Notes on connected cuts

Tomáš Turek

December 11, 2024

Contents

1 Preliminaries **1**

2 Connected cuts definitions **1**
2.1 Single cuts . . . 1
2.2 Multi commodity cuts . . . 2
2.3 Other cuts . . . 2

3 Absorptive flow **3**
3.1 Induced cut by the absorptive flow . . . 3

4 Integer program **3**
4.1 Variables . . . 3
4.2 Constraints . . . 3
4.3 Optimization function . . . 4
4.4 The whole formulation . . . 4
4.5 Properties . . . 4
4.6 Example (which is not correct anymore) . . . 4
4.7 Update . . . 5

5 Links **6**

1 Preliminaries

We define graph $G = (V, E)$ as usual. Then we talk about edges defined by a cut in the following way. For vertices $S \subseteq V$ we define $E(S, V \setminus S) = \{e \in E \text{ s.t. } |e \cap S| = 1\}$, then the size of a cut is $e(S, V \setminus S) = |E(S, V \setminus S)|$. Also we will talk about the induced subgraphs which will be denoted as $G[S]$ for some vertices $S \subseteq V$. Just to recall that graph is called connected if in between every pair of vertices exist a walk. Also in most cases we will be considering graphs which are connected, but sometimes this is not necessary.

2 Connected cuts definitions

We will now proceed to some definitions of cuts which are in one way or another connected. We will start simply and build on that.

2.1 Single cuts

Definition 1 (Connected cut)**.** *For a connected graph $G = (V, E)$ we define connected cut as $S \subseteq V$ for which $G[S]$ is connected. The cut itself is $E(S, V \setminus S)$. Later on we may exchange if we will talk about vertices or edges.*

Now we will like to minimize the size of the cut, i.e. the value of $e(S, V \setminus S)$. Sometimes we may even define a connected cut with specific source vertex. This can be formulated by the next definition 2.

Definition 2 (Connected s cut)**.** *For a connected graph $G = (V, E)$ and given vertex $s \in V$ we define connected cut as $S \subseteq V$ for which $G[S]$ is connected and also $s \in S$.*

This is pretty much the same problem as in previous definition 1. Note that we would also like to minimize the size of the cut, i.e. $e(S, V \setminus S)$. And if we can solve it for the connected s cut we may also apply it for all $s \in V$ to get the value for general connected cut.

Some may already know that commonly used cut is for defined source and target distinct vertices. We could also use it in our case and only extend the previous definition by saying that $t \notin S$. It is somewhat tempting to also require that $G[V \setminus S]$ is supposed to be also connected. Therefore we can get the full connected $s - t$ cut.

Definition 3 (Connected $s - t$ cut)**.** *For a connected graph $G = (V, E)$ and given two distinct vertices $s, v \in V$ we define connected $s - t$ cut as $S \subseteq V$ for which all following properties hold.*

1. *$s \in S$ and $t \notin S$.*
2. *Both $G[S]$ and $G[V \setminus S]$ are connected.*

2.2 Multi commodity cuts

Now we can furthermore generalize the notion of connected cuts to multi-way connected cuts.

Definition 4 (Multi-way connected cut)**.** *For a connected graph $G = (V, E)$ and pairwise distinct vertices $s_1, s_2, \ldots, s_k \in V$ for $k \in \mathbb{N}$ we define connected cut as partition $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$ of vertices (that is $\bigcup_{i=1,\ldots,k} V_i = V$ and for $i \neq j$ $V_i \cap V_j = \emptyset$) such that the following holds:*

1. *$\forall i \in [k] : s_i \in V_i$ and*
2. *$\forall i \in [k] : G[V_i]$ is connected.*

In this specific definitions we may look at our problem from two perspective. Those two options can be seen by the optimization function for the given problem. Sum version is generally more easy to find the solution, or at least very good approximation. On the other hand optimizing over the max function can be way more tricky. Observe that the sum size is already computed with multi-commodity cut.

- *Sum* size as $\sum_{i<j} E(V_i, V_j)$.
- *Max* size as $\max_{i \in [k]} E(V_i, V \setminus V_i)$.

Also we may define **Flexible multi-way connected cut** as relaxing the previous problem. That is the partition will have l partitions where $0 < l \leq k$ and only l sources are representing their partition. So $\forall i \in [l], \exists k : s_k \in V_l$.

2.3 Other cuts

Now we can even further increase the number of requirements. In this case to the size of $|S|$. Now we will also state what is the optimization function and hence declare a problem.

Problem 1 (k-connected cut)**.** *For a connected graph $G = (V, E)$ we say $S \subseteq V$ is k connected cut such that all properties hold:*

1. *$|S| = k$.*
2. *$G[S]$ is connected.*
3. *And we want to minimize $e(S, V \setminus S)$.*

From algorithmic perspective we may also have given source vertex $s \in V$, but as it was stated before we may solve the general case by running $|V|$ times the algorithm for the problem with source.

Note that choosing only two properties from all three can be computed. If we skip the very first one, we may use the result from Garg, which states a linear program having all vertices as such result. Excluding the second one can be also computed via some approximation algorithm for bisection. And Overlooking the last one we just use some search, because we don't care about the size of the result.

3 Absorptive flow

We will be now talking about an absorptive flow. Firstly we will state the problem in a common sense. For a graph and a source we get a flow which flows through the graph and every time it goes through a vertex some of the flow gets absorbed into the given vertex. After that we will define a cut which is induced by such flow and later on state an integer program and its linear approximation. Now we properly state the instance.

Definition 5 (Absorptive flow)**.** *For a graph $G = (V, E)$ and a vertex $s \in V$, also called the source, and for $k \in \mathbb{N}$, such that $|V| \geq k$, we define **absorptive flow** as a tuple of functions denoted as (f_V, f_E), where $f_V : V \to \mathbb{R}$ and $f_E : E \to \mathbb{R}$. Now these two function must have these properties.*

1. $\sum_{v \in V} f_V(v) = k$*, that is every part of the flow gets absorbed,*
2. $f_V(s) = 1$*,*
3. $\forall v \in V : 0 \leq f_V(v) \leq 1$*, so all vertices have some limits,*
4. $\sum_{v \in V,(s,v) \in E} f_E(s, v) = k - 1$*, the flow starts from the source,*
5. $\forall e \in E : 0 \leq f_E(e)$*, the flow has to be non-negative, but can be unlimited,*
6. $\forall v \in V \setminus \{s\} : \sum_{u \in V,(u,v) \in E} f_E((u, v)) = \sum_{u \in V,(v,u) \in E} f_E((v, u)) + f_V(v)$*, thus the whole flow continue unless part of it is absorbed.*

One can already see that it resembles a linear program. Some can expect we would define a size of the flow, but in this special instance we won't be defining it, since the main purpose is to look at the cut, which is defined by the flow. So now we will define the cut.

3.1 Induced cut by the absorptive flow

Firstly we will define $S \subseteq V$ as the vertices which have nonzero function f_V, that is $\forall v \in S : f_V(s) > 0$. Then the **induced cut** defined by absorptive flow is defined as $E(S, V \setminus S)$ and its size as $e(S, V \setminus S)$. We will furthermore want to minimize the size of such cut.

So far the only property is that $s \in S$, which can be seen only from the definition. Next observations come from the linear program and its properties.

4 Integer program

In this section we will establish the linear program which works with this flow and its cut.

4.1 Variables

Firstly we declare the variables for edges and for vertices.

$$f_v = \begin{cases} 1 & \text{if it absorbs the flow} \\ 0 & \text{otherwise} \end{cases}$$

$f_{uv} \in [0, k]$ is for the amount of flow on the edge uv.

$$x_{uv} = \begin{cases} 1 & \text{if } uv \in E(S, V \setminus S) \\ 0 & \text{otherwise} \end{cases}$$

See that these variables arise only from the definition of the problem.

4.2 Constraints

Now we need to state the constraints. Firstly set the connection between the absorbed vertices and the cut.

$$\begin{aligned} x_{uv} &\geq f_u - f_v \quad \forall \{uv\} \in E \\ x_{uv} &\geq f_v - f_u \quad \forall \{uv\} \in E \end{aligned}$$

Which is basically that $x_{uv} \geq |f_u - f_v|$. Next we have to set the flow, so its properties hold.

$$\begin{aligned}
\textstyle\sum_{v\in V, sv\in E} f_{sv} &= k-1 \\
f_s &= 1 \\
\textstyle\sum_{u\in V, uv\in E} f_{uv} = \sum_{u\in V, vu\in E} f_{vu} + f_v &\quad \forall v \in V, s \neq v \\
\textstyle\sum_{u\in V} f_u &= k \\
f_v \geq \frac{1}{k-1} \textstyle\sum_{u\in V, \{uv\}\in E} f_{uv} &\quad \forall v \in V \setminus \{s\}
\end{aligned}$$

4.3 Optimization function

Lastly the optimization function will be to minimize the flow throughput and number of cut edges.

$$\min \sum_{e\in E} x_e$$

4.4 The whole formulation

$$\begin{aligned}
\min \textstyle\sum_{e\in E} x_e & \\
x_{uv} \geq f_u - f_v &\quad \forall \{uv\} \in E \\
x_{uv} \geq f_v - f_u &\quad \forall \{uv\} \in E \\
\textstyle\sum_{v\in V, sv\in E} f_{sv} &= k-1 \\
f_s &= 1 \\
\textstyle\sum_{u\in V, uv\in E} f_{uv} = \sum_{u\in V, vu\in E} f_{vu} + f_v &\quad \forall v \in V, s \neq v \\
\textstyle\sum_{u\in V} f_u &= k \\
f_v \geq \frac{1}{k-1} \textstyle\sum_{u\in V, \{uv\}\in E} f_{uv} &\quad \forall v \in V \setminus \{s\} \\
f_v \in \{0,1\} &\quad \forall v \in V \\
f_{uv} \in \mathbb{R}^+ &\quad \forall \{u,v\} \in E \\
x_{uv} \in \{0,1\} &\quad \forall \{u,v\} \in E
\end{aligned} \tag{1}$$

4.5 Properties

Lets talk about some crucial properties of this integer program.

Observation 1. *Every vertex in the flow absorb.*

Proof. See that due to the last constraint whenever a flow goes inside the vertex we must set the vertex to absorb some portion of the flow. Exactly 1 in the case of integer program. □

Observation 2. *f_V defines a connected induced subgraph of G.*

Proof. Since $f_s = 1$ we know that s is inside the $G[S]$. For contradiction assume there is a vertex $v \in S$ which is not connected to the source s. Because $f_v = 1$ we must have that there is a flow over this vertex due to the fifth constraint. And since there is a flow to the vertex v which starts in s there is also a walk from s to v, therefore no such vertex v exists and all vertices are connected to s and hence they induce connected subgraph. □

Observation 3. *We have a minimum connected s cut from the optimal integer value of the lp formulation x^*.*

Now it is also a time to talk about integrality, moreover what will happen if we switch to lp relaxation.

4.6 Example (which is not correct anymore)

Now consider a simple example of a graph G which is depicted on picture 1. It is a so called star, where v is the center point with Δ neighbors. In this special instance s is already provided, now we need to set $k = 2$.

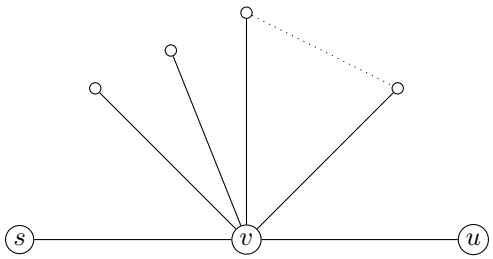

Figure 1: Star graph G with special vertices v, s and u.

We can easily observe that the optimal solution of such connected s cut is by setting $S = \{s, v\}$ since if $s \in S$ and $G[S]$ has to be connected we can only choose v, so that also $|S| = k = 2$. But mainly we are interested in observing how will the integer or linear relaxation of our program work on such instance.

Observation 4. *Integer linear program will find the optimal solution.*

Proof. See that $f(s) = 1$ and $f(sv) = 1$ thus $f(v) \geq \frac{1}{\Delta(2-1)}1 = \frac{1}{\Delta}$. From the property that we have integer variables $f(v)$ must be 1. And hence we absorbed the whole flow and satisfied all constraints. Also the solution can be seen on the picture 2. □

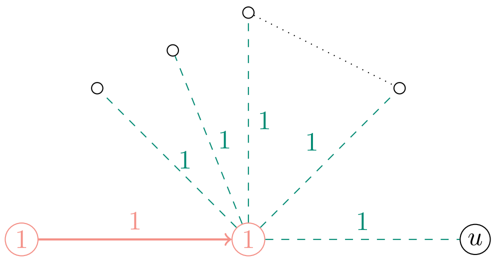

Figure 2: Absorbing flow of the integer program.

Observation 5. *Linear relaxation will provide a solution with $S = V$ and $x_e > 0$ only for edge sv.*

Proof. The starting point is the same as in previous case, that is $f(s) = 1$ and $f(sv) = 1$. Then we have that $f(v) \geq \frac{1}{\Delta}$. Now suppose that all $N[v] \setminus s$ will have the same value of f. So that the cut edge will be only one sv.

Firstly we must see that this is indeed possible. What if we set $f(vx) = \frac{1}{\Delta}$ for all such neighbors $x \in N[v] \setminus s$, and also $f(x) = \frac{1}{\Delta}$. We have to check if such solution is indeed feasible. Lets sum all

$$\sum_{v \in V} f(v) = f(s) + \sum_{v \in V \setminus s} f(v) = 1 + \sum_{v \in V \setminus s} \frac{1}{\Delta} = 1 + (1 + \Delta - 1)\frac{1}{\Delta} = 1 + \frac{\Delta}{\Delta} = 2$$

Therefore we have satisfied the constrained of k chosen vertices. Also we can see that the other properties still indeed hold, which are not so hard. For example the flow follow its restrictions. And we have the found the solution given in this observation and shown that it is feasible. The solution is shown on the picture 3. Now the value of the optimization function is $1 - \frac{1}{\Delta} = \frac{\Delta - 1}{\Delta}$, which $\frac{1}{\Delta}$ times less then the OPT.

Lastly we have to see that this is indeed optimal. What can be changed? Only the value of $f(v)$ and flows from vertex v. Suppose we would move λ back from u to v. This will increase $f(v)$ to $\frac{1}{\Delta} + \lambda$ and decreased both $f(u)$ and $f(vu)$ by λ. This will decrease $x(sv)$ by λ and increased $f(vx)$ by λ for all neighbors of v without u, s. Because for $x(vu)$ we will now have 2λ so in total we haven't made a better solution. We would need to split λ amongst all $\Delta - 2$ such neighbors. But then we will again increase $x(sv)$, so we are in a dead end. All other cases are similar. □

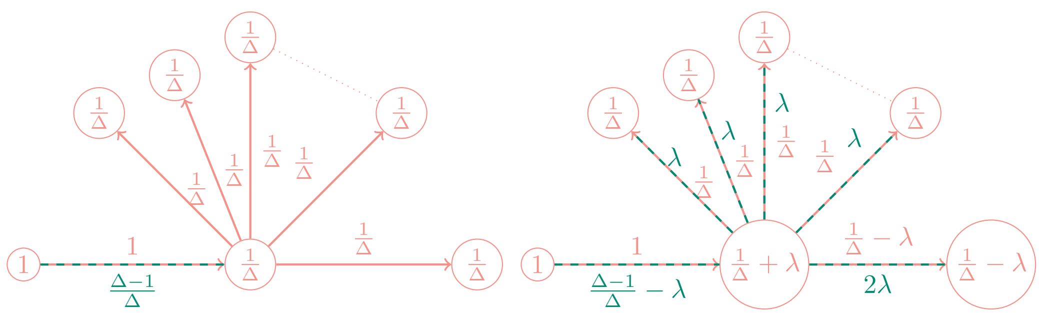

Figure 3: Absorbing flow of the linear relaxation.

4.7 Update

In this subsection we will provide even better model, which arises from the previously mentioned one. This time we will also use the notation $d(u, v)$ for shortest path between vertices u and v. This is a well known

property of graphs, which can be computed in polynomial time (e.g. using Dijsktra's algorithm). Therefore we will increase what we want the vertex absorb by setting the fraction $\frac{1}{k-1}$ to $\frac{1}{k-d(s,u)}$ for every u. That is we combine two different metrics of path length. One in terms of standard length and the other of terms flow. The linear program is as follows.

$$\begin{aligned}
\min \textstyle\sum_{e\in E} x_e & \\
x_{uv} \geq f_u - f_v & \quad \forall \{uv\} \in E \\
x_{uv} \geq f_v - f_u & \quad \forall \{uv\} \in E \\
\textstyle\sum_{v\in V, sv\in E} f_{sv} = k-1 & \\
f_s = 1 & \\
\textstyle\sum_{u\in V, uv\in E} f_{uv} = \sum_{u\in V, vu\in E} f_{vu} + f_v & \quad \forall v \in V, s \neq v \\
\textstyle\sum_{u\in V} f_u = k & \\
f_v \geq \textstyle\sum_{u\in V, \{uv\}\in E} \frac{1}{k-d(s,v)} f_{uv} & \quad \forall v \in V \setminus \{s\} \\
f_v \in \{0,1\} & \quad \forall v \in V \\
f_{uv} \in \mathbb{R}^+ & \quad \forall \{u,v\} \in E \\
x_{uv} \in \{0,1\} & \quad \forall \{u,v\} \in E
\end{aligned} \tag{2}$$

5 Links

Here I am keeping some useful links.

1. Approximating unique games
2. Minimum Bisection
3. Min max and small set expansion
4. Approximation algorithms for maximally balanced connected graph partition
5. On size-constrained minimum s–t cut problems and size-constrained dense subgraph problems
6. On the minimum $s-t$ cut problem with budget constraints
7. Balanced Crown Decomposition for Connectivity Constraints