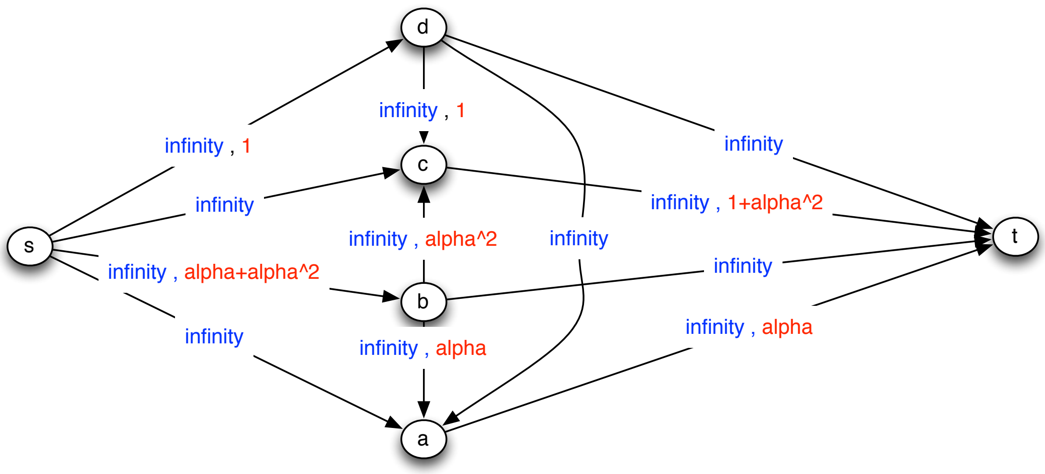


FIGURE 2 – An example of Network flow, with arc capacities in blue, and initial flow (of value $1 + \alpha + \alpha^2$) in red, with $0 < \alpha < 1$. (**Note :** *alpha* in the Figure means α and *alpha*ˆ2 in the Figure means $alpha^2$, i.e., α^2)

Question 8 *Prove that if $c : A \to \mathbb{N}$ and $f_0 : A \to \mathbb{N}$, then Algorithm 1 terminates and that its returns a function $f : A \to \mathbb{N}$ with $v(f) \in \mathbb{N}$.*

Let $\mathcal{N} = (D = (V, A), s, t, c : A \to \mathbb{R}^+)$ be a flow network and $f_0 : A \to \mathbb{R}^+$ be an initial s-t flow. Assume that Algorithm 1, applied on $\mathcal{N}$ and f_0, terminates. Let $\mathcal{N}'$ and f' be the values of $\mathcal{N}_{aux}$ and of f at the last iteration of the While-loop of Algorithm 1 before terminating. Note that Algorithm 1 returns f' and that (because of the While-loop condition), there is no directed path from s to t in $\mathcal{N}'$. Let V_s be the set of vertices v such that there is a directed path (with positive capacities) from s to v in the auxiliary digraph $\mathcal{N}'$, and let $V_t = V \setminus V_s$.

Question 9 *Prove that (V_s, V_t) is an s-t cut (in $\mathcal{N}$) with capacity $v(f')$.*

Question 10 *Deduce that, if Algorithm 1 terminates, it computes a s-t flow with maximum value.*

Question 11 *Prove the following theorem :*

Theorem 1 (Max flow-Min Cut) *In any network flow $\mathcal{N} = (D = (V, A), s, t, c : A \to \mathbb{N})$, the minimum capacity of an s-t cut equals the maximum value of a flow from s to t.*

Question 12 *Assume that $c : A \to \mathbb{N}$. Prove that the Ford-Fulkerson algorithm (Algorithm 1) computes a maximum s-t flow in time $O(f_{max}|A|)$ where f_{max} is any upper bound on the maximum value of a s-t flow in $\mathcal{N}$.*

2 Flow and Linear Programming

2.1 Duality in Linear Programming

Let $n, m \in \mathbb{N}$. Let $x_1, \cdots, x_n$ be n non-negative real variables and, for every $1 \leq i \leq n$ and $1 \leq j \leq m$, let $a_{j,i}$ and $c_i \in \mathbb{R}$ be given constants. Let us consider the following Linear Program.

$$\begin{array}{rll}
\textbf{maximize} & \displaystyle\sum_{1\leq i\leq n} c_i x_i & \\
\textbf{subject to} & & \\
(\textit{constraint } C_j :) & \displaystyle\sum_{1\leq i\leq n} a_{j,i} x_i \leq b_j & \forall 1 \leq j \leq m \\
& x_i \geq 0 & \forall 1 \leq i \leq n
\end{array}$$

The *solution* of the LP, denoted by OPT, is the optimal value of its objective function. A *feasible assignement* is any assignment $(x_1, \cdots, x_n)$ of the variables that satisfies all constraints C_j and such that $x_i \geq 0$ for all $1 \leq i \leq n$.

Question 13 *Give four simple concrete examples of LPs with 2 variables x_1 and x_2 : one with no solution (i.e., no feasible assignment), one with unbounded solution (i.e., $OPT = \infty$), one with one finite solution and infinite number of feasible assignments achieving this solution, one with one finite solution and a single feasible assignment achieving this solution.*

For each of the four required LPs, draw the feasibility domain in $\mathbb{R}^2$ and show where the optimal solution (if any) is achieved.

The goal of this section is to find a "good" upper bound on OPT.

Question 14 *Let $1 \leq j \leq m$ and let $\beta = \max_{1\leq i\leq n} |\frac{c_i}{a_{j,i}}|$. Show that, if the above LP admits a solution OPT, then $OPT \leq \beta \cdot b_j$.*

Above, we gave an upper bound on OPT by using a single constraint (C_j). To obtain a better (i.e., smaller) upper bound, let us consider a linear combination of all the constraints.

Question 15 *Let $(y_1, \cdots, y_m) \in (\mathbb{R}^+)^n$ be such that, for every $1 \leq i \leq n$, $\sum_{1\leq j\leq m} y_j a_{j,i} \geq c_i$. Show that, if the above LP admits a solution OPT, then $OPT \leq \sum_{1\leq j\leq m} y_j b_j$.*

Let us consider the following LP with variables $y_1, \cdots, y_m$ which is called the *dual* of the above LP (which is called the primal)

$$\begin{array}{rll}
\textbf{minimize} & \displaystyle\sum_{1\leq j\leq m} b_j y_j & \\
\textbf{subject to} & & \\
(\textit{constraint } C_i^* :) & \displaystyle\sum_{1\leq j\leq m} a_{j,i} y_j \geq c_i & \forall 1 \leq i \leq n \\
& y_j \geq 0 & \forall 1 \leq j \leq m
\end{array}$$

Question 16 *Show that the dual of the dual of a LP is the primal LP.*

Question 17 *Assume that the above primal LP and dual LP admit bounded solutions, respectively OPT and OPT'. Show that $OPT \leq OPT'$.*

Then, if $x^* = (x_1, \cdots, x_n)$ is a feasible assignment of the primal LP and $y^* = (y_1, \cdots, y_m)$ is a feasible assignment of the dual LP such that $\sum\limits_{1 \leq j \leq m} c_j y_j = \sum\limits_{1 \leq i \leq n} c_i x_i = OPT^*$, show that OPT^* is an optimal solution of both the primal and the dual.

In what follows, we will assume the following fundamental theorem (that can be proved, e.g., using the simplex method and previous question).

Theorem 2 (LP duality) *A primal LP has a bounded solution OPT if and only if its dual has a bounded solution OPT'. Moreover, in that case, $OPT = OPT'$.*

2.2 Max Flow - Min Cut duality

Let $\mathcal{N} = (D = (V, A), s, t, c)$ be a network flow.

Question 18 *Give a LP for solving the maximum s-t flow problem, using one variable f_a per arc $a \in A$, and one constraint per vertex.*

The minimum s-t cut problem consists in computing an s-t cut, in $\mathcal{N}$, with minimum capacity.

Question 19 *Let $F \subseteq A$ be a subset of arcs such that, for every directed path P from s to t, $A(P) \cap F \neq \emptyset$. Show that there is an s-t cut (V_s, V_t) such that $\delta(V_s, V_t) \leq \sum\limits_{a \in F} c(a)$.*

Reciprocally, show that for every s-t cut (V_s, V_t), there exists some $F \subseteq A$ intersecting every directed path P from s to t, such that $\sum\limits_{a \in F} c(a) \leq \delta(V_s, V_t)$.

Question 20 *Give a LP for solving the minimum s-t cut problem, using one variable y_a per arc and one constraint per directed path P from s to t.*

Question 21 *Give the dual LP of the previous Linear program, with one variable per directed path P from s to t and one constraint per arc.*

Question 22 *Show that the LP defined in previous question actually defines a s-t flow.*

From what precedes, we aim at proving Theorem 1 in a different way than the proof provided in Section 1.

Question 23 *Use Theorem 2 and above questions to prove Theorem 1, i.e., that the value of a maximum s-t flow equals the minimum capacity of a s-t cut.*

3 Applications of Flows in Graphs

3.1 Maximum matching in Bipartite Graphs

Let $G = (A, B)$ be a bipartite graph. Recall that a *matching* is a set of pairwise disjoint edges. Let D_G be the digraph obtained from G by orienting every edge $uv \in E(G)$ from $u \in A$ to $v \in B$. Let $\mathcal{N} = (D = (V, A), s, t, c)$ be the network flow obtained from D_G by adding to D_G one vertex s with arcs su for all $u \in A$ and one vertex t with arcs vt for all $v \in B$. Finally, let $c : A(D) \to \mathbb{R}^+$ such that $c(a) = 1$ for all $a \in A(D)$.

Question 24 *Let $k \in \mathbb{N}$. Show that there is bijection between any integral flow in $\mathcal{N}$ (i.e., flow $f : A \to \mathbb{N}$) of value k and any matching in G of size k.*

Question 25 *Deduce, from previous question, question 8 and question 12, a polynomial-time algorithm that computes a maximum matching in any bipartite graph G.*

3.2 Vertex-disjoint paths and Menger's theorem

In any (tele)communication network, it is important to ensure that several paths exist between any pair of vertices. Therefore, if some path cannot be use because of some problem/fault, another one may be used.

Let $G = (V, E)$ be an undirected graph and $s, t \in V$ be two distinct vertices. The question addressed in this section aims at finding a maximum number of internally vertex-disjoint s-t (simple) paths in G (two s-t paths P and Q are internally vertex-disjoint if $V(P) \cap V(Q) \subseteq \{s, t\}$).

Let $D = (V, A)$ be the digraph defined as follows. Let $V(D) = \{v^+, v^- \mid v \in V\}$ and $A(D) = \{v^-v^+ \mid v \in V\} \cup \{v^+w^- \mid vw \in E\}$. Let $\mathcal{N} = (D = (V, A), s, t, c)$ be the network flow obtained from D by having capacity one to every arc.

Question 26 *Let $k \in \mathbb{N}$. Show that there is a bijection between any integral flow in $\mathcal{N}$ (i.e., flow $f : A \to \mathbb{N}$) of value k and any k internally vertex disjoint s-t paths in G.*

A set of vertices $S \subseteq V \setminus \{s, t\}$ is an s-t *separator* in G if every s-t path intersects S or, equivalently, if s and t are in distinct connected components of $G \setminus S$.

Question 27 *Let $k \in \mathbb{N}$. Show that there is an s-t separator of size k in G if and only if there is a s-t cut of size k in $\mathcal{N}$.*

Question 28 *Prove the following theorem :*

Theorem 3 (Menger's theorem) *Let $G = (V, E)$ be any graph and $s, t \in V$. Then the maximum number of internally vertex disjoint s-t paths equals the minimum size of an s-t separator. Moreover, such a maximum number of internally vertex disjoint s-t paths can be computed in polynomial time.*