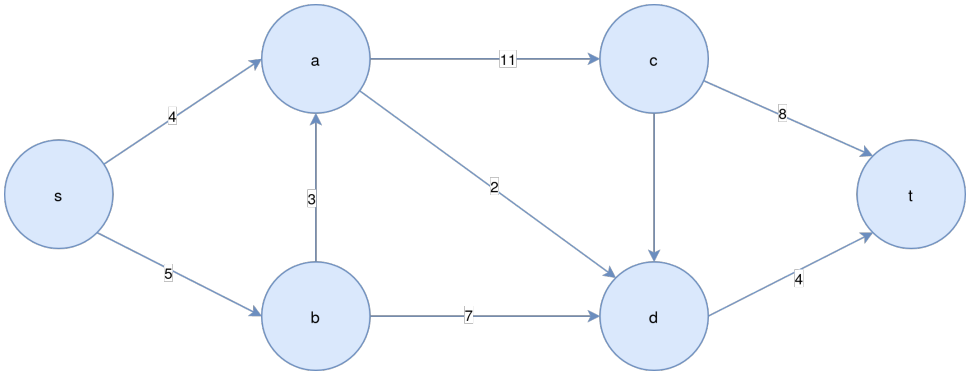
Flows, paths and cuts

Tomáš Turek



October 20, 2023

Contents

1. Introduction

Firstly we remind some basics from flows and cuts.

1.1 Network flow

Flow is defined on a **network**. Network is on a oriented graph $G = (V,E)$ and it has two special vertices $s,t \in V$ called source and target. Also we have a capacity, which is a mapping $c : E \to \mathbb{R}_0^+$. Flow then is a mapping $f : E \to \mathbb{R}_0^+$ which has two properties.

1. $\forall e \in E : f(e) \leq c(e)$
2. Kirchoff's law: $\forall v \neq s,t \in V : \sum_{uv \in E} f(uv) - \sum_{vu \in E} f(vu) = 0$

1.2 Min s,t-cut

Now we also remind ourselves another term which is a s,t-cut. Which is a $M \subset E$ such that no s,t-path exists in $G' = (V, E \setminus M)$.

These basic terms can be generalized to a **multi-commodity flow** problem and **multi-cut** problem.

2. Multi-commodity problem

On a graph $G = (V,E)$ and k tuples $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$ inV^2 known as **commodities** we can also define a network and a flow. Same as before we have a capacity $c : E \to \mathbb{R}_0^+$. These commodities are shared on the same resources.

We may define $\mathcal{P}_i$ as all paths between s_i and t_i. And also $\mathcal{P} := \bigcup_{i=1}^{k} \mathcal{P}_i$.

For a single commodity flow problem we could define an *linear program* (LP):

$$\begin{array}{ll} & \max \sum_{p \in \mathcal{P}_{st}} f_p \\ \forall e \in E & \sum_{p:e \in p \in \mathcal{P}_{st}} f_p \leq c(e) \\ & f \geq 0 \end{array}$$

From this we may define an linear program denoted as **LP1** for multi commodity problem.

$$\begin{array}{ll} & \max \sum_{p \in \mathcal{P}} f_p = \sum_{i=1}^{k} \sum_{p \in \mathcal{P}_i} f_p \\ \forall e \in E & \sum_{j=1}^{k} \sum_{p:e \in p \in \mathcal{P}_j} f_p \leq c(e) \\ & f \geq 0 \end{array}$$

Alternatively we could define it by a single variables for flows on each edge. For multi commodity problem we would have k flows which adds up to one single flow. Thus we can see that the problem is in P (polynomial).

2.1 Multi cut problem

As a cuts for single flow we may define somewhat similiar definition. Such that between every tuple of s_i and t_i there is no such path. We will also look at a linear program which will be denoted as **LP2**.

$$\forall e : x_e = \begin{cases} 1 & e \text{ in cut} \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{array}{rl} \Phi := & \min \sum_{e \in E} x(e)c(e) \\ \forall i \in [k] \forall p \in \mathcal{P}_i & \sum_{e \in p \in \mathcal{P}_i} x(e) \geq 1 \\ \text{(ILP)} & x(e) \in \{0, 1\} \\ \text{(LP)} & x(e) \geq 0 \end{array}$$

First ILP is integer linear program which is generally NP hard. Thus we will look on the relaxation of LP program. We may observe that we don't need to specify that $x(e) \geq 1$.

Now we may see that indeed **LP1** and **LP2** are dual programs. Thus resulting in knowing that the maximum flow is the same as minimum fractional cut.

2.2 Example

Before we continue we will take a look at a simple example (2.1). The graph is as follows. All capacities are equal to 1.

We may observe that the max flow is $|f| = \frac{3}{2}$ because we can put $\frac{1}{2}$ on every edge. Then multi-cut is 2 since we need to remove always at least two edges. But minimum

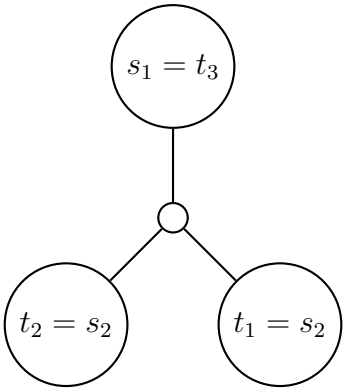


Figure 2.1: Example graph.

fractional multi-cut is only $\frac{3}{2}$ because we only have to "cut" half of each edge. Thus it is the exact same result as max flow.

2.3 Preparation for algorithm

We will show an algorithm which will give an approximate result of a multi-cut problem. We may look at it like we would cut off some parts of the graph which are close to the sources and continue.

Given $\bar{G} = (\bar{V}, \bar{E}), (s_i, t_i), k, c : E \to \mathbb{R}_0^+$ and solution x of **LP2**. We define:

$$B_x(s_i, r) := \{u \in \bar{V} | d_x(s_i, u) \leq r\}$$

As a ball around s_i with diameter w.r.t x. $d_x(s_i, u) :=$ the length of the shortest $s_i u$ path in $\bar{G}$ w.r.t. the edge length $x(e)$.

$$\delta(B_x(s_i, r)) = \{\{u,v\} \in \bar{E} : |\{u,v\} \cap B_x(s_i, r)| = 1\}$$

$$V_x(s_i, r) = \frac{\Phi}{k} + \sum_{\{u,v\} \in \bar{E}, u,v \in B_x(s_i,r)} c(e)x(e) + \sum_{\{u,v\} \in \delta(B_x(s_i,r)) : u \in B_x(s_i,r)} c(e)(r - d_x(s_i, u))$$

This we call a volume of a ball. We will denoted as a function of r $f(r)$. The last sum is of all edges which are only partly inside the ball. Next we also define the following.

$$C_x(s_i, r) = \sum_{\{u,v\} \in \delta(B_x(s_i,r))} c(u,v)$$

This will be denoted as a function $g(r)$. We may see some really nice properties these functions have. For instance $f(r)$ is a growing function which is increasing linearly and then it jumps to another point. On the other hand $g(r)$ is constant at some parts and then it jumps to a certain point, these jumps are for both function in the exact same spots. Next we can see that for some nice points it holds that $f'(r) = g(r)$.

Lemma 1. *For each $i \in [n]$ s.t. $s_i, t_i \in \bar{V}$ there exist $r \in (0, 1/2)$ s.t.*

$$\frac{C_x(s_i, r)}{V_x(s_i, r)} \leq 2 \ln 2k.$$

Proof. By contradiction. For fixed i $\forall r \in (0, 1/2)$, $\frac{f'(r)}{f(r)} > 2 \ln 2k$. We will have $r_0 = 0 < r_1 < r_2 < \cdots < r_{l-1} < 1/2 = r_l$ which are the values where $f(r)$ is not continuous (there are these "jumps"). First we consider (r_j, r_{j+1}) for some j.

$$\frac{f'(r)}{f(r)} = (\ln f(r))'$$

So $\forall r \in (r_j, r_{j+1})$, $(\ln f(r))' > 2 \ln 2k$. We will compute the integral over all of these values. We may see that the right side is just a constant so we get

$$\int_{r_j}^{r_{j+1}} (\ln f(r))' > (r_{j+1} - r_j) 2 \ln 2k.$$

In r_{j+1} there may be jump so we instead take the $\lim_{r \to r_{j+1}^-} f(r) = f^-(r_{j+1})$. Note that $f^-(r_{j+1}) \leq f(r_{j+1})$.

$$\ln f(r_{j+1}) - \ln f(r_j) \geq \ln f^-(r_{j+1}) - \ln f(r_j) = \int_{r_j}^{r_{j+1}} (\ln f(r))' > (r_{j+1} - r_j) 2 \ln 2k.$$

Now we sum our inequality over all intervals $(r_j, r_{j+1}), j = 0, \ldots, l$. We will only have the very ends because the rest will be once added and once removed.

$$\sum_{j=0}^{l-1} (\ln f(r_{j+1}) - \ln f(r_j)) > \sum_{j=0}^{l-1} (r_{j+1} - r_j) 2 \ln 2k$$

$$\ln f(r_l) - \ln f(r_0) > (r_l - r_0) 2 \ln 2k$$

$$\ln f(1/2) - \ln(0) > \ln 2k$$

$$\ln \frac{f(1/2)}{\frac{\Phi}{k}} > \ln 2k$$

$$\frac{f(1/2)}{\frac{\Phi}{k}} > 2k$$

$$f(1/2) > 2\Phi$$

As the volume of the entire pipe system is at most 2Φ it means that we have a contradiction.

□

Note that choosing $1/2$ is not necessary for the proof, but for the algorithm to work. Because if we choose $1/2$ it means that no s_j, t_j will both be in a ball for the index i. That is because the length w.r.t x of paths from s_j to t_j need to be at least 1.

2.4 Pipe cut algorithm

INPUT: $\bar{G} = (\bar{V}, \bar{E})$.
OUTPUT: F multi cut.

1: $F \leftarrow \emptyset$
2: **for** $i = 1 \dots k$ **do**
3: **if** $s_i - t_i$ are still connected in $(\bar{V}, \bar{E} \setminus F)$ **then**
4: Choose $r \in (0, 1/2)$ by Lemma.
5: $F \leftarrow F \cup \delta(B_x(s_i, r))$
6: Remove edges inside $B_x(s_i, r)$ and $\delta(B_x(s_i, r))$.
7: **end if**
8: **end for**
9: **return** F

There are few things to talk about. To get r we will check all "almost ends" of all intervals. The time complexity is polynomial since everything that is inside the code is polynomial. Correctness of the algorithm is easily observable since no pair p_i,t_i is inside some other ball and all balls will separate pairs s_j, t_j. Other thing to consider is what is the approximation ratio?

Theorem 1. *Approximation ratio of the Pipe cut algorithm is* $O(\log k)$.

Proof. Lets define C_i as the cost of the cut of the ball from iteration i and V_i as the volume of it. We know that $C_i \leq 2 \ln 2k \cdot V_i$.

$$\sum_i C_i \leq 2 \ln 2k \sum_i V_i \leq 2 \ln 2k \cdot 2\Phi = 4 \ln 2k \cdot \Phi = O(\log k)\Phi$$

□

For a single commodity we know that max flow = min cut. Where $\leq$ is trivial and $\geq$ is a little harder. This is a case of **exact duality**. On the other hand we already shown that this doesn't hold for multi-commodity, but what if we can define **approximate duality**.

Corollary. Max flow $\leq$ min cut $\leq O(\log k)$ max flow. For multi-commodity.

Proof. Because of the duality of LP1 and LP2 we know that max flow is the same as min fractional multi-cut. And because of the algorithm we know that the fractional multi-cut is in $O(\log k)$.

□

2.5 How to solve LP

There is still a problem with our LP which can have up to exponential many of constraints. But this can be solved fast by using **Ellipsoid algorithm** on $Ax \leq b$. Only think it needs is an so called **ORACLE** which is that for given $\bar{x}$, check whether $A\bar{x} \leq b$ and if not return a violated constraint.

In our case **ORACLE** is for each i find the shortest $s_i - t_i$ path w.r.t $\bar{x}$. This can be either 1 and we are happy or < 1 then this constraint is violated.

2.6 Is there any better approximation?

We will show that indeed this approximation is the best we can get. Firstly we will define a new property of graphs.

A graph $G = (V,E)$ is an **α-expander** if $\forall S \subseteq V, |S| \leq \frac{n}{2}, \delta(S) \leq \alpha|S|$.

We take as granted that it holds: 3-regular α-expanders exist for $\alpha > 0$. Now lets observe (2.2)that at most $1 + 3 \cdot 2^{l-1}$ vertices are reachable by a path of length $\leq l$.

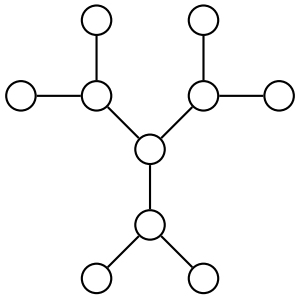


Figure 2.2: How to get the upper bound.

If we take $l = \log_2 \frac{n-2}{6} + 1$ so with the upper bound we get $1 + 3\frac{n-2}{6} = 1 + \frac{n-2}{2} = \frac{n}{2}$. And also we define an instance of multi-commodity problem: $T = \{\{u,v\}|\delta(u,v) > l\}$. A unit of flow consumes $\geq l$ units of volume of the entire system. Thus $|E| = O(n)$. Therefore max flow $\leq O(\frac{n}{l}) = O(\frac{n}{\log n})$. But for min cut we take the optimum $F \subseteq E$. Every path in $G = (V, E \setminus F)$ is $\leq \frac{n}{2}$ so min cut os $\Theta(n)$. Thus it is indeed tight.

3. The Sparsest cut problem

Same as before we have an undirected graph $G = (V,E)$ and k-pairs of sources and targets $(s_1,t_1),(s_2,t_2),\ldots,(s_k,t_k) \in V^2$. But we will introduce a new parameters $d_1,d_2,\ldots,d_k \in \mathbb{R}^+$ called **demands**.

Firstly we will take a look at linear program for solving this problem for $k = 1$.

$$\begin{array}{rc} & \max f \\ & \sum_{p\in\mathcal{P}_{st}} x_p \geq f \cdot d_1 \\ \forall e \in E & \sum_{e\in\in\mathcal{P}_{st}} x_p \leq c(e) \\ & x \geq 0 \end{array}$$

Where $\mathcal{P}_{st}$ are all paths between s and t. We may see that the optimum of the max flow is the same as this optimum just divided by d_1. We will denote $\mathcal{P}_i = \mathcal{P}_{s_i,t_i}$.

3.1 Concurrent multicommodity flow

Thus we are getting this LP for all k commodities and k demands.

$$\begin{array}{rc} & \max f \\ \forall i \in [k] & \sum_{p\in\mathcal{P}_i} x_p \geq f \cdot d_i \\ \forall e \in E & \sum_{i=0}^{k} \sum_{e\in\in\mathcal{P}_i} x_p \leq c(e) \\ & x \geq 0 \end{array}$$

We will take a look at the matrix of this LP and after that find a dual program. But firstly we modify $\sum_{p\in\mathcal{P}_i} x_p \geq f \cdot d_i$ to $f \cdot d_i - \sum_{p\in\mathcal{P}_i} x_p \leq 0$. Then the matrix is as follows:

	f		$\mathcal{P}_1$			$\mathcal{P}_2$		
1	d_1	-1	-1	$\ldots$	0	$\ldots$	0	$\ldots$
2	d_2	0	0	$\ldots$	-1	-1	$\ldots$	0
$\vdots$								
k	d_k	0	0	$\ldots$	0	0	0	$\ldots$
e_1	0	1	0		1	0	0	$\ldots$
$\vdots$								
$e_{\lvert E\rvert}$	0	0	1		1	0	1	$\ldots$

Where for the first k lines it is ≤ 0 and for edges it is $\leq c(e)$. We visualized the matrix and thus we can make the dual. We will have variables x_e for edges and y_i for $i \in [k]$. Thus the dual is:

$$\begin{array}{rc} & \min \sum_{e\in E} x_e c(e) \\ & \sum_{i=0}^{k} y_i d_i \geq 1 \\ \forall i \in [k] \forall p \in \mathcal{P}_i & \sum_{e\in p} x_e - y_i \geq 0 \\ & x,y \geq 0 \end{array}$$

Definition 1. *For $S \subseteq V$ we define $\delta(S) = \{\{u,v\} \in E : |\{u,v\} \cap S| = 1\}$ and then $I(S) = \{i \in [k] : |\{s_i,t_i\} \cap S| = 1\}$. Then the **sparsity** of S is*

$$\rho(S) = \frac{\sum_{e\in\delta(S)} c(e)}{\sum_{i\in I(S)} d_i}$$

Example. We will have a simple example where all capacities are 1 and all demands are 1. So we have the graph **??**.

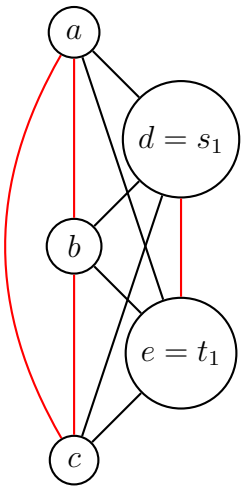


Figure 3.1: Sparse cut example.

The demands are the red edges. We may see that if we choose $S = \{c, e\}$ then $\sum_{e\in\delta(S)} c(e) = 3$ and $\sum_{i\in I(S)} d_i = 3$ therefore $\rho(S) = 1$.

We may see that each pair s_i t_i consumes at least 2 units of a flow of the network for a single unit of the flow. Then we set f as a max flow and see what we get. For example for paths $\mathcal{P}_1 = \{(d,a,e) = p_1, (d,b,e) = p_2, (d,c,e) = p_3\}$. $x_{p_1} + x_{p_2} + x_{p_3} \geq f \cdot d - 1 = f$. Thus the total volume consumed by a flow with objective value f is $\geq k2f = 8f$. Total volume of G is 6. Therefore $f \leq \frac{6}{8} = \frac{3}{4}$.

Maybe we can ask if there exist such a flow with this volume. We can obtain it by pushing $\frac{1}{4}$ from d to c on each path and $\frac{3}{8}$ between all other pairs on all paths. All edges are not over their capacities and we get $\frac{3}{4}$ for all demands.

Definition 2. *Now* $F \subseteq E : I(F) = \{i \in [k] : s_i, t_i$ *are in different components in* $(V, E \setminus F)\}$. *And* ***sparsity*** *of* F *be*

$$\rho(F) = \frac{\sum_{e\in F} c(e)}{\sum_{i\in I(F)} d_i}$$

Lemma 2. $\min_{S\subseteq V} \rho(S) = \min_{F\subseteq E} \rho(E)$.

Proof. The $\geq$ inequality can be easily seen if we set F to be $\delta(S)$. Now we need to show the other inequality. For given $F \subseteq E$, let $s_1, \dots s_l$ be the components of connectivity of $(V, E \setminus F)$. For that we will proof that $\min_{i\in[l]} \rho(s_i) \leq \rho(F)$. This will bw shown by a contradiction. Assume $\forall i$:

$$\frac{\sum_{e\in\delta(s_i)} c(e)}{\sum_{j\in I(s_i)} d_j} > \frac{\sum_{e\in F} c(e)}{\sum_{j\in I(F)} d_j}$$

$$\sum_{e\in\delta(s_i)} c(e) > \rho(F) \cdot \sum_{j\in I(s_i)} d_j$$

Now we sum all i inequalities.

$$\sum_{i=1}^{l} \sum_{e \in \delta(s_i)} c(e) > \rho(F) \cdot \sum_{i=1}^{l} \sum_{j \in I(s_i)} d_j$$

We can see that $\sum_{i=1}^{l} \sum_{e \in \delta(s_i)} c(e) = 2 \sum_{e \in F} c(e)$ because all edges are counted twice and similarly $\sum_{i=1}^{l} \sum_{j \in I(s_i)} d_j = 2 \sum_{j \in I(F)} d_j$. So we get:

$$\sum_{e \in F} c(e) > \rho(F) \cdot \sum_{j \in I(F)} d_j$$

Which is a contradiction. So for each F we can find s_i that satisfies the inequality. □

Now we can use this for integer program and then use relaxation. The program will look like this:

$$\begin{aligned} & \min \frac{\sum_{e \in E} c(e) x_e}{\sum_{i=1}^{k} d_i y_i} \\ \forall i \in [k] \forall p \in \mathcal{P}_i \quad & \sum_{e \in p} x_e \geq y_i \\ & \sum_{i=1}^{k} d_i y_i \geq 1 \\ \forall e \in E \quad & x_e \in \{0,1\} \\ \forall i \in [k] \quad & y_i \in \{0,1\} \end{aligned}$$

At least one edge has to be removed from each path. Plus we assume that $d_i \geq 1$. Now we could just put $x_e \geq 0$ and $y_i \geq 0$. But the thing is that we don't have a linear function in the objective function. What if we have a vector $(x,y) \to (\alpha x, \alpha y)$ for $\alpha > 0$. You can see that the feasible solution don't change and also the objective is the same. So we could put $\alpha = \frac{1}{\sum_{i=1}^{k} d_i y_i}$ and we know that the $\sum_{i=1}^{k} d_i y_i = 1$. Thus the linear program will be:

$$\begin{aligned} & \min \sum_{e \in E} c(e) x_e \\ \forall i \in [k] \forall p \in \mathcal{P}_i \quad & \sum_{e \in p} x_e \geq y_i \\ & \sum_{i=1}^{k} d_i y_i = 1 \\ \forall e \in E \quad & x_e \geq 0 \\ \forall i \in [k] \quad & y_i \geq 0 \end{aligned}$$