Testing $\mathcal{MP}(G)$

Péter Hajnal

2024. Fall

Reminder

Reminder

Definition (matching polytope)

$$\mathcal{MP}(G) = \mathrm{conv}\{\chi_M : M \text{ is a matching in } G\}.$$

Reminder

Definition (matching polytope)

$$\mathcal{MP}(G) = \mathrm{conv}\{\chi_M : M \text{ is a matching in } G\}.$$

Edmonds Polytope Theorem)

$\mathcal{MP}(G)$ consists exactly of the vectors $(x_e)_{e\in E} \in \mathbb{R}^{E(G)}$ that satisfy the following three types of inequalities:

$$(E_e): \qquad x_e \geq 0 \qquad \forall e \in E(G)$$

$$(E_v): \qquad \sum_{e:v I e} x_e \leq 1 \qquad \forall v \in V(G)$$

$$(E_S): \qquad \sum_{e \subseteq S} x_e \leq \frac{|S|-1}{2} \qquad \forall S \in \mathcal{O}$$

The Problems

The Problems

The description of $\mathcal{MP}(\mathcal{G})$ in the Edmonds theorem contains $|V(G)| + |E(G)| + 2^{|V(G)|-1}$ inequalities. That is exponentially many inequalities needed.

The Problems

The description of $\mathcal{MP}(\mathcal{G})$ in the Edmonds theorem contains $|V(G)| + |E(G)| + 2^{|V(G)|-1}$ inequalities. That is exponentially many inequalities needed.

Some of these may be redundant (we saw that for bipartite G this can lead to significant simplification), but the polytope is generally complicated (the number of required inequalities is exponential in the number of vertices and edges of the graph).

The Basic Question

The Basic Question

Can we test *quickly* whether a given $x \in \mathbb{R}^{E(G)}$ is an element of $\mathcal{MP}(\mathcal{G})$?

The Basic Question

Can we test *quickly* whether a given $x \in \mathbb{R}^{E(G)}$ is an element of $\mathcal{MP}(\mathcal{G})$? Let's complicate things a bit:

The Basic Question

Can we test *quickly* whether a given $x \in \mathbb{R}^{E(G)}$ is an element of $\mathcal{MP}(\mathcal{G})$? Let's complicate things a bit:

The Basic Question

Find an efficient algorithm that, given a $x \in \mathbb{Q}^{E(G)}$, outputs either the true information that x is an element of $\mathcal{MP}(\mathcal{G})$, or a defining inequality violated by the x vector.

The Basic Question

Can we test *quickly* whether a given $x \in \mathbb{R}^{E(G)}$ is an element of $\mathcal{MP}(\mathcal{G})$? Let's complicate things a bit:

The Basic Question

Find an efficient algorithm that, given a $x \in \mathbb{Q}^{E(G)}$, outputs either the true information that x is an element of $\mathcal{MP}(\mathcal{G})$, or a defining inequality violated by the x vector.

By efficient we mean polynomial time in the size of G. So the naive algorithm (substitute x coordinates into every inequality and after checks announce the result) won't do.

The Obvious Part

The Obvious Part

The algorithm for testing whether x falls into the Edmonds polytope is, of course, straightforwardly started:

The Obvious Part

The algorithm for testing whether x falls into the Edmonds polytope is, of course, straightforwardly started:

The Start

Check inequalities (E_v) and (E_e).

The Obvious Part

The algorithm for testing whether x falls into the Edmonds polytope is, of course, straightforwardly started:

The Start

Check inequalities (E_v) and (E_e).

Two possibilities arise:

The Obvious Part

The algorithm for testing whether x falls into the Edmonds polytope is, of course, straightforwardly started:

The Start

Check inequalities (E_v) and (E_e).

Two possibilities arise:

- We get an inequality violated by x,
- All (E_v) and (E_e) inequalities are satisfied by x.

The Obvious Part

The algorithm for testing whether x falls into the Edmonds polytope is, of course, straightforwardly started:

The Start

Check inequalities (E_v) and (E_e).

Two possibilities arise:

- We get an inequality violated by x,
- All (E_v) and (E_e) inequalities are satisfied by x.

In the first case, we're done, as we now know that $x \notin \mathcal{MP}(G)$ and have found a violated inequality.

The Obvious Part

The algorithm for testing whether x falls into the Edmonds polytope is, of course, straightforwardly started:

The Start

Check inequalities (E_v) and (E_e).

Two possibilities arise:

- We get an inequality violated by x,
- All (E_v) and (E_e) inequalities are satisfied by x.

In the first case, we're done, as we now know that $x \notin \mathcal{MP}(G)$ and have found a violated inequality.

So for the rest, we can assume that the (E_v) and (E_e) inequalities hold for the vector x to be tested.

Edge Weighted Graphs

Edge Weighted Graphs

We switch to a new notation and perspective.

Edge Weighted Graphs

We switch to a new notation and perspective.

$x \in \mathbb{R}^E$ means that x is a vector, its coordinates correspond to the edges. x_e is the component of x corresponding to edge e. x_e can also be interpreted as the weight of edge e. We'll use this perspective henceforth.

Edge Weighted Graphs

We switch to a new notation and perspective.

$x \in \mathbb{R}^E$ means that x is a vector, its coordinates correspond to the edges. x_e is the component of x corresponding to edge e. x_e can also be interpreted as the weight of edge e. We'll use this perspective henceforth.

So $(x_e)_{e\in E} \in \mathbb{R}^{E(G)}$ is an edge-weighted graph.

Edge Weighted Graphs

We switch to a new notation and perspective.

$x \in \mathbb{R}^E$ means that x is a vector, its coordinates correspond to the edges. x_e is the component of x corresponding to edge e. x_e can also be interpreted as the weight of edge e. We'll use this perspective henceforth.

So $(x_e)_{e \in E} \in \mathbb{R}^{E(G)}$ is an edge-weighted graph.

We assumed the weights to be non-negative. At every vertex, the sum of weights of incident edges is at most 1.

Vector vs Function Perspective

Vector vs Function Perspective

Edge weighting can also be naturally thought of as a function $x : E(G) \to \mathbb{R}_+$ (we know that x satisfies the (E_e) inequalities). So $x(e) = x_e$ will denote the weight of edge e.

Vector vs Function Perspective

Edge weighting can also be naturally thought of as a function $x : E(G) \to \mathbb{R}_+$ (we know that x satisfies the (E_e) inequalities). So $x(e) = x_e$ will denote the weight of edge e.

For a subset $F \subset E(G)$, we use the notation

$$x(F) = \sum_{e:e \in F} x_e = \sum_{e:e \in F} x(e)$$

Vector vs Function Perspective

Edge weighting can also be naturally thought of as a function $x : E(G) \to \mathbb{R}_+$ (we know that x satisfies the (E_e) inequalities). So $x(e) = x_e$ will denote the weight of edge e.

For a subset $F \subset E(G)$, we use the notation

$$x(F) = \sum_{e:e \in F} x_e = \sum_{e:e \in F} x(e)$$

If $R \subset V(G)$, then

$$x(R) = \sum_{e:e=xy \in E(G), x,y \in R} x_e = \sum_{e:e=xy \in E(G), x,y \in R} x(e)$$

notation is also used.

Vector vs Function Perspective

Edge weighting can also be naturally thought of as a function $x : E(G) \to \mathbb{R}_+$ (we know that x satisfies the (E_e) inequalities). So $x(e) = x_e$ will denote the weight of edge e.

For a subset $F \subset E(G)$, we use the notation

$$x(F) = \sum_{e:e\in F} x_e = \sum_{e:e\in F} x(e)$$

If $R \subset V(G)$, then

$$x(R) = \sum_{e:e=xy\in E(G),x,y\in R} x_e = \sum_{e:e=xy\in E(G),x,y\in R} x(e)$$

notation is also used.

At first glance, these conventions may be confusing. The meaning of $x(\cdot)$ depends on whether the parentheses contain an edge, an edge set, or a vertex set. Let's take the time and effort to get used to it.

Observation

Observation

$2^{|V|}$ inequalities

$$x(R) := \sum_{e \subseteq R} x_e \leq \frac{|R|}{2} \qquad \forall R \in \mathcal{P}(V)$$

each is implied by the (E_v) and (E_e) inequalities.

Observation

Observation

$2^{|V|}$ inequalities

$$x(R) := \sum_{e\subseteq R} x_e \leq \frac{|R|}{2} \qquad \forall R \in \mathcal{P}(V)$$

each is implied by the (E_v) and (E_e) inequalities.

Summarizing, the (E_S) conditions *only* mean that the above estimate (which holds for every vertex set) can be sharpened by $1/2$ for subsets with odd cardinality.

Proof of Observation

Proof of Observation

Sum up the inequalities

$$\sum_{e \in E: v I e} x_e \leq 1, v \in R$$

Proof of Observation

Sum up the inequalities

$$\sum_{e \in E: vIe} x_e \leq 1, v \in R$$

The result is

$$\sum_{e=uv \in E: u \in R, v \notin R} x_e + 2 \cdot x(R) \leq |R|.$$

Proof of Observation

Sum up the inequalities

$$\sum_{e\in E:vIe} x_e \leq 1, v \in R$$

The result is

$$\sum_{e=uv\in E:u\in R,v\notin R} x_e + 2 \cdot x(R) \leq |R|.$$

In the following,

$$\partial R := \{e = uv \in E : u \in R, v \notin R\}, \qquad x(\partial R) := \sum_{e=uv\in E:u\in R,v\notin R} x_e$$

notations are used.

Proof of Observation

Sum up the inequalities

$$\sum_{e \in E: v I e} x_e \leq 1, v \in R$$

The result is

$$\sum_{e=uv \in E: u \in R, v \notin R} x_e + 2 \cdot x(R) \leq |R|.$$

In the following,

$$\partial R := \{e = uv \in E : u \in R, v \notin R\}, \qquad x(\partial R) := \sum_{e=uv \in E: u \in R, v \notin R} x_e$$

notations are used.

Since the components of x are non-negative, for any vertex set R

$$2 \cdot x(R) \leq x(\partial R) + 2 \cdot x(R) \leq |R|, \quad \text{thus} \quad x(R) \leq \frac{|R|}{2}.$$

The First Goal

The First Goal

The testing of the (E_S) conditions needs to be performed on any edge-weighted graph (G, x) for which the (E_v) and (E_e) conditions are satisfied.

The First Goal

The testing of the (E_S) conditions needs to be performed on any edge-weighted graph (G, x) for which the (E_v) and (E_e) conditions are satisfied.

1st Goal

We show that if this problem is solved for non-negative vectors where all (E_v) conditions are satisfied with equality, then the general problem can be solved.

The First Goal

The testing of the (E_S) conditions needs to be performed on any edge-weighted graph (G, x) for which the (E_v) and (E_e) conditions are satisfied.

1st Goal

We show that if this problem is solved for non-negative vectors where all (E_v) conditions are satisfied with equality, then the general problem can be solved.

Given a non-negative edge weighting, assuming that for every vertex the sum of weights of incident edges is at most 1.

The First Goal

The testing of the (E_S) conditions needs to be performed on any edge-weighted graph (G, x) for which the (E_v) and (E_e) conditions are satisfied.

1st Goal

We show that if this problem is solved for non-negative vectors where all (E_v) conditions are satisfied with equality, then the general problem can be solved.

Given a non-negative edge weighting, assuming that for every vertex the sum of weights of incident edges is at most 1.

From a (G, x) construct a $\widetilde{G}, \widetilde{x}$ pair, which already satisfies the (E_v) inequalities with equality (the sum of weights of incident edges at each vertex is exactly 1).

The Reduction

The Reduction

- Take the graph G and a copy of it, G'.

The Reduction

- Take the graph G and a copy of it, G'.

- Consider a complete matching between the *twin vertices*. The $\widetilde{x}$ weighting in G and G' is the same as x, and the crossing edges' weight in $\widetilde{x}$ is

$$\widetilde{x}_{vv'} = 1 - \sum_{e \in E(G), v I e} x_e, \quad \forall v \in V.$$

The Reduction

- Take the graph G and a copy of it, G'.

- Consider a complete matching between the *twin vertices*. The $\widetilde{x}$ weighting in G and G' is the same as x, and the crossing edges' weight in $\widetilde{x}$ is

$$\widetilde{x}_{vv'} = 1 - \sum_{e \in E(G), v I e} x_e, \quad \forall v \in V.$$

- This quantity is non-negative due to the (E_v) condition, so the sign conditions hold for the $\widetilde{G}, \widetilde{w}$ pair, and moreover, the (E_v) inequalities are satisfied with equality.

The Reduction

- Take the graph G and a copy of it, G'.
- Consider a complete matching between the *twin vertices*. The $\widetilde{x}$ weighting in G and G' is the same as x, and the crossing edges' weight in $\widetilde{x}$ is

$$\widetilde{x}_{vv'} = 1 - \sum_{e \in E(G), v I e} x_e, \quad \forall v \in V.$$

- This quantity is non-negative due to the (E_v) condition, so the sign conditions hold for the $\widetilde{G}, \widetilde{w}$ pair, and moreover, the (E_v) inequalities are satisfied with equality.

Claim supporting the 1st Goal

For (G, x), then all (E_S) conditions hold if and only if they hold for $(\widetilde{G}, \widetilde{x})$.

Odd Vertex Sets in $\widetilde{G}$

Odd Vertex Sets in $\widetilde{G}$

One direction of the claim is obvious:

Odd Vertex Sets in $\widetilde{G}$

One direction of the claim is obvious:

If for (G, x) any (E_S) condition is false, then the same S set (which is also a subset of $V(\widetilde{G})$) will violate the conditions in $\widetilde{G}$.

Odd Vertex Sets in $\widetilde{G}$

One direction of the claim is obvious:

If for (G, x) any (E_S) condition is false, then the same S set (which is also a subset of $V(\widetilde{G})$) will violate the conditions in $\widetilde{G}$.

We only need to prove that if for (G, x) all (E_S) conditions are true, then these also hold for $\widetilde{G}$.

Odd Vertex Sets in $\widetilde{G}$

One direction of the claim is obvious:

If for (G, x) any (E_S) condition is false, then the same S set (which is also a subset of $V(\widetilde{G})$) will violate the conditions in $\widetilde{G}$.

We only need to prove that if for (G, x) all (E_S) conditions are true, then these also hold for $\widetilde{G}$.

To do this, take an odd cardinality set S from $V(\widetilde{G})$.

Odd Vertex Sets in $\widetilde{G}$

One direction of the claim is obvious:

If for (G, x) any (E_S) condition is false, then the same S set (which is also a subset of $V(\widetilde{G})$) will violate the conditions in $\widetilde{G}$.

We only need to prove that if for (G, x) all (E_S) conditions are true, then these also hold for $\widetilde{G}$.

To do this, take an odd cardinality set S from $V(\widetilde{G})$.

Notation

Let $S \subset V(\widetilde{G})$ be arbitrary. $R = S \cap V(G)$ and $T' = S \cap V(G')$ are the two parts of set S. Think of T' as a twin of a vertex set $T \subset V(G)$.

Odd Vertex Sets in $\widetilde{G}$

One direction of the claim is obvious:

If for (G, x) any (E_S) condition is false, then the same S set (which is also a subset of $V(\widetilde{G})$) will violate the conditions in $\widetilde{G}$.

We only need to prove that if for (G, x) all (E_S) conditions are true, then these also hold for $\widetilde{G}$.

To do this, take an odd cardinality set S from $V(\widetilde{G})$.

Notation

Let $S \subset V(\widetilde{G})$ be arbitrary. $R = S \cap V(G)$ and $T' = S \cap V(G')$ are the two parts of set S. Think of T' as a twin of a vertex set $T \subset V(G)$.

If S has an odd cardinality, then one of S and T is odd, and the other has an even cardinality.

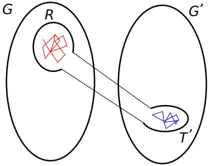
Case 1

Case 1

Case 1: $R \cap T = \emptyset$.

Case 1

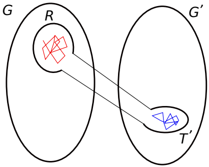
Case 1: $R \cap T = \emptyset$.



The red edges are edges inside a set with an odd number of elements, the blue edges are edges inside a set with an even number of elements.

Case 1

Case 1: $R \cap T = \emptyset$.

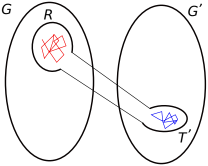


The red edges are edges inside a set with an odd number of elements, the blue edges are edges inside a set with an even number of elements.

This is a straightforward case. Then the set of edges $E(R)$ inside R and the set of edges $E(T')$ inside T' together give the set of edges $E(S)$ inside S. Specifically, $x(S) = x(R) + x(T') = x(R) + x(T)$.

Case 1

Case 1: $R \cap T = \emptyset$.



The red edges are edges inside a set with an odd number of elements, the blue edges are edges inside a set with an even number of elements.

This is a straightforward case. Then the set of edges $E(R)$ inside R and the set of edges $E(T')$ inside T' together give the set of edges $E(S)$ inside S. Specifically, $x(S) = x(R) + x(T') = x(R) + x(T)$.

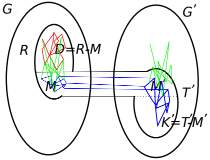
We can estimate $x(R)$ and $x(T)$ by $|R|/2$ and $|T|/2$, respectively, and even the upper bound sharpened by $1/2$ for the set with an odd cardinality.

Case 2

Case 2: $M := R \cap T \neq \emptyset$.

Case 2

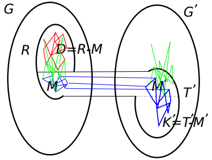
Case 2: $M := R \cap T \neq \emptyset$.



The red edges are edges inside a set with an odd number of elements, the blue edges are edges inside a set with an even number of elements. The green edges are the edges $E(R-M, M)$ and $E(R'-M', M')$, part of the boundary $\partial(M \cup M' \cup K')$. This boundary is included in an inequality derived from (E_v) and (E_e) inequalities earlier.

Case 2

Case 2: $M := R \cap T \neq \emptyset$.



The red edges are edges inside a set with an odd number of elements, the blue edges are edges inside a set with an even number of elements. The green edges are the edges $E(R - M, M)$ and $E(R' - M', M')$, part of the boundary $\partial(M \cup M' \cup K')$. This boundary is included in an inequality derived from (E_v) and (E_e) inequalities earlier.

We can assume that $D \subset V(G)$ is a set with an odd number of elements:

$$x(D) \leq \frac{|D| - 1}{2}.$$

Case 2 (continued)

Case 2 (continued)

We will be more cautious when estimating $x(M \cup M' \cup K')$.

$$x(M \cup M' \cup K') + \frac{1}{2}(x(\partial_G(M) + \partial_{G'}(M' \cup K')) \leq \frac{|M| + |M'| + |K'|}{2}$$

inequality is used, which we derived from the (E_v) and (E_e) inequalities.

Case 2 (continued)

We will be more cautious when estimating $x(M \cup M' \cup K')$.

$$x(M \cup M' \cup K') + \frac{1}{2}(x(\partial_G(M) + \partial_{G'}(M' \cup K')) \leq \frac{|M| + |M'| + |K'|}{2}$$

inequality is used, which we derived from the (E_v) and (E_e) inequalities.

Regarding the previously neglected, *halved* term, it is obvious that

$$x(E(D, M)) \leq \frac{1}{2}(x(\partial_G(M) + \partial_{G'}(M' \cup K')),$$

where $E(D, M)$ is the number of edges crossing between D and M.

Case 2 (continued)

We will be more cautious when estimating $x(M \cup M' \cup K')$.

$$x(M \cup M' \cup K') + \frac{1}{2}(x(\partial_G(M) + \partial_{G'}(M' \cup K')) \leq \frac{|M| + |M'| + |K'|}{2}$$

inequality is used, which we derived from the (E_v) and (E_e) inequalities.

Regarding the previously neglected, *halved* term, it is obvious that

$$x(E(D, M)) \leq \frac{1}{2}(x(\partial_G(M) + \partial_{G'}(M' \cup K')),$$

where $E(D, M)$ is the number of edges crossing between D and M.

Combining these two inequalities yields the desired result straightforwardly.

Case 2 (continued)

We will be more cautious when estimating $x(M \cup M' \cup K')$.

$$x(M \cup M' \cup K') + \frac{1}{2}(x(\partial_G(M) + \partial_{G'}(M' \cup K')) \leq \frac{|M| + |M'| + |K'|}{2}$$

inequality is used, which we derived from the (E_v) and (E_e) inequalities.

Regarding the previously neglected, *halved* term, it is obvious that

$$x(E(D, M)) \leq \frac{1}{2}(x(\partial_G(M) + \partial_{G'}(M' \cup K')),$$

where $E(D, M)$ is the number of edges crossing between D and M.

Combining these two inequalities yields the desired result straightforwardly.

We have thus obtained that testing the (E_S) inequalities is equivalent for (G, x) and $(\widetilde{G}, \widetilde{x})$.

Break

Cuts

Cuts

- In what follows, we only deal with edge-weighted (G, x) graphs where the (E_v) inequalities hold with equality, that is, for every $v \in V(G)$ vertex

$$\sum_{e \in E: v I e} x_e = 1,$$

and the vertex set has an even cardinality.

Cuts

- In what follows, we only deal with edge-weighted (G, x) graphs where the (E_v) inequalities hold with equality, that is, for every $v \in V(G)$ vertex

$$\sum_{e \in E : vIe} x_e = 1,$$

and the vertex set has an even cardinality.

- Our previous derivation can be repeated (now with equalities):

$$2 \sum_{e=xy \in E : x,y, \in S} x_e + \sum_{e \in \partial S} x_e = |S|.$$

Cuts

• In what follows, we only deal with edge-weighted (G, x) graphs where the (E_v) inequalities hold with equality, that is, for every $v \in V(G)$ vertex

$$\sum_{e \in E: v I e} x_e = 1,$$

and the vertex set has an even cardinality.

• Our previous derivation can be repeated (now with equalities):

$$2 \sum_{e = xy \in E: x, y, \in S} x_e + \sum_{e \in \partial S} x_e = |S|.$$

• We change the language slightly: ∂S is the boundary of set S. However, this can be regarded as the edge set $E(\mathcal{V})$ of the cut $\mathcal{V} = (S, \overline{S})$. Let $x(\mathcal{V}) = x(E(\mathcal{V}))$. The $\mathcal{V}$ cut is odd if both its sides are sets with an odd cardinality (we already assume G has an even number of vertices).

Reformulation

Reformulation

- We obtained that for every odd $\mathcal{V}$ cut

$$x(\mathcal{V}) = |S| - 2x(S) = 2\left(\frac{|S|-1}{2} - x(S)\right) + 1.$$

Reformulation

- We obtained that for every odd $\mathcal{V}$ cut

$$x(\mathcal{V}) = |S| - 2x(S) = 2\left(\frac{|S|-1}{2} - x(S)\right) + 1.$$

That is

$$\frac{|S|-1}{2} - x(S) = \frac{x(\mathcal{V})-1}{2}.$$

Reformulation

- We obtained that for every odd $\mathcal{V}$ cut

$$x(\mathcal{V}) = |S| - 2x(S) = 2\left(\frac{|S|-1}{2} - x(S)\right) + 1.$$

That is

$$\frac{|S|-1}{2} - x(S) = \frac{x(\mathcal{V})-1}{2}.$$

- Accordingly, all (E_S) inequalities hold if and only if the weight of every edge set of an odd cut $\mathcal{V}$ is at least 1.

Reformulation

- We obtained that for every odd $\mathcal{V}$ cut

$$x(\mathcal{V}) = |S| - 2x(S) = 2\left(\frac{|S|-1}{2} - x(S)\right) + 1.$$

That is

$$\frac{|S|-1}{2} - x(S) = \frac{x(\mathcal{V})-1}{2}.$$

- Accordingly, all (E_S) inequalities hold if and only if the weight of every edge set of an odd cut $\mathcal{V}$ is at least 1.

Goal 2

Given an edge-weighted, non-negative, and even-sized graph (G, x).

Reformulation

- We obtained that for every odd $\mathcal{V}$ cut

$$x(\mathcal{V}) = |S| - 2x(S) = 2\left(\frac{|S|-1}{2} - x(S)\right) + 1.$$

That is

$$\frac{|S|-1}{2} - x(S) = \frac{x(\mathcal{V})-1}{2}.$$

- Accordingly, all (E_S) inequalities hold if and only if the weight of every edge set of an odd cut $\mathcal{V}$ is at least 1.

Goal 2

Given an edge-weighted, non-negative, and even-sized graph (G, x). Determine efficiently the minimum weight odd cut.

Reformulation

- We obtained that for every odd $\mathcal{V}$ cut

$$x(\mathcal{V}) = |S| - 2x(S) = 2\left(\frac{|S|-1}{2} - x(S)\right) + 1.$$

That is

$$\frac{|S|-1}{2} - x(S) = \frac{x(\mathcal{V})-1}{2}.$$

- Accordingly, all (E_S) inequalities hold if and only if the weight of every edge set of an odd cut $\mathcal{V}$ is at least 1.

Goal 2

Given an edge-weighted, non-negative, and even-sized graph (G, x). Determine efficiently the minimum weight odd cut.

- If Goal 2 is achievable, then it implies solving the Edmonds' polytope testing problem.

Related Cut Problems

Related Cut Problems

- If we seek $\min\limits_{\mathcal{V}\text{ cut}} x(\mathcal{V})$, this can be easily done using flow theory.

Related Cut Problems

- If we seek $\min_{\mathcal{V} \text{ cut}} x(\mathcal{V})$, this can be easily done using flow theory.
- However, if we seek $\min_{\mathcal{V}=(S,T),|S|=|T|} x(\mathcal{V})$,

Related Cut Problems

- If we seek $\min_{\mathcal{V} \text{ cut}} x(\mathcal{V})$, this can be easily done using flow theory.
- However, if we seek $\min_{\mathcal{V}=(S,T),|S|=|T|} x(\mathcal{V})$, this is an $\mathcal{NP}$-complete problem.

Related Cut Problems

- If we seek $\min_{\mathcal{V} \text{ cut}} x(\mathcal{V})$, this can be easily done using flow theory.
- However, if we seek $\min_{\mathcal{V}=(S,T),|S|=|T|} x(\mathcal{V})$, this is an $\mathcal{NP}$-complete problem.
- Thus, if we impose oddness on $\mathcal{V}$, the complexity of our question is not clear.

Related Cut Problems

- If we seek $\min_{\mathcal{V}\text{ cut}} x(\mathcal{V})$, this can be easily done using flow theory.
- However, if we seek $\min_{\mathcal{V}=(S,T),|S|=|T|} x(\mathcal{V})$, this is an $\mathcal{NP}$-complete problem.
- Thus, if we impose oddness on $\mathcal{V}$, the complexity of our question is not clear.
- If our vertex set has an odd cardinality, then one side of every cut would be odd. Thus, determining the minimum weight among odd sets would be equivalent to searching among all subsets.

The New Problem

The New Problem

So, we have reduced the fundamental problem (testing $\mathcal{MP}(G)$) to determining the minimum weight odd cut (or minimizing a weighted boundary between odd sets):

The New Problem

So, we have reduced the fundamental problem (testing $\mathcal{MP}(G)$) to determining the minimum weight odd cut (or minimizing a weighted boundary between odd sets):

Minimize	$w(\mathcal{V})$-t
subject to	$\mathcal{V}$ is an odd cut

The New Problem

So, we have reduced the fundamental problem (testing $\mathcal{MP}(G)$) to determining the minimum weight odd cut (or minimizing a weighted boundary between odd sets):

Minimize	$w(\mathcal{V})$-t
subject to	$\mathcal{V}$ is an odd cut

- Given a graph G and a non-negative weighting w, with $n = |V(G)|$ even.

The New Problem

So, we have reduced the fundamental problem (testing $\mathcal{MP}(G)$) to determining the minimum weight odd cut (or minimizing a weighted boundary between odd sets):

Minimize	$w(\mathcal{V})$-t
subject to	$\mathcal{V}$ is an odd cut

- Given a graph G and a non-negative weighting w, with $n = |V(G)|$ even.
- We seek the minimum weight (minimize the total weight of crossing edges) odd cut (both sides contain an odd number of vertices).

The New Problem

So, we have reduced the fundamental problem (testing $\mathcal{MP}(G)$) to determining the minimum weight odd cut (or minimizing a weighted boundary between odd sets):

Minimize	$w(\mathcal{V})$-t
subject to	$\mathcal{V}$ is an odd cut

- Given a graph G and a non-negative weighting w, with $n = |V(G)|$ even.
- We seek the minimum weight (minimize the total weight of crossing edges) odd cut (both sides contain an odd number of vertices).
- The initial LP formulation made the use of x natural for weighting. However, w is the most common notation for weight. We switch to it now.

The New Problem

So, we have reduced the fundamental problem (testing $\mathcal{MP}(G)$) to determining the minimum weight odd cut (or minimizing a weighted boundary between odd sets):

Minimize	$w(\mathcal{V})$-t
subject to	$\mathcal{V}$ is an odd cut

- Given a graph G and a non-negative weighting w, with $n = |V(G)|$ even.
- We seek the minimum weight (minimize the total weight of crossing edges) odd cut (both sides contain an odd number of vertices).
- The initial LP formulation made the use of x natural for weighting. However, w is the most common notation for weight. We switch to it now.
- Solving this efficiently requires introducing a new concept.

Gomory–Hu Tree

Gomory–Hu Tree

Consider a tree F on $V(G)$. Then F has $n-1$ edges, and note that deleting any edge of F separates F into two components. If e was the deleted edge, let the vertex sets be S_e and T_e. Then $\mathcal{V}_e = (S_e, T_e)$ is a cut.

Gomory–Hu Tree

Consider a tree F on $V(G)$. Then F has $n-1$ edges, and note that deleting any edge of F separates F into two components. If e was the deleted edge, let the vertex sets be S_e and T_e. Then $\mathcal{V}_e = (S_e, T_e)$ is a cut.

Definition

The tree F is a *Gomory–Hu tree* if for every $e = xy \in E(F)$ the cut (S_e, T_e) is w-optimal as an xy cut in G, meaning

$$\min_{(S,T)\ xy\ \text{cut}} w(\partial S) = w(\partial S_e)$$

Gomory–Hu Tree

Consider a tree F on $V(G)$. Then F has $n-1$ edges, and note that deleting any edge of F separates F into two components. If e was the deleted edge, let the vertex sets be S_e and T_e. Then $\mathcal{V}_e = (S_e, T_e)$ is a cut.

Definition

The tree F is a *Gomory–Hu tree* if for every $e = xy \in E(F)$ the cut (S_e, T_e) is w-optimal as an xy cut in G, meaning

$$\min_{(S,T)\ xy\ \text{cut}} w(\partial S) = w(\partial S_e)$$

The T tree is a graph on the vertex set of G. However, its edges have *nothing* to do with G. It is not necessarily a subgraph.

What's the Use of Gomory–Hu Trees?

What's the Use of Gomory–Hu Trees?

That is, a Gomory–Hu property of a tree F is composed of $n-1$ conditions. Each edge of F imposes one condition. These $n-1$ conditions are about the optimality of cuts.

What's the Use of Gomory–Hu Trees?

That is, a Gomory–Hu property of a tree F is composed of $n-1$ conditions. Each edge of F imposes one condition. These $n-1$ conditions are about the optimality of cuts.

Beyond explicit optimality in the definition, additional information can be extracted from a Gomory–Hu tree.

Actually, We Have $\binom{n}{2}$ Optimal Cuts

Actually, We Have $\binom{n}{2}$ Optimal Cuts

Lemma

Given a Gomory–Hu tree F, then for every pair of vertices $x, y \in V$, among the $n-1$ cuts determined by F, there is a minimum xy cut.

Actually, We Have $\binom{n}{2}$ Optimal Cuts

Lemma

Given a Gomory–Hu tree F, then for every pair of vertices $x, y \in V$, among the $n-1$ cuts determined by F, there is a minimum xy cut.

Let $x, y \in V$ be arbitrary. There exists a unique xy path in F. Let the edges on this path be $e_1, e_2, \ldots, e_\ell$, and the cuts associated with these edges be $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_\ell$.

Actually, We Have $\binom{n}{2}$ Optimal Cuts

Lemma

Given a Gomory–Hu tree F, then for every pair of vertices $x, y \in V$, among the $n-1$ cuts determined by F, there is a minimum xy cut.

Let $x, y \in V$ be arbitrary. There exists a unique xy path in F. Let the edges on this path be $e_1, e_2, \ldots, e_\ell$, and the cuts associated with these edges be $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_\ell$.

Observation

Each $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_\ell$ separates x and y.

Actually, We Have $\binom{n}{2}$ Optimal Cuts

Lemma

Given a Gomory–Hu tree F, then for every pair of vertices $x, y \in V$, among the $n-1$ cuts determined by F, there is a minimum xy cut.

Let $x, y \in V$ be arbitrary. There exists a unique xy path in F. Let the edges on this path be $e_1, e_2, \ldots, e_\ell$, and the cuts associated with these edges be $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_\ell$.

Observation

Each $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_\ell$ separates x and y.

Let $\mathcal{V}$ be the minimum weight cut among $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_\ell$ that is an xy cut.

Actually, We Have $\binom{n}{2}$ Optimal Cuts

Lemma

Given a Gomory–Hu tree F, then for every pair of vertices $x, y \in V$, among the $n-1$ cuts determined by F, there is a minimum xy cut.

Let $x, y \in V$ be arbitrary. There exists a unique xy path in F. Let the edges on this path be $e_1, e_2, \ldots, e_\ell$, and the cuts associated with these edges be $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_\ell$.

Observation

Each $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_\ell$ separates x and y.

Let $\mathcal{V}$ be the minimum weight cut among $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_\ell$ that is an xy cut.

Stronger Lemma

$\mathcal{V}$ is an optimal xy cut.

Proof of the Stronger Lemma

Proof of the Stronger Lemma

Assume (for contradiction) that $\mathcal{V}_{opt}$ is a minimum weight xy cut, and

$$w(\mathcal{V}_{opt}) < w(\mathcal{V})$$

Proof of the Stronger Lemma

Assume (for contradiction) that $\mathcal{V}_{opt}$ is a minimum weight xy cut, and

$$w(\mathcal{V}_{opt}) < w(\mathcal{V})$$

Then there exists an edge e_i such that its endpoints belong to different sides of V_{opt}.

Proof of the Stronger Lemma

Assume (for contradiction) that $\mathcal{V}_{opt}$ is a minimum weight xy cut, and

$$w(\mathcal{V}_{opt}) < w(\mathcal{V})$$

Then there exists an edge e_i such that its endpoints belong to different sides of V_{opt}.

But since F is a Gomory–Hu tree, $\mathcal{V}_i$ is an optimal cut separating the endpoints of e_i.

Proof of the Stronger Lemma

Assume (for contradiction) that $\mathcal{V}_{opt}$ is a minimum weight xy cut, and

$$w(\mathcal{V}_{opt}) < w(\mathcal{V})$$

Then there exists an edge e_i such that its endpoints belong to different sides of V_{opt}.

But since F is a Gomory–Hu tree, $\mathcal{V}_i$ is an optimal cut separating the endpoints of e_i.

Thus,

$$w(\mathcal{V}_{opt}) \geq w(\mathcal{V}_i) \geq w(\mathcal{V}),$$

a contradiction.

Proof of the Stronger Lemma

Assume (for contradiction) that $\mathcal{V}_{opt}$ is a minimum weight xy cut, and

$$w(\mathcal{V}_{opt}) < w(\mathcal{V})$$

Then there exists an edge e_i such that its endpoints belong to different sides of V_{opt}.

But since F is a Gomory–Hu tree, $\mathcal{V}_i$ is an optimal cut separating the endpoints of e_i.

Thus,

$$w(\mathcal{V}_{opt}) \geq w(\mathcal{V}_i) \geq w(\mathcal{V}),$$

a contradiction.

This establishes the claim and hence the lemma.

Further Information in a Gomory–Hu Tree

Further Information in a Gomory–Hu Tree

Let (G, w) and F be given as a Gomory-Hu tree. This defines $n - 1$ cuts, each pair has an optimal separator.

Further Information in a Gomory–Hu Tree

Let (G, w) and F be given as a Gomory-Hu tree. This defines $n - 1$ cuts, each pair has an optimal separator.

We assumed $|V|$ is even: There are both even-even and odd-odd cuts. (If $|V|$ were odd, then all cuts would be even-odd.)

Further Information in a Gomory–Hu Tree

Let (G, w) and F be given as a Gomory-Hu tree. This defines $n-1$ cuts, each pair has an optimal separator.

We assumed $|V|$ is even: There are both even-even and odd-odd cuts. (If $|V|$ were odd, then all cuts would be even-odd.)

Remark

Among the $n-1$ cuts determined by F, there must be an odd-odd cut. Indeed, an edge adjacent to a leaf corresponds to a cut where one side has 1 vertex and the other has $n-1$.

Further Information in a Gomory–Hu Tree

Let (G, w) and F be given as a Gomory-Hu tree. This defines $n-1$ cuts, each pair has an optimal separator.

We assumed $|V|$ is even: There are both even-even and odd-odd cuts. (If $|V|$ were odd, then all cuts would be even-odd.)

Remark

Among the $n-1$ cuts determined by F, there must be an odd-odd cut. Indeed, an edge adjacent to a leaf corresponds to a cut where one side has 1 vertex and the other has $n-1$.

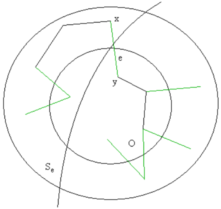
Theorem

Among the $n-1$ cuts implied by F, the smallest weight odd-odd cut exists.

Proof of the Theorem

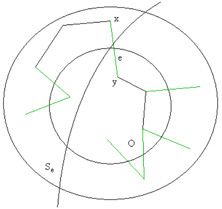
Proof of the Theorem

Let O be an optimal odd set.



Proof of the Theorem

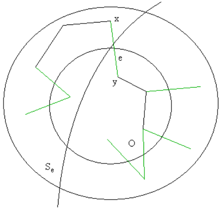
Let O be an optimal odd set.



$V(G) = V(F)$. For every $e \in \partial_F O$, consider the cut $\mathcal{V}_e = (S_e, T_e)$ determined by F, where S_e contains the endpoint of e outside O.

Proof of the Theorem

Let O be an optimal odd set.

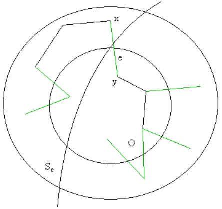


$V(G) = V(F)$. For every $e \in \partial_F O$, consider the cut $\mathcal{V}_e = (S_e, T_e)$ determined by F, where S_e contains the endpoint of e outside O.

We use the notation $S_{\overrightarrow{e}}$ where $e \in E(F)$. e determines (with F) a cut, S denotes the side referred to as S *set* by this cut.

Proof of the Theorem

Let O be an optimal odd set.



$V(G) = V(F)$. For every $e \in \partial_F O$, consider the cut $\mathcal{V}_e = (S_e, T_e)$ determined by F, where S_e contains the endpoint of e outside O.

We use the notation $S_{\overrightarrow{e}}$ where $e \in E(F)$. e determines (with F) a cut, S denotes the side referred to as S *set* by this cut. That is, an edge of the tree F defines a cut. The orientation shows which side is referred to as S *set*.

Proof of the Stronger Theorem

$$\sum_{e\in\partial_F O} |S_{\overrightarrow{e}}| \underset{\pmod 2}{\equiv} \sum_{\substack{x\in O,\ e\in E(F),\\ \overrightarrow{e} \text{ points out of } x}} |S_{\overrightarrow{e}}| =$$

$$= \sum_{x\in O} \sum_{\substack{e\in \mathrm{E}(F)\\ \overrightarrow{e}=\overrightarrow{xu}}} |S_{\overrightarrow{e}}| = \sum_{x\in O} (|V|-1) \underset{\pmod 2}{\equiv} 1.$$

Proof of the Stronger Theorem

$$\sum_{e\in\partial_F O} |S_{\overrightarrow{e}}| \underset{(\text{mod } 2)}{\equiv} \sum_{\substack{x\in O,\ e\in E(F),\\ \overrightarrow{e} \text{ points out of } x}} |S_{\overrightarrow{e}}| =$$

$$= \sum_{x\in O} \sum_{\substack{e\in \mathrm{E}(F)\\ \overrightarrow{e}=\overrightarrow{xu}}} |S_{\overrightarrow{e}}| = \sum_{x\in O}(|V|-1) \underset{(\text{mod } 2)}{\equiv} 1.$$

The first congruence holds because the extra terms in the sum come in pairs (one for each e edge adjacent to O), and each pair contributes $|V|$, which is even.

Proof of the Stronger Theorem

$$\sum_{e\in\partial_F O} |S_{\overrightarrow{e}}| \underset{(\text{mod } 2)}{\equiv} \sum_{\substack{x\in O,\ e\in E(F),\\ \overrightarrow{e} \text{ points out of } x}} |S_{\overrightarrow{e}}| =$$

$$= \sum_{x\in O} \sum_{\substack{e\in \mathrm{E}(F)\\ \overrightarrow{e}=\overrightarrow{xu}}} |S_{\overrightarrow{e}}| = \sum_{x\in O} (|V|-1) \underset{(\text{mod } 2)}{\equiv} 1.$$

The first congruence holds because the extra terms in the sum come in pairs (one for each e edge adjacent to O), and each pair contributes $|V|$, which is even.

The second congruence holds because an odd number of odd numbers is being summed up.

Where Are We?

Where Are We?

- Introduced the concept of Gomory–Hu trees.

Where Are We?

- Introduced the concept of Gomory–Hu trees.
- If we have a Gomory–Hu tree in a graph with an even number of vertices, then it's easy to extract an optimal odd cut.

Where Are We?

- Introduced the concept of Gomory–Hu trees.
- If we have a Gomory–Hu tree in a graph with an even number of vertices, then it's easy to extract an optimal odd cut.
- This allows us to test whether a nonnegative weighting belongs to $\mathcal{MP}(G)$, provided each vertex has a weight sum of 1.

Where Are We?

- Introduced the concept of Gomory–Hu trees.
- If we have a Gomory–Hu tree in a graph with an even number of vertices, then it's easy to extract an optimal odd cut.
- This allows us to test whether a nonnegative weighting belongs to $\mathcal{MP}(G)$, provided each vertex has a weight sum of 1.
- Thus, every vector in $\mathbb{R}^{E(G)}$ can be efficiently tested.

Where Are We?

- Introduced the concept of Gomory–Hu trees.
- If we have a Gomory–Hu tree in a graph with an even number of vertices, then it's easy to extract an optimal odd cut.
- This allows us to test whether a nonnegative weighting belongs to $\mathcal{MP}(G)$, provided each vertex has a weight sum of 1.
- Thus, every vector in $\mathbb{R}^{E(G)}$ can be efficiently tested.

3rd Goal $\equiv$ Gomory–Hu Theorem

For every G, w, there exists a Gomory–Hu tree F, and one can be computed in polynomial time.

Where Are We?

- Introduced the concept of Gomory–Hu trees.
- If we have a Gomory–Hu tree in a graph with an even number of vertices, then it's easy to extract an optimal odd cut.
- This allows us to test whether a nonnegative weighting belongs to $\mathcal{MP}(G)$, provided each vertex has a weight sum of 1.
- Thus, every vector in $\mathbb{R}^{E(G)}$ can be efficiently tested.

3rd Goal $\equiv$ Gomory–Hu Theorem

For every G, w, there exists a Gomory–Hu tree F, and one can be computed in polynomial time.

Consequence

Given a graph G and $w \in \mathbb{Q}^{\mathrm{E}(G)}$, there exists a polynomial-time algorithm to decide whether w is an element of $\mathcal{MP}(G)$; if not, it provides an Edmonds condition violated by w.

Break Time

Introductory Lemma

Introductory Lemma

Lemma

The mapping $f = w \circ \partial : \mathcal{P}(V) \to \mathbb{R}_+$

(i) is symmetric, i.e., $f(S) = f(\overline{S}) \quad \forall S \subseteq V$,

(ii) is submodular, i.e.,
$f(S) + f(T) \geq f(S \cap T) + f(S \cup T) \quad \forall S, T \subseteq V$,

(iii) is posimodular, i.e.,
$f(S) + f(T) \geq f(S \setminus T) + f(T \setminus S) \quad \forall S, T \subseteq V$.

Proof

Proof

(i): Symmetry is clear since $\partial S = \partial \overline{S}$.

Proof

(i): Symmetry is clear since $\partial S = \partial \overline{S}$.

(ii): Submodularity holds: Summing weights on both sides. In the left expression, each edge is counted at least as many times as in the right expression (by case analysis). Since weights are nonnegative, the inequality holds.

Proof

(i): Symmetry is clear since $\partial S = \partial \overline{S}$.

(ii): Submodularity holds: Summing weights on both sides. In the left expression, each edge is counted at least as many times as in the right expression (by case analysis). Since weights are nonnegative, the inequality holds.

(iii): Posimodularity follows from the previous two properties:

$$f(S)+f(T)=f(S)+f(\overline{T})\geq f(S\cap\overline{T})+f(S\cup\overline{T})=f(S\setminus T)+f(\overline{T\setminus S})=$$

$$=f(S\setminus T)+f(T\setminus S).$$

The Main Lemma

Main Lemma

Let $\mathcal{V}$ be an xy optimal cut and x', y' vertices. Then there exists a $\mathcal{V}'$ $x'y'$ cut, which is $x'y'$ optimal, and $\mathcal{V}$ and $\mathcal{V}'$ are non-crossing cuts.

The Main Lemma

Main Lemma

Let $\mathcal{V}$ be an xy optimal cut and x', y' vertices. Then there exists a $\mathcal{V}'$ $x'y'$ cut, which is $x'y'$ optimal, and $\mathcal{V}$ and $\mathcal{V}'$ are non-crossing cuts.

Definition

The cuts (S, T) and (S', T') are crossing cuts if
$S \cap S', S \cap T', T \cap S', T \cap T' \neq \emptyset$.

The Main Lemma

Main Lemma

Let $\mathcal{V}$ be an xy optimal cut and x', y' vertices. Then there exists a $\mathcal{V}'$ $x'y'$ cut, which is $x'y'$ optimal, and $\mathcal{V}$ and $\mathcal{V}'$ are non-crossing cuts.

Definition

The cuts (S, T) and (S', T') are crossing cuts if $S \cap S', S \cap T', T \cap S', T \cap T' \neq \emptyset$.

This is equivalent to saying that cuts (S, T) and (S', T') are non-crossing cuts if $S \subseteq S'$ or $S' \subseteq S$ or $S \cap S' = \emptyset$.

Proof of the Main Lemma

Proof of the Main Lemma

Let $\mathcal{V}'$ be any $x'y'$ optimal cut. Suppose $\mathcal{V}$ and $\mathcal{V}'$ are crossing cuts. Two cases arise.

Proof of the Main Lemma

Let $\mathcal{V}'$ be any $x'y'$ optimal cut. Suppose $\mathcal{V}$ and $\mathcal{V}'$ are crossing cuts. Two cases arise.

Case 1: x', y' are on the same side of $\mathcal{V}$ (suppose this is the x side). Let x' and y' be renamed such that x' falls on the same side of x as x'.

Proof of the Main Lemma

Let $\mathcal{V}'$ be any $x'y'$ optimal cut. Suppose $\mathcal{V}$ and $\mathcal{V}'$ are crossing cuts. Two cases arise.

Case 1: x', y' are on the same side of $\mathcal{V}$ (suppose this is the x side). Let x' and y' be renamed such that x' falls on the same side of x as x'.

Case 2: x', y' are on different sides of $\mathcal{V}$ (suppose x' falls on the side of x).

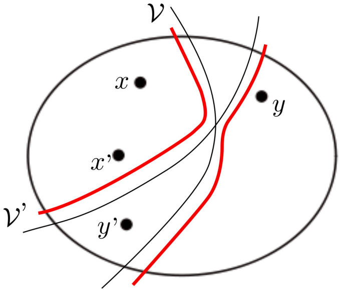
Proof of the Main Lemma: Case 1

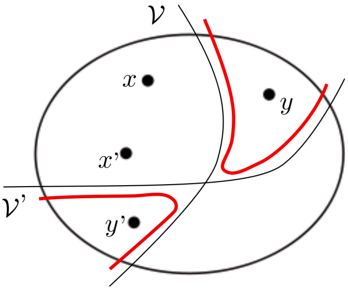
Proof of the Main Lemma: Case 1

Two more subcases are possible here (diagrams above) depending on whether $\mathcal{V}'$ cuts x and y or not.

Proof of the Main Lemma: Case 1

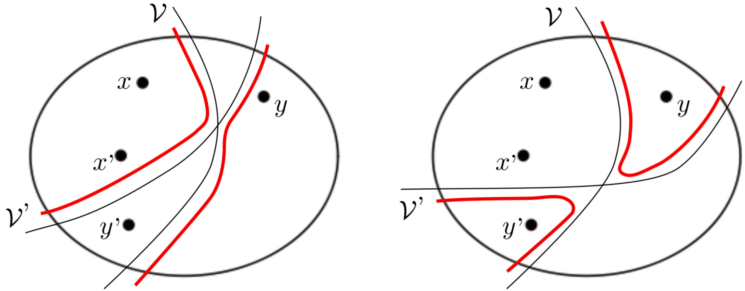
Two more subcases are possible here (diagrams above) depending on whether $\mathcal{V}'$ cuts x and y or not.





Proof of the Main Lemma: Case 1

Two more subcases are possible here (diagrams above) depending on whether $\mathcal{V}'$ cuts x and y or not.



Among the cuts marked in red on the figure, one is an xy cut (let this be $\mathcal{V}^*$), and the other is an $x'y'$ cut (let this be $\mathcal{V}^{**}$).

Proof of the Main Lemma: Case 1 (Continued)

Proof of the Main Lemma: Case 1 (Continued)

Then by submodularity, or posimodularity (depending on which case we are in and how we label the S sides), we have

$$f(\mathcal{V}) + f(\mathcal{V}') \geq f(\mathcal{V}^*) + f(\mathcal{V}^{**})$$

Proof of the Main Lemma: Case 1 (Continued)

Then by submodularity, or posimodularity (depending on which case we are in and how we label the S sides), we have

$$f(\mathcal{V}) + f(\mathcal{V}') \geq f(\mathcal{V}^*) + f(\mathcal{V}^{**})$$

However, equality must hold, since $\mathcal{V}$ and $\mathcal{V}'$ were optimal/minimal cuts and "performing their task" $\mathcal{V}^*$ and $\mathcal{V}^{**}$ also do. So,

$$f(\mathcal{V}') = f(\mathcal{V}^{**})$$

and $\mathcal{V}^{**}$ does not cross $\mathcal{V}$, thus $\mathcal{V}^{**}$ fulfills the desired property.

Proof of the Main Lemma: Case 2

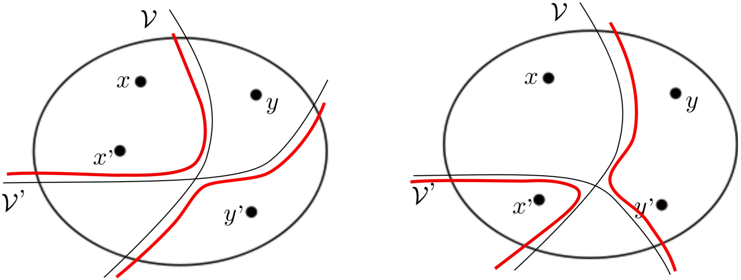
Proof of the Main Lemma: Case 2

If $\mathcal{V}'$ separates x and y, then $\mathcal{V}'$ can be chosen as $\mathcal{V}$, and we are done (a cut does not cross itself).

Proof of the Main Lemma: Case 2

If $\mathcal{V}'$ separates x and y, then $\mathcal{V}'$ can be chosen as $\mathcal{V}$, and we are done (a cut does not cross itself).

If $\mathcal{V}'$ does not separate x and y, then similarly as in the Case 1, another $x'y'$ optimal cut can be found that does not cross $\mathcal{V}$:



Figure

Start of Our Algorithm: Bisection

Start of Our Algorithm: Bisection

Let's arbitrarily choose two vertices, let them be x and y. Determine the optimal xy cut. Let this be (S, T) such that $x \in S$ and $y \in T$.

Start of Our Algorithm: Bisection

Let's arbitrarily choose two vertices, let them be x and y. Determine the optimal xy cut. Let this be (S, T) such that $x \in S$ and $y \in T$.

We *bisect* G: Let G/T be the graph whose vertices are the vertices in S plus one meta-vertex m_T, representing T, and edges are the edges within S plus the edges incident to ∂S, where each edge from ∂S is connected to m_T instead of its original endpoint in T.

Start of Our Algorithm: Bisection

Let's arbitrarily choose two vertices, let them be x and y. Determine the optimal xy cut. Let this be (S, T) such that $x \in S$ and $y \in T$.

We *bisect* G: Let G/T be the graph whose vertices are the vertices in S plus one meta-vertex m_T, representing T, and edges are the edges within S plus the edges incident to ∂S, where each edge from ∂S is connected to m_T instead of its original endpoint in T.

The definition of G/S is similar, except here m_S is the new meta-vertex.

Bisection on a picture

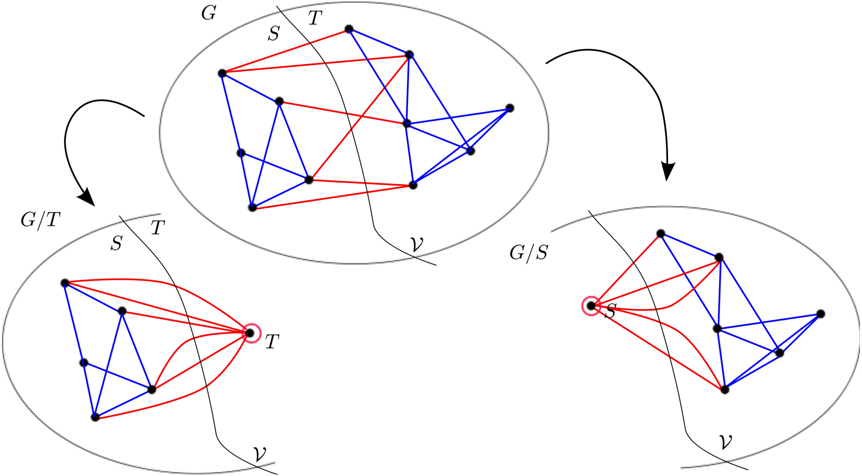


Figure: The initial bisection

The Algorithm: Recursive Bisection

The Algorithm: Recursive Bisection

Finally, we connect the two parts with one meta-edge, fitting onto m_T and m_S.

The Algorithm: Recursive Bisection

Finally, we connect the two parts with one meta-edge, fitting onto m_T and m_S.

Then we continue bisecting the two parts until we can:

The Algorithm: Recursive Bisection

Finally, we connect the two parts with one meta-edge, fitting onto m_T and m_S.

Then we continue bisecting the two parts until we can:

Gomory–Hu Algorithm: Recursive Bisection Algorithm

Perform the initial bisection.

While each part contains at least 2 original vertices, **repeat** the bisection (the vertices x and y defining the bisection are always original vertices).

The Gomory–Hu Algorithm in a Figure

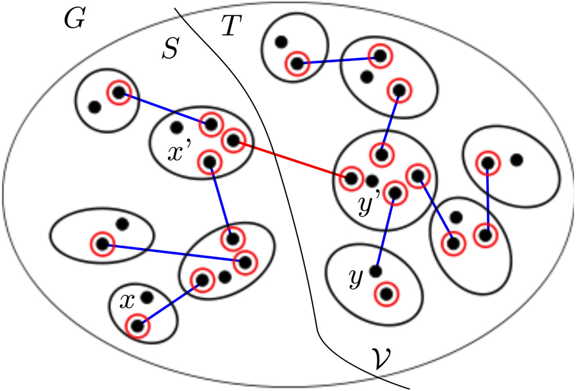


Figure: The meta-vertices are circled in red, and the edges passing through them are the meta-edges. x and y define the original $\mathcal{V}$ cut. Only one edge from the computed tree passes through this cut. This edge is not necessarily the xy edge.

What's the Output?

What's the Output?

- At the end of the recursion, we have bisected the graph into parts such that each part contains exactly one original vertex.

What's the Output?

- At the end of the recursion, we have bisected the graph into parts such that each part contains exactly one original vertex.
- Thus, the parts computed by the algorithm are identified with the vertices.

What's the Output?

- At the end of the recursion, we have bisected the graph into parts such that each part contains exactly one original vertex.
- Thus, the parts computed by the algorithm are identified with the vertices.
- The meta-edges connect different parts corresponding to different vertices.

What's the Output?

- At the end of the recursion, we have bisected the graph into parts such that each part contains exactly one original vertex.
- Thus, the parts computed by the algorithm are identified with the vertices.
- The meta-edges connect different parts corresponding to different vertices.
- So, the meta-edges can be viewed as the edges between the original vertices.

What's the Output?

- At the end of the recursion, we have bisected the graph into parts such that each part contains exactly one original vertex.
- Thus, the parts computed by the algorithm are identified with the vertices.
- The meta-edges connect different parts corresponding to different vertices.
- So, the meta-edges can be viewed as the edges between the original vertices.
- These meta-edges constitute the computed graph (on the vertex set of G).

Where's the Tree? ... Found It!

Where's the Tree? ... Found It!

Observation

The graph computed by the Gomory–Hu algorithm is a tree.

Where's the Tree? ... Found It!

Observation

The graph computed by the Gomory–Hu algorithm is a tree.

Indeed:

Where's the Tree? ... Found It!

Observation

The graph computed by the Gomory–Hu algorithm is a tree.

Indeed:

- We compute an $n-1$ edge graph on n vertices.

Where's the Tree? ... Found It!

Observation

The graph computed by the Gomory–Hu algorithm is a tree.

Indeed:

- We compute an $n-1$ edge graph on n vertices.
- It is clear from the recursion (and can be formally proved by induction) that the output is a connected graph.

Correctness of the Gomory–Hu Algorithm

Correctness of the Gomory–Hu Algorithm

Theorem

The tree computed by the Gomory–Hu algorithm is a Gomory–Hu tree.

Correctness of the Gomory–Hu Algorithm

Theorem

The tree computed by the Gomory–Hu algorithm is a Gomory–Hu tree.

The assertion is to show that each cut defined by a meta-edge is an optimal separation of its endpoints. That is, we need to prove $n-1$ statements.

Cuts of Our Graph Compared to the Base Cut

Cuts of Our Graph Compared to the Base Cut

Let $\mathcal{V}$ be the cut corresponding to the initial bisection. The cuts of G can be grouped as follows:

1. Crossing cuts with $\mathcal{V}$,
2. Non-crossing cuts with $\mathcal{V}'$:
   a) $\mathcal{V}' = \mathcal{V}$
   b) $\mathcal{V}' = (S', T')$, $S' \subsetneq S$
   c) $\mathcal{V}' = (S', T')$, $T' \subsetneq T$

Cuts of Our Graph Compared to the Base Cut

Let $\mathcal{V}$ be the cut corresponding to the initial bisection. The cuts of G can be grouped as follows:

1. Crossing cuts with $\mathcal{V}$,
2. Non-crossing cuts with $\mathcal{V}'$:
   a) $\mathcal{V}' = \mathcal{V}$
   b) $\mathcal{V}' = (S', T')$, $S' \subsetneq S$
   c) $\mathcal{V}' = (S', T')$, $T' \subsetneq T$

Observation

The cuts of G/T can be paired with $\mathcal{V}$ and cuts of type 2.b), while cuts of G/S can be paired with $\mathcal{V}$ and cuts of type 2.c). All cuts in G/T and G/S are present in either $G \backslash T$ or $G \backslash S$ (and nowhere else).

Proving Correctness: Induction

Proving Correctness: Induction

- We proceed with induction, proving recursively.

Proving Correctness: Induction

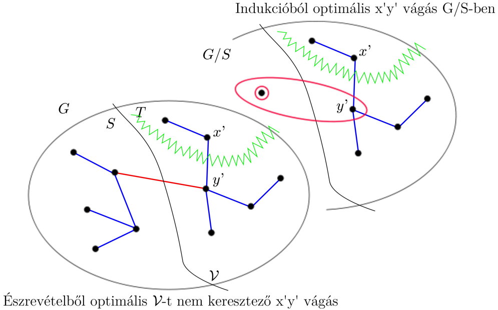
- We proceed with induction, proving recursively.
- The base case is obvious.

Proving Correctness: Induction

- We proceed with induction, proving recursively.
- The base case is obvious.
- From the $n-1$ assertions, $(|S|-1)+(|T|-1)=|V|-2=n-2$ edges come from the Gomory–Hu tree of $G\setminus S$ and $G\setminus T$. According to the recursion, these apply to the graphs $G\setminus S$ and $G\setminus T$, so we have optimality there.

Proving Correctness: Induction

- We proceed with induction, proving recursively.
- The base case is obvious.
- From the $n-1$ assertions, $(|S|-1)+(|T|-1)=|V|-2=n-2$ edges come from the Gomory–Hu tree of $G\setminus S$ and $G\setminus T$. According to the recursion, these apply to the graphs $G\setminus S$ and $G\setminus T$, so we have optimality there.
-



The Main Edge

The Main Edge

The only issue is with the red edge (arising from the initial bisection).

The Main Edge

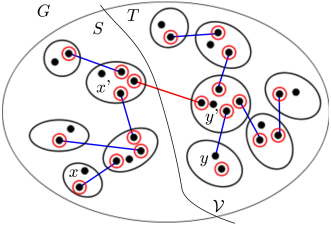
The only issue is with the red edge (arising from the initial bisection).

The crosscut of the Gomory–Hu tree crosses $x'y'$. This is the only edge in F where we don't yet know the assertion. The problem is that we chose an optimal xy cut for the initial bisection. However, the bisection led to an $x'y'$ crosscut, and it's possible that $x \neq x'$ and $y \neq y'$.

The Main Edge

The only issue is with the red edge (arising from the initial bisection).

The crosscut of the Gomory–Hu tree crosses $x'y'$. This is the only edge in F where we don't yet know the assertion. The problem is that we chose an optimal xy cut for the initial bisection. However, the bisection led to an $x'y'$ crosscut, and it's possible that $x \neq x'$ and $y \neq y'$.



The meta-vertices are circled in red, and the edges passing through them are the meta-edges. x and y define the original $\mathcal{V}$ cut. Only one edge

The Last Remaining Assertion

The Last Remaining Assertion

Assertion

The cut of $x'y'$ edge $(S, T) = \mathcal{V}$ (which was chosen as the w-optimal xy cut) is also a w-optimal $x'y'$ cut.

The Last Remaining Assertion

Assertion

The cut of $x'y'$ edge $(S, T) = \mathcal{V}$ (which was chosen as the w-optimal xy cut) is also a w-optimal $x'y'$ cut.

We prove this by contradiction. Assume there exists a $\mathcal{V}'$ w-optimal $x'y'$ cut such that $w(\mathcal{V}') < w(\mathcal{V})$.

The Last Remaining Assertion

Assertion

The cut of $x'y'$ edge $(S, T) = \mathcal{V}$ (which was chosen as the w-optimal xy cut) is also a w-optimal $x'y'$ cut.

We prove this by contradiction. Assume there exists a $\mathcal{V}'$ w-optimal $x'y'$ cut such that $w(\mathcal{V}') < w(\mathcal{V})$.

From the main lemma, we know that $\mathcal{V}' = (S', T')$ can be chosen such that $S' \subset S$ or $T' \subset T$. We may assume $S' \subsetneq S$.

Continuation of the Last Remaining Assertion Proof

Continuation of the Last Remaining Assertion Proof

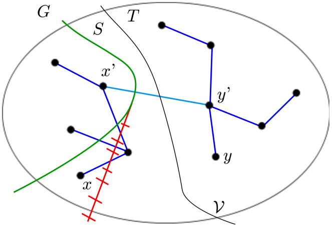
Observation

It cannot be that $\mathcal{V}'$ doesn't separate x and x'.

Continuation of the Last Remaining Assertion Proof

Observation

It cannot be that $\mathcal{V}'$ doesn't separate x and x'.

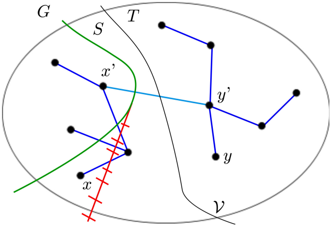


Figure

Continuation of the Last Remaining Assertion Proof

Observation

It cannot be that $\mathcal{V}'$ doesn't separate x and x'.



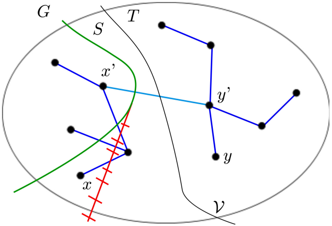
Figure

That is, it cannot be that x and x' are on the same side while the entire T is on the other side, including y.

Continuation of the Last Remaining Assertion Proof

Observation

It cannot be that $\mathcal{V}'$ doesn't separate x and x'.



Figure

That is, it cannot be that x and x' are on the same side while the entire T is on the other side, including y.

In this case, $\mathcal{V}'$ would be an xy cut, which is a contradiction. $\mathcal{V}'$ separates x and x'.

Continuation of the Last Remaining Assertion Proof

Continuation of the Last Remaining Assertion Proof

Observation

G/T has a w-optimal xx' cut in its Gomory–Hu tree.

Continuation of the Last Remaining Assertion Proof

Observation

G/T has a w-optimal xx' cut in its Gomory–Hu tree.

The $\mathcal{V}''$ cut has x on one side, x' and m_T on the other side (where x' and m_T stick together after bisections, hence the F crosscut fits onto x').

Continuation of the Last Remaining Assertion Proof

Observation

G/T has a w-optimal xx' cut in its Gomory–Hu tree.

The $\mathcal{V}''$ cut has x on one side, x' and m_T on the other side (where x' and m_T stick together after bisections, hence the F crosscut fits onto x').

By our initial observation, $\mathcal{V}''$ corresponds to a cut $\widetilde{\mathcal{V}''}$ in G. From the above, this is an xy cut, with weight smaller than $\mathcal{V}$'s weight, which contradicts.

Continuation of the Last Remaining Assertion Proof

Observation

G/T has a w-optimal xx' cut in its Gomory–Hu tree.

The $\mathcal{V}''$ cut has x on one side, x' and m_T on the other side (where x' and m_T stick together after bisections, hence the F crosscut fits onto x').

By our initial observation, $\mathcal{V}''$ corresponds to a cut $\widetilde{\mathcal{V}''}$ in G. From the above, this is an xy cut, with weight smaller than $\mathcal{V}$'s weight, which contradicts.

The contradiction proves the assertion, the only missing piece in proving the correctness of the Gomory–Hu algorithm.

Summary

Gomory–Hu Theorem

For every (G, w), there exists a Gomory–Hu tree F, which can be computed by determining $n-1$ minimal st cuts, achievable by applying the flow algorithm $n-1$ times. Specifically, the Gomory–Hu algorithm is polynomial.

This is the End!

Thnak you for your attention!