Notes on connected cuts

Tomáš Turek

April 4, 2025

Contents

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 between every pair of vertices there exists a walk. Also in most cases we will be considering graphs which are connected, but sometimes this is not necessary. Also we will be using commonly use notion for number of vertices $n = |V|$ and for edges as $m = |E|$. The neighborhood of a vertex is denoted as $N(v)$ and closed neighborhood as $N[v]$.

One can already know some basic algorithmic aspects of some cut related problems. For example finding a minimal cut in a graph can be done using flow algorithms. As an exercise the reader may see that this way we actually obtain a subset $S \subseteq V$ which will be indeed connected, i.e. the induced subgraph $G[S]$.

Another well known problem is a maximal cut, which is known to be NP hard. But on the other hand there exists an $0,878\ldots$-approximation algorithm.

Lastly we mention that $[n]$ is an abbreviation for a set $\{1, 2, \ldots, n\}$. Also the notion $[a, b]$ is for a set $\{a, a+1, \ldots, b\}$.

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 (BFS or DFS), because we don't care about the size of the result.

2 Connected cuts definitions

We will now proceed to some definitions of cuts which are in one way or another connected (meaning that they induce a connected subgraph). We will start simply and build on that.

2.1 Single commodity connected 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. Also a graph $(V, E \setminus E(S, V \setminus S))$ is a disconnected graph.*

Now we will like to minimize the size of the cut, i.e. the value of $e(S, V \setminus S)$. Also note that the requirement for G being connected is actually not necessary. 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 s-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 see that the property G being connected is still not necessary.

Reader could 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 $s, t \in V$ 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 connected cuts

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

Definition 4 (Multi-way connected cut with sources)**.** *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 perspectives. 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. Though some modifications has to be done.

- *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 $\forall i \in [l], \exists k : s_k \in V_l$.

We may also present another problem which is similar to the previously mentioned one, the only change is that we do not have predefined sources.

Definition 5 (Multi-way connected cut)**.** *For a connected graph $G = (V, E)$ and $k \in \mathbb{N}$ 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. *$\forall i \in [k] : G[V_i]$ is connected.*

Then we would like to minimize the maximum $\max_{i \neq j} e(V_i, V_j)$.

2.3 Other cuts

Now we can even further increase the number of requirements. In this case to the size of $|S|$. We will also state what is the optimization function and hence establish 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. *The value $e(S, V \setminus S)$ is minimized.*

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

3 Absorptive flow and its linear program

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 boundary which is induced by such flow and later on state an integer program and its linear approximation. Now we properly state the instance.

Definition 6 (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. *$\forall v \in V : 0 \leq f_V(v) \leq 1$, all vertices are bounded,*
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.*

We will be calling function f_E a flow and f_V a vertex flow or absorption. Reader can see this flow in a following way. We will have k resources in s and we will be pushing the flow throughout the graph. The targets are dynamically created based on their absorption. In another way there we can imagine having a target t which is connected to all vertices. The demand is k for such vertex t.

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 boundary (cut), which is defined by the flow. So now we will define it. But also in that case we would like to enforce that when a flow passes through a vertex then some of the value gets absorbed. This will lead to a connected absorption of vertices.

3.1 Boundary of the absorptive flow

Firstly we will define $S \subseteq V$ as the vertices which have nonzero absorption, that is $\forall v \in S : f_V(s) > 0$. Then the **boundary** induced by an 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 boundary under the additional condition of connectivity.

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.

3.2 Integer program

In this section we will establish the linear program which works with this flow and its boundary. We will be forcing connectivity of the subgraph $G[S]$ thus the term boundary is in a sense same as cut. Also when we have a graph $G = (V, E)$ we will modify the graph by setting edges to be directed in both directions. That is from edge $\{i, j\}$ we create two arcs (i, j) and (j, i).

3.2.1 Variables

Firstly we declare the variables for edges and for vertices.

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

$$f_{uv} \in [0, k] \text{ is for the amount of flow on the edge } uv. \tag{2}$$

$$x_{uv} = \begin{cases} 1 & \text{if } uv \in E(S, V \setminus S) \text{ and} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

See that these variables arise only from the definition of the problem. There is only change of f_{uv} being limited by k, which can be easily observed to be the same as unlimited. We will be calling variables (1) as an absorption, variables (2) as flow and variables (3) as cut edges.

3.2.2 The mixed integer program

Now we will show the whole MIP formulation. And describe the constraints.

$$\min \sum_{e \in E} x_e \tag{4a}$$

$$x_{uv} \geq f_u - f_v \qquad \forall \{uv\} \in E \tag{4b}$$

$$x_{uv} \geq f_v - f_u \qquad \forall \{uv\} \in E \tag{4c}$$

$$\sum_{v \in V, sv \in E} f_{sv} = k - 1 \tag{4d}$$

$$f_s = 1 \tag{4e}$$

$$\sum_{u \in V, uv \in E} f_{uv} = \sum_{u \in V, vu \in E} f_{vu} + f_v \qquad \forall v \in V, s \neq v \tag{4f}$$

$$\sum_{u \in V} f_u = k \tag{4g}$$

$$(k-1) \cdot f_v \geq \sum_{u \in V, \{uv\} \in E} f_{uv} \qquad \forall v \in V \setminus \{s\} \tag{4h}$$

$$f_v \in \{0, 1\} \qquad \forall v \in V \tag{4i}$$

$$f_{uv} \in \mathbb{R}^+ \qquad \forall \{u, v\} \in E \tag{4j}$$

$$x_{uv} \in \{0, 1\} \qquad \forall \{u, v\} \in E \tag{4k}$$

First of all the objective function (4a) follows from the fact that we want to minimize the size of the cut. Then the inequalities (4b) and (4c) are for the inequality $x_{uv} \geq |f_u - f_v|$; that is the difference between the absorbance. Now we need to establish the flow to be proper. Therefore the equality (4d) means that we have k resources which are distributed from s and so the equality (4e) includes the source to be the starting point of the flow. Next we have a slightly changed Kirchoff's law (4f); meaning the amount of flow going to vertex will either flow out or the vertex will absorb some of it. Lastly we have the inequality (4h) which enforces the connectivity. If we have non-zero flow going to the vertex we have to absorb some portion of it. Lastly (4i), (4j) and (4k) only lists the boundaries of these variables.

3.3 Consider the distance from source

In this subsection we will provide even better model, which arises from the previously mentioned one (4). This time we will also use the notation $d(u, v)$ for the 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 Djisktra'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 in terms of flow. The mixed integer program is as follows.

$$(k - d(s, v)) \cdot f_v \geq \sum_{u \in V, \{uv\} \in E} f_{uv} \qquad \forall v \in V \setminus \{s\} \tag{5a}$$

Hence we take our MIP (4) and replace inequality (4h) with more strict inequality (5a). Unfortunately this does not lead to a better algorithm. That is it enforces a bigger absorption and also implicitly we get that vertices which are further away won't be considered in any way. So that it may help the program solver for a faster searching of an optimal solution.

3.3.1 Properties

Lets talk about some crucial properties of this mixed integer program.

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

Proof. See that due to the constraint (4h) 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. *A set $S = \{v | f_v = 1\}$ defines a connected induced subgraph $G[S]$.*

Proof. Since $f_s = 1$ we know that s is inside the S. For contradiction assume there is a vertex $v \in S$ which is not connected to the source s by a path. Because $f_v = 1$ we must have that there is a flow over this vertex due to the fifth constraint (4f). 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. □

Lemma 1. *The optimal solution x^* and f^* correspond to a minimal k-connected cut and vice versa.*

Proof. We will show how one solution can be translated from one to the other. Then the optimality will arise from this fact. Otherwise we could create the solution by translation followed by finding a better one and translating it back and hence obtaining a better solution.

Firstly suppose we have an optimal solution of the mixed integer program. Then we define the cut as $S = \{v | f_v = 1\}$. From the previous observations we know that the induced subgraph $G[S]$ is connected. And also from the constraint (4g) we have that $|S| = k$.

Suppose we have S as an minimal k-connected cut. Lets build a shortest path tree starting from s where we consider only the graph $G[S]$. Lets define set f_V to all vertices in S to 1 and to the rest 0. Also define x to be 1 only if exactly one end is inside S. Lastly the function f_E will be defined by the shortest path tree. That is from s we will send the flow with the size of the sub-tree and recursively call on that. See that it satisfy all constraints. x_{uv} is satisfied purely from the definition, the sum of outgoing flow from s is equal to the sum of all sub-trees which is indeed $k - 1$. Also there is k vertices with 1 value f_v and also $f_s = 1$. From the tree structure it is easily observed that the Kirchoff's law is satisfied. And also the connected constraint. □

Lemma 2 (Integrality gap)**.** *The integrality gap between the presented MIP and its linear relaxation is $\Omega(k)$.*

Proof. Suppose we have $k \in \mathbb{N}$ and a clique graph with $\mathbb{N} \ni n \geq k$ vertices. The optimal value of MIP will choose arbitrarily k vertices, because we cannot get better solution by exchanging one vertex for another. That means we will have $n - k$ vertices in the rest. The size of the cut is therefore $k(n - k)$. On the other hand in the linear relaxation we will send from s the same value to all vertices so that only $n - 1$ edges will have non-zero cut-value. For $n - 1$ vertices we will split $k - 1$ equally, hence we will have a cut of size $(n - 1) \cdot \left(1 - \frac{k-1}{n-1}\right) = n - 1 - k + 1 = n - k$. Hence the integrality gap is in this case k. □

Conjecture 1. *The integrality gap between the presented MIP and its linear relaxation is $\mathcal{O}(()k)$.*

Observation 3. *We may exchange our LP solution so that the flow induces a tree rooted at s.*

Proof. Whenever there would be a flow going across forming a cycle we may revert this value from all the edges along the path back to the root and send it via the correct branch. This we can do for all such edges. The values are not changed. □

Lemma 3. *The approximation ratio of a simple greedy algorithm is $\mathcal{O}(k)$.* *?*

Proof. We will firstly assume that the vertices are a set of $[n]$ numbers where $s = 0$. Therefore we have them linearly ordered. Now we run depth first search from vertex s and record its DFS-tree. Observe that every time we choose one vertex based on the absorption we decrease the size of the cut by 1 and at most increase it by the degree of the vertex - 1. This holds in MIP. In the linear relaxation we decrease the cut by its difference from the previous vertex and we may increase it by the flows to its neighbors.

Lets denote $\mathrm{OPT}_{\mathrm{LP}}$ as the result of our LP. We can actually compute the result precisely. Let $s = v_1$ and $v_1, v_2, \ldots, v_k$ be the vertices selected to the set S. Then we have the following.

$$\mathrm{OPT}_{\mathrm{LP}} - \left(\sum_{1 \leq i < j \leq k} x_{v_i v_j} + x_{v_j v_i} \right) + \left(\sum_{i=1}^{k} \sum_{u \in N(v_i) \setminus S} (1 - x_{v_i u}) + (1 - x_{u v_i}) \right) - \left(\sum_{u \neq v \in V \setminus S} x_{uv} + x_{vu} \right) \tag{6}$$

Now lets denote $\hat{x}_{uv} = \max(x_{uv}, x_{vu})$. Observe that if $x_{uv} = x_{vu}$; hence $\hat{x}_{uv} = x_{uv} = x_{vu}$. Now we may rewrite our equation (6).

$$\mathrm{OPT}_{\mathrm{LP}} - \sum_{1 \leq i < j \leq k} 2\hat{x}_{v_i v_j} + \left(\sum_{i=1}^{k} \sum_{u \in N(v_i) \setminus S} 2 - 2\hat{x}_{v_i u} \right) - \left(\sum_{u \neq v \in V \setminus S} 2x_{uv} \right) \tag{7}$$

Keep in mind we have doubled our edges so the result itself is twice the optimum. Hence the real optimum OPT can be lower bounded by $\mathrm{OPT}_{\mathrm{LP}}/2$. That is since the linear relaxation gives better fractional result and we have doubled our edges hence the division by 2. Hence we divide our equation (7) and upper bound it by OPT.

$$\leq \mathrm{OPT} - \sum_{1 \leq i < j \leq k} \hat{x}_{v_i v_j} + \left(\sum_{i=1}^{k} \sum_{u \in N(v_i) \setminus S} 1 - \hat{x}_{v_i u} \right) - \left(\sum_{u \neq v \in V \setminus S} x_{uv} \right) \tag{8}$$

Now see that if the first sum is equal to 0 then all other sums are also 0. That is due to the fact, that $x_{uv} = 0$ for all vertices in S and therefore the absorptions f_u are all same. Because we set $f_s = 1$ that means we have absorbed everything in S. Lets consider the case that the sum is non-zero. □

4 Approximation - naive

The final approximation we will employ is quite simple. That is we will have a vertex and a set of already chosen vertices. Firstly add s to the set and set it as a current vertex. Then always look at neighbors with non-zero vertex flow and consider them as another choice. With the flow values choose one with these probabilities.

Algorithm 1 Approximation of the values from linear program.

Require: A graph G with source s and capacity k and absorption f on vertices from LP.
Ensure: A connected cut S.

```
S ← {s} and cv ← s.
while |S| < k do
    for all neighbors v of cv do
        If v ∉ S and f(v) > 0 add it to consideration.
    end for
    Choose one vertex u from the considered ones based on their absorption f.
    S ← S ∪ {u} and cv ← u.
end while
return S.
```

5 Another approximation

In this section lets use the result of our LP (4) with distances (5). Lets denote d_x as a metric on $V \times V \to \mathbb{R}^+$. For vertices $u, v \in V$ we will have $d_x(u, v)$ as the shortest path between these two vertices, where the length of edges are defined by the cut variables x_e.

Definition 7. *Lets define a ball $\mathcal{B}(u, r)$ as a set $\{v \in V | d_x(u, v) \leq r\}$ for $u \in V$ and $r \in [0, 1]$.*

Observation 4. $\mathcal{B}(s, 1) = V$.

Proof. For contradiction say that some vertex $v \notin \mathcal{B}(s, 1)$. Then $d_x(s, v) > 1$, which means that starting from source vertex s with value $f_s = 1$ we are decreasing the value of absorption along the path for the vertices. Every time if some amount is decreased then the same amount is on the edge x_e. And since we cannot get the values f_v lower than zero, then there must be some vertex on the path which has bigger value than its predecessor, which contradicts that it is shortest path. □

Lemma 4. *For all vertices $v \in V$ the inequality $f_v \geq 1 - d_x(s, v)$ holds.*

Proof. Lets prove this by an induction. Firstly see that for s we obtain $f_s \geq 1 - 0 = 1$ which holds from the equality (4e). Then for all neighbors of s we have $f_v \geq 1 - d_x(s, v) \geq 1 - x_{sv}$ and from the inequalities (4b) and (4c) we have that $f_v \geq 1 - f_s + f_v = 1 - 1 + f_v = f_v$. Thus it also holds.

Now lets consider a vertex $v \in V$ and for contradiction suppose that $f_v < 1 - d_x(s, v)$ and that it is the closest vertex in terms of d_x to the source with wrong value. Hence for all neighbors which are closer it must hold. Take the one which has shortest path from source summed with the value of the common edge. Denote this vertex u. Therefore $f_u \geq 1 - d_x(s, u)$. Now $1 - d_x(s, v) = 1 - (d_x(s, u) + x_{uv}) \leq 1 - d_x(s, u) - (f_u - f_v) = 1 - d_x(s, u) - f_u + f_v \leq f_v$. Which is a contradiction. □

Now the algorithm will be in a following matter. We will uniformly at random choose $r \in [0, 1]$ and set $S = \mathcal{B}(s, r)$. In a real implementation we will try this multiple times with different r values and try to find the best algorithm. We must point out that indeed this algorithm will result in bicriterial approximation.

Lemma 5. *The expected values of the cut and the size of $\mathcal{B}(s, r)$ is optimal.*

Proof. Firstly see that the probability $\mathbb{P}[e \in E(\mathcal{B}(s, r), V \setminus \mathcal{B}(s, r))] = x_e$. This can be seen by a simple observation. The edge e will be in cut if r will be inside the bounds $[\alpha, \alpha + x_e]$ for some $\alpha \in \mathbb{R}_0^+$. But the probability is the same as in the case of $[0, x_e]$. Hence the expected value

$$\mathbb{E}[e(\mathcal{B}(s, r), V \setminus \mathcal{B}(s, r))] = \sum_{e \in E} \mathbb{P}[e \in E(\mathcal{B}(s, r), V \setminus \mathcal{B}(s, r))] = \sum_{e \in E} x_e = \mathrm{OPT}_{\mathrm{LP}}.$$

Now we will look at the size of the set $\mathcal{B}(s, r)$.

$$\mathbb{P}[v \in \mathcal{B}(s, r)] = \mathbb{P}[d_x(s, v) \leq r] \geq \mathbb{P}[1 - f_v \leq r] = f_v$$

For the last equality see that having r larger than $1 - f_v$ is same as the value of f_v itself, for example if f_v is closer to 1 the the probability is relative high oppositely to the case where f_v is close to zero then the value of r must be close to 1. And the previous inequality follows from the previous lemma 4. □

5.1 Improving the result – heuristic

Also there is the fact that we want to have cut edges only when the flow is really low. Otherwise it does not make any sense. So either we may be improving our linear program along the way or just consider this during the approximation. That is we will call the edges with flow > 1 and cut > 0 as a bad edges. Then we will have a procedure which will augment the linear program. We will gradually add bad edges to the LP statement where we set these to 0.

Note that this augmentation will in worse case go through all edges, thus $m = |E|$. We should take a while and convince ourselves that this improvement does not diverge from the optimal solution, moreover it will get closer to some optimal solution. On the other hand the optimal value can be different; see `necklace-alt` experiment.

5.2 Algorithm for finding k-connected cut

Now we will present the whole algorithm which will find an optimal k-connected cut.

Conjecture 2. *The above mentioned linear program with augmentation and approximation algorithm will find the optimal connected k-cut for trees in polynomial time.*

Algorithm 2 Augmentation

Require: Graph G with source s and capacity k.
Ensure: Augmented LP.

```
Solve the original LP.
F ← ∅ and cont ← false.
while cont do
    cont ← false
    for all edges e in G do
        if f_e > 1 and x_e > 0 then
            F ← F ∪ {e}.
            cont ← true.
        end if
    end for
    for all e in F do
        Set x(e) = 0 in the LP.
    end for
    Solve updated LP and update solution.
end while
return Last LP.
```

Algorithm 3 k-connected cut algorithm.

Require: A graph G with source s and capacity k.
Ensure: A connected cut S which minimizes its cost.

```
S ← ∅
Create a LP (5).
Solve the LP and run Approximation (1) n times and update S.
Augment the LP via algorithm (2).
Solve the LP and run Approximation (1) n times and update S.
return S.
```

6 Example graphs

We will be looking at a few graphs and mostly their classes and try to observe some properties.

6.1 Star graph

Definition 8. *Star graph is a graph with one vertex with degree Δ and the rest of the vertices have degree 1 and are connected to the main one.*

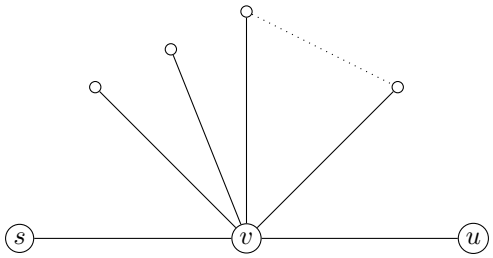


Figure 1: Star graph G with special vertices v, s and u.

6.2 Comet graph

Definition 9. *Comet graph contain a star and a tail – hence the name comet.*

7 More sources

Now we will try to use our linear program to also introduce a solution for the multi-commodity case; when we have more sources and all of the parts have to be connected. In this particular case we are considering the case

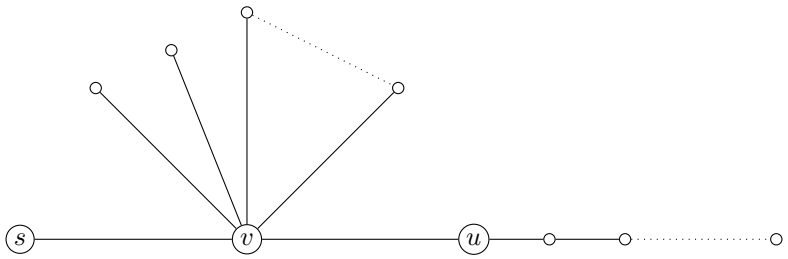


Figure 2: Comet graph G.

with minimizing the sum.

7.1 Linear program

We will now fully establish the linear program, derived from an integer program by its relaxation.

$$\min \sum_{e \in E} \sum_{i=1}^{k} x_e^i \tag{9a}$$

$$x_{uv}^i \geq f_u^i - f_v^j \qquad \forall \{uv\} \in E \; \forall j \in [k] \tag{9b}$$

$$x_{uv}^i \geq f_v^j - f_u^i \qquad \forall \{uv\} \in E \; \forall j \in [k] \tag{9c}$$

$$\sum_{v \in V, sv \in E} f_{s_i v}^i = n_i - 1 \qquad \forall i \in [k] \tag{9d}$$

$$f_{s_i} = 1 \qquad \forall i \in [k] \tag{9e}$$

$$\sum_{u \in V, uv \in E} f_{uv}^i = \sum_{u \in V, vu \in E} f_{vu}^i + f_v^i \qquad \forall v \in V, s \neq v \; \forall i \in [k] \tag{9f}$$

$$\sum_{u \in V} f_u^i = n_i \qquad \forall i \in [k] \tag{9g}$$

$$(n-k) \cdot f_v^i \geq \sum_{u \in V, \{uv\} \in E} f_{uv}^i \qquad \forall v \in V \setminus \{s\} \; \forall i \in [k] \tag{9h}$$

$$\sum_{i=1}^{k} n_i = n \tag{9i}$$

$$\sum_{i=1}^{k} f_v^i = 1 \qquad \forall v \in V \tag{9j}$$

$$f_v^i \in \{0,1\} \qquad \forall v \in V \; \forall i \in [k] \tag{9k}$$

$$f_{uv}^i \in \mathbb{R}^+ \qquad \forall \{u,v\} \in E \; \forall i \in [k] \tag{9l}$$

$$x_{uv}^i \in \{0,1\} \qquad \forall \{u,v\} \in E \tag{9m}$$

Or alternatively if we want to solve min-max problem then we can switch the function (9a) by (10a) and also add (10b).

$$\min c \tag{10a}$$

$$\sum_{e \in E} x_e^i \leq c \qquad \forall i \in [k]. \tag{10b}$$

7.1.1 Variables

Instead of one flow we will have multiple flows. That is $f_v^i \in \{0,1\}$ for all $i \in [k]$ and $v \in V$. Therefore we will also have $f_e^i \in \mathbb{R}_0^+$ for all $i \in [k]$ and $e \in E$. We now do not know the size of each part, hence we also introduce variables $k^i \in \mathbb{N}$ for all $i \in [k]$. Lastly we need cut variables $x_e^i \in \{0,1\}$ for all $i \in [k]$.

7.1.2 Constraints

Some constraints are quite easy to establish. Firstly we set $\sum_{i=1}^{k} k^i = |V|$ that means that we have covered all vertices. Also $\forall v \in V : \sum_{i=1}^{k} f_v^i = 1$, that is every vertex is in one of the absorptive flow. Now we also hard code that $f_{s_i}^i = 1$ meaning that s_i is in i-th part. The cut we will be similar to the single source problem, that is $x_e^i \geq \left|f_u^i - f_v^i\right|$ for $e = \{u, v\}$ and all $i \in [k]$.

There is way more constraints, but they are just translated from the single source absorptive flow to multiple ones. Some of them were even already presented. Note that the constraint which enforces the flow to be connected must change; otherwise we would have a quadratic constraint. But switching n for k will do the job, but it will be little worse.

Also we want to emphasize that there will be one flow which will dominate. That is $\forall e \in E, i \in [k] : \left|f_e^i - \sum_{j \neq i} f_e^j\right| > 1/2$.

7.1.3 Optimization function

Now we have two options for optimizing the values. Firstly the sum would be $\min \sum_{e \in E} \sum_{i \in [k]} x_e^i$. For min-max problem we would do a simple trick, which would be adding a constraints $\forall i \in [k] : \sum_{e \in E} x_e^i \leq c$ and now we will minimize $\min c$.

8 NP hardness

It is crucial to ask ourselves if the problem k-connected cut is NP-complete problem. If we show that it is indeed that hard then we might not expect any polynomial time algorithm which solves it. In this section we will show a polynomial reduction from bisection problem.

8.1 Reduction from minimal bisection with source

Lets have a graph $G = (V, E)$ for a minimal bisection problem. We will create a graph $G' = (V \cup \{s\}, E \cup \{\{u, s\} | \forall u \in V\})$ and also set $k = n/2 + 1$. Now we could use k-connected cut with source s.

Lemma 6. *The optimal value of k-connected cut with source for the instance (G', k, s) is same as for minimal bisection on graph G subtracted by $n/2$.*

Proof. Since $s \in S$ then we have $k - 1 = n/2$ many vertices we have to choose to add to S. Since the cut edges between s and the rest of the graph will always be $n/2$ then we can only minimize the cut within the graph G. The connectivity of the k-connected cut does not change the result since all vertices from V are connected to s anyway. Hence we want to find a set of size $n/2$ which minimizes the cut between them, which is exactly the problem of minimal bisection. □

8.2 Reduction from minimal bisection without source

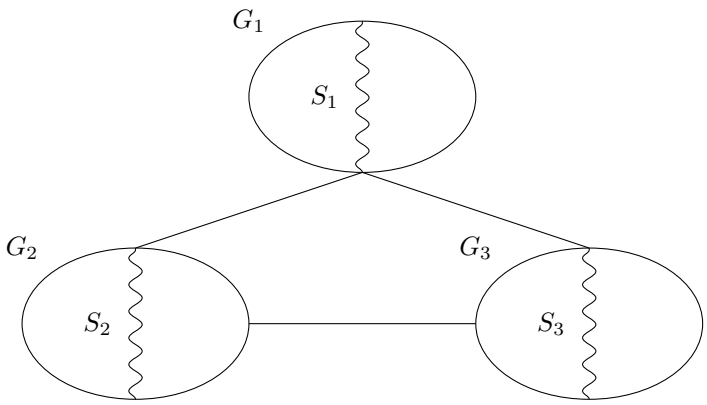
Now we will show a reduction from minimal bisection for the problem of k-connected cut without source. The reduction itself will be in a following way. Firstly check if the given graph is a clique. This can be done by checking if all vertices have degree $n - 1$; hence in polynomial time. If it is a clique run k-connected cut on the graph for $k = n/2$. If not then create auxiliary graph $G' = (V', E')$. Start by creating three copies of G that is $G_i = (V_i, E_i)$ for $i \in \{1, 2, 3\}$. Now create $v' \bigcup_{i=1}^{3} V_i$ and $E' = \bigcup_{i=1}^{3} E_i \cup \bigcup_{i=1}^{3} \{\{u, v\} | \forall i \neq j \in \{1, 2, 3\} \forall u \in V_i \forall v \in V_j\})$. Now run k-connected cut on the graph G' for $k = 3/2n$.

8.2.1 Computing the size of the cut

Lets now compute the size of the cut. Lets say we have chosen $\alpha_i n$ vertices from G_i where $\alpha_i \in [0, 1]$ and $\sum_{i=1}^{3} \alpha_i = 3/2$. Denote the subsets as S_i respectively to their parts of G_i. Now the size of the cut $S = \bigcup_{i=1}^{3} S_i$ of G' can be computed precisely.

$$e(S, V' \setminus S) = \sum_{i=1}^{3} \sum_{j \in \{1,2,3\}; j \neq i} \alpha_i n (1 - \alpha_j) n + \sum_{i=1}^{3} e(S_i, V_i \setminus S_i) \tag{11}$$

When all α_i's would be equal to $1/2$ then we would obtain $3/2n^2$ for the first term. Observe that if either one of the α_i's would exactly equal to $1/2$ then we have found our minimal bisection. That is due to the fact, that the interaction between three copies will be same for any $1/2$ choice of vertices therefore we would minimize such inner cut.



Lets now say that none of the α_i's are equal to $1/2$. Therefore one must be greater than $1/2$ and also one must be lesser. Let us update the equation (11). We can also use the fact that $\alpha_3 = 3/2 - \alpha_1 - \alpha_2$.

$$
\begin{aligned}
e(S, V' \setminus S) &= \sum_{i=1}^{3} \sum_{j \in \{1,2,3\}; j \neq i} \alpha_i n (1 - \alpha_j) n + \sum_{i=1}^{3} e(S_i, V_i \setminus S_i) \\
&= \alpha_1 n^2 (1 - \alpha_2) + \alpha_1 n^2 (1 - \alpha_3) + \alpha_2 n^2 (1 - \alpha_1) + \alpha_2 n^2 (1 - \alpha_3) + \\
&\quad + \alpha_3 n^2 (1 - \alpha_1) + \alpha_3 n^2 (1 - \alpha_2) + \sum_{i=1}^{3} e(S_i, V_i \setminus S_i) \\
&= 2\alpha_1 n^2 - \alpha_1 \alpha_2 n^2 - \alpha_1 \alpha_3 n^2 + 2\alpha_2 n^2 - \alpha_1 \alpha_2 n^2 - \alpha_2 \alpha_3 n^2 + \\
&\quad + 2\alpha_3 n^2 - \alpha_1 \alpha_3 n^2 - \alpha_2 \alpha_3 n^2 + \sum_{i=1}^{3} e(S_i, V_i \setminus S_i) \\
&= 2\alpha_1 n^2 - 2\alpha_1 \alpha_2 n^2 - 2\alpha_1 \alpha_3 n^2 + 2\alpha_2 n^2 - 2\alpha_2 \alpha_3 n^2 + \\
&\quad + 2\alpha_3 n^2 + \sum_{i=1}^{3} e(S_i, V_i \setminus S_i) \\
&= 2\alpha_1 n^2 - 2\alpha_1 \alpha_2 n^2 - 2\alpha_1 (3/2 - \alpha_1 - \alpha_2) n^2 + 2\alpha_2 n^2 - 2\alpha_2 (3/2 - \alpha_1 - \alpha_2) n^2 + \\
&\quad + 2(3/2 - \alpha_1 - \alpha_2) n^2 + \sum_{i=1}^{3} e(S_i, V_i \setminus S_i) \\
&= 2\alpha_1 n^2 - 2\alpha_1 \alpha_2 n^2 - 3\alpha_1 n^2 + 2\alpha_1^2 n^2 + 2\alpha_1 \alpha_2 n^2 + 2\alpha_2 n^2 - 3\alpha_2 n^2 + 2\alpha_1 \alpha_2 n^2 + 2\alpha_2^2 n^2 + \\
&\quad + 3n^2 - 2\alpha_1 n^2 - 2\alpha_2 n^2 + \sum_{i=1}^{3} e(S_i, V_i \setminus S_i) \\
&= 3n^2 - 3\alpha_1 n^2 - 3\alpha_2 n^2 + 2\alpha_1 \alpha_2 n^2 + 2\alpha_1^2 n^2 + 2\alpha_2^2 n^2 + \sum_{i=1}^{3} e(S_i, V_i \setminus S_i) \\
&= n^2 \left(3 - 3\alpha_1 - 3\alpha_2 + 2\alpha_1 \alpha_2 + 2\alpha_1^2 + 2\alpha_2^2\right) + \sum_{i=1}^{3} e(S_i, V_i \setminus S_i)
\end{aligned}
\tag{12}
$$

We can observe or use any plotting tool to see that the term $\left(3 - 3\alpha_1 - 3\alpha_2 + 2\alpha_1 \alpha_2 + 2\alpha_1^2 + 2\alpha_2^2\right)$ is minimized if $\alpha_1 = \alpha_2 = 1/2$. You may see the Fig. 3.

Without loss of generality we may assume that $\alpha_1 > 1/2$ and $\alpha_2 < 1/2$ due to the fact, that at least one must be larger and one lower. Now we can also assume that $\alpha_3 < 1/2$ because if that won't be the case then we could always take the complements $S_i' := V_i \setminus S_i$ for all i's resulting in this case. See that the size of the cut is same. For contradiction suppose that this cut minimizes the sum. We can now add all edges between all pairs of vertices within S_i's and $V_i \setminus S_i$'s while not increasing the optimum but making it way more expensive to take halves, i.e. all $\alpha_i = 1/2$.

Lets compute the rise possible increase in the inner cuts.

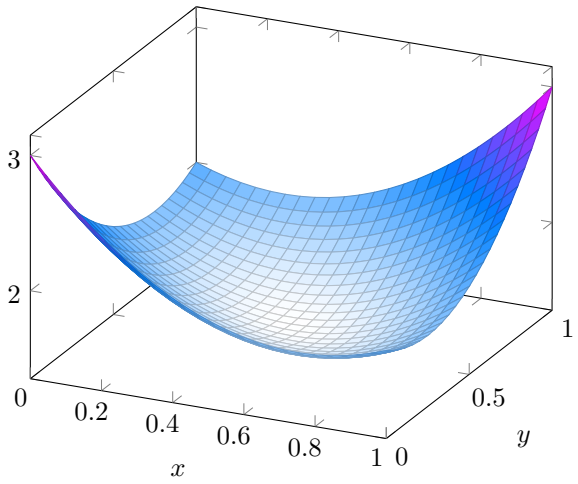


Figure 3: Plotted function $f(x, y) = 3 - 3x - 3y + 2xy + 2x^2 + 2y^2$.

$$
\begin{aligned}
&= 1/2n^2(\alpha_1 - 1/2) + 1/2n^2(1 - \alpha_2 - 1/2) + 1/2n^2(1 - \alpha_3 - 1/2) \\
&= 1/2\alpha_1 n^2 - 1/4n^2 + 1/4n^2 - 1/2\alpha_2 n^2 + 1/4n^2 - 1/2\alpha_3 n^2 \\
&= 1/2\alpha_1 n^2 - 1/2\alpha_2 n^2 + 1/4n^2 - 1/2(3/2 - \alpha_1 - \alpha_2)n^2 \\
&= 1/2\alpha_1 n^2 - 1/2\alpha_2 n^2 + 1/4n^2 - 3/4n^2 + 1/2\alpha_1 n^2 + 1/2\alpha_2 n^2 \\
&= \alpha_1 n^2 - 1/2n^2 = n^2(\alpha_1 - 1/2)
\end{aligned} \tag{13}
$$

So that in total we would have $n^2 + \alpha_1 n^2 + \sum_{i=1}^{3} e(S_i, V_i \setminus S_i)$. Now $3n^2 - 3\alpha_1 n^2 - 3\alpha_2 n^2 + 2\alpha_1\alpha_2 n^2 + 2\alpha_1^2 n^2 + 2\alpha_2^2 n^2 + \sum_{i=1}^{3} e(S_i, V_i \setminus S_i) - \left(n^2 + \alpha_1 n^2 + \sum_{i=1}^{3} e(S_i, V_i \setminus S_i)\right) = 2n^2 - 4\alpha_1 n^2 - 3\alpha_2 n^2 + 2\alpha_1\alpha_2 n^2 + 2\alpha_1^2 n^2 + 2\alpha_2^2 n^2$. We want to showcase that this value will be for $x \in (1/2, 1]$ and $y \in [0, 1/2)$ is always > 0.

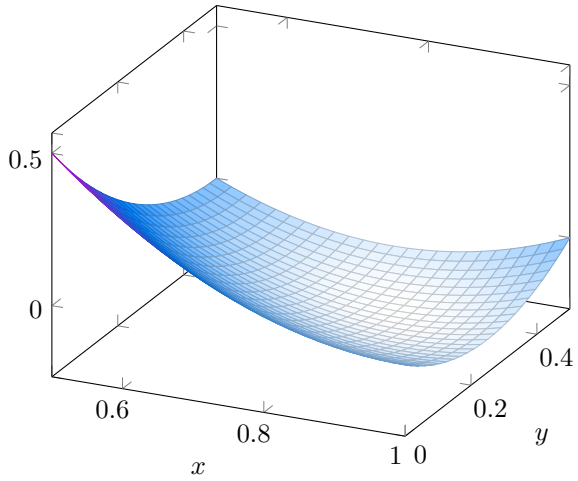


Figure 4: Plotted function $f(x, y) = 2 - 4x - 3y + 2xy + 2x^2 + 2y^2$.

Links

- Anupam Gupta, Kunal Talwar: Approximating unique games, 2006
- Uriel Feige†, Robert Krauthgamer: Minimum Bisection, 2001
- Min max and small set expansion
- Approximation algorithms for maximally balanced connected graph partition
- On size-constrained minimum s–t cut problems and size-constrained dense subgraph problems
- On the minimum $s - t$ cut problem with budget constraints

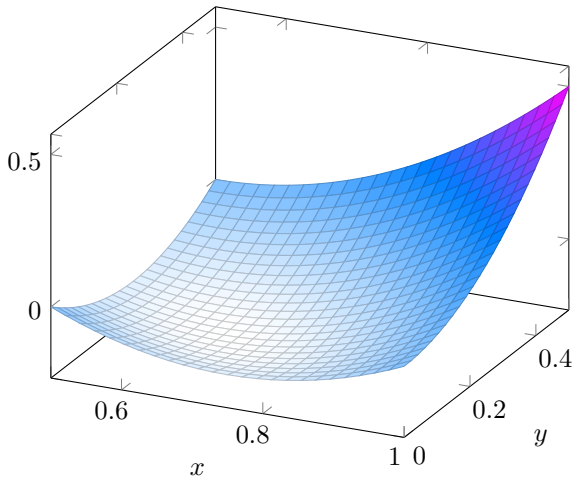


Figure 5: Plotted function $f(x, y) = 1 - 3 * x - 2 * y + 2 * x * y + 2 * x^2 + 2 * y^2$.

- Balanced Crown Decomposition for Connectivity Constraints
- (Almost) Tight Bounds and Existence Theorems for Single-Commodity Confluent Flows, 2007
- Polylogarithmic Approximations for the Capacitated Single-Sink Confluent Flow Problem, 2015
- R. K. Ahuja, T. L. Magnati, J. B. Orlin. Network Flows: Theory, Algorithms, and Applications, Pearson, 1993.