

Sparsest cut

Alan Chang

Mathcamp 2021 Week 1

Contents

1	Introduction	3
1.1	Course blurb	3
1.2	References	3
1.3	Guide for these notes	4
2	Day 1	4
2.1	Introduction	4
2.2	Graphs	4
2.3	Cuts	5
2.3.1	Minimum cut (or min-cut)	6
2.3.2	Maximum cut (or max-cut)	6
2.3.3	Sparsest cut	6
2.4	Algorithms and running-time efficiency	7
2.5	Linear programming	8
2.5.1	Algorithms	10
2.6	A reformulation of min-cut	11
2.7	The LP relaxation of min-cut – part1	12
3	Day 1 exercises	12
4	Day 2	15
4.1	The LP relaxation of min-cut – part 2	15
4.2	From the LP relaxation back to min-cut, the easy way	16
4.3	Min-cut and its LP relaxation have the same minimum value	17
5	Day 2 exercises	20

6	Day 3	21
6.1	Rounding the LP-relaxation of min-cut	21
6.2	Summary of our min-cut algorithm	22
6.3	Max-cut	22
6.3.1	A randomized $\frac{1}{2}$ -approximation algorithm for max-cut	23
6.4	Metrics and metric spaces	24
6.5	The spaces ℓ_1^n and ℓ_1	25
6.6	Pseudometrics	25
7	Day 3 exercises	26
8	Day 4	28
8.1	Examples of pseudometrics	28
8.2	Cut-cone representation of ℓ_1 pseudometrics	29
8.3	Sparsest cut	30
8.4	A reformulation of sparsest cut	31
8.5	Three minimization problems related to sparsest cut	33
9	Day 4 exercises	34
10	Day 5	35
10.1	Metric embeddings	35
10.2	Rounding the Leighton–Rao LP	37
10.3	Bourgain’s theorem	37
10.4	Proof of the $O(\log V)$ -approximation algorithm	38
10.5	Sharpness of this approximation algorithm	39
10.6	End of class remark	41
10.7	Bonus: Better approximation algorithms for sparsest cut	42
11	Day 5 exercises	43
12	Acknowledgments	44

1 Introduction

1.1 Course blurb

The sparsest cut problem asks the following: Given a (finite simple undirected) graph G , find a subset S of the vertices that minimizes

$$\frac{\text{number of edges with one endpoint in } S \text{ and one endpoint in } S^c}{\min(\text{number of vertices in } S, \text{number of vertices in } S^c)}. \quad (1.1)$$

The point of the denominator is to try to balance the sizes of S and S^c . You can think of sparsest cut as a discrete analogue of the isoperimetric problem.

Is there an efficient way to find the minimum? The short answer is “probably not,” but fortunately, there is an efficient way to approximate the solution: if G has N vertices, then there is an algorithm that gives you a subset S which does not necessarily attain the minimum, but it will not be too far off. In particular, it will be within a factor of $\log N$ of the minimum.

In this class, we will introduce the algorithm and then discuss why this algorithm produces good approximations. A key part of the proof is connecting the sparsest cut problem to a fundamental question about discrete geometric spaces. If there is time, we will briefly mention how the discrete 5-dimensional Heisenberg group plays a role in a better (but more difficult) approximation algorithm for sparsest cut. These are just some of the many examples of the connections between theoretical computer science and metric geometry.

Prerequisites: There are no official prerequisites for this class. In particular, you do not need to have any background in theoretical computer science or graph theory. We will introduce many simple but new ideas rather quickly, which is why this is a 3-chili class.

Class format: I will give lectures using Google Jamboard, and I will screen-share Jamboard from my tablet. If you’d like, you can open up Jamboard in your browser to browse through previous slides.

1.2 References

1. Matoušek, Lectures in discrete geometry, Chapter 15. The book was published in 2002. A revised version of Chapter 15 is available on the author’s webpage: <https://kam.mff.cuni.cz/~matousek/dg.html>. This version covers new developments up to 2005.
2. Yury Makarychev’s notes for his Computational and Metric Geometry course: <https://home.ttic.edu/~yury/courses/geom2019/>
3. Luca Trevisan’s notes for his Graph Partitioning and Expanders course: <https://lucatrevisan.wordpress.com/lecture-notes/>

1.3 Guide for these notes

For the exercises, here is the difficulty scale:

- 🐣 : easy
- 🐣🐣 : medium
- 🐣🐣🐣 : hard

The words “easy,” “medium,” and “hard” are not well-defined. Don’t be afraid of difficult problems! It’s by struggling with these exercises that you really learn.

Things labeled “Fun fact” are not needed for the class.

2 Day 1

2.1 Introduction

This class touches on many area of mathematics. First, there is graph theory, since our main object of study will be graphs. Next, since we are studying some algorithm problems on graphs, we will see some ideas from theoretic computer science.

It is perhaps not surprising to see graph theory and theoretic computer science appearing together, since there are many algorithm questions for graphs. What is perhaps much more surprising is that to understand the sparsest cut problem, we will need to study metric geometry (the geometry of metric spaces).

This is not intended to be a class on graph theory, theoretic computer science, or metric geometry. Instead, the main goal of this class is to see how these areas are related through the sparsest cut problem.

Because of time, there will be many details we will have to skip. We will state several results without proving them. Some of you may have the idea that mathematics is about proving everything rigorously. While that is one part of math, it is not all of it. See <https://terrytao.wordpress.com/career-advice/theres-more-to-mathematics-than-rigour-and-proofs/> for a nice perspective of the role of rigor in mathematics.

2.2 Graphs

This is a quick introduction to graphs. See Wikipedia [https://en.wikipedia.org/wiki/Graph_\(discrete_mathematics\)](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)) for a more detailed introduction.

A *graph* is a mathematical object with (1) *vertices* and (2) *edges* between the vertices. We usually call a graph G , and we denote its set of vertices by V and its set of edges by E . To abbreviate this we often say “Let $G = (V, E)$ be a graph.”

For us, all graphs will be finite, simple, undirected. The meaning of these words are:

1. finite: There are finitely many edges.
2. simple: You cannot have multiple edges with the same pair of endpoints.
3. undirected: The edges are not directed.

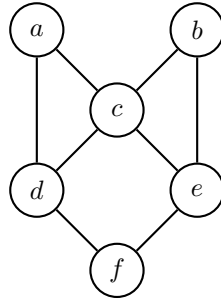


Figure 1: A finite, simple, undirected graph

Example 2.1. The vertex set and edge set for the graph shown in Figure 1 are

$$V = \{a, b, c, d, e, f\} \quad (2.1)$$

$$E = \{ac, ad, bc, be, cd, ce, df, ef\} \quad (2.2)$$

Note that we use uv to denote the edge between u and v . Since the graph is undirected, uv means the same thing as vu .

2.3 Cuts

[https://en.wikipedia.org/wiki/Cut_\(graph_theory\)](https://en.wikipedia.org/wiki/Cut_(graph_theory))

Let $G = (V, E)$ be a graph. A *cut* of G is a partition of the vertices V into two disjoint subsets. We can just think of a cut as a subset $S \subset V$. The corresponding partition is $\{S, S^c\}$. Note that S and S^c refer to the same cut.

We define

$$E(S, S^c) = \{e \in E : e \text{ has one endpoint in } S \text{ and one endpoint in } S^c\} \quad (2.3)$$

We call $E(S, S^c)$ the *cut-set* of S . The edges in $E(S, S^c)$ are said to *cross* the cut.

Now we introduce three natural questions we can ask about cuts: min-cut, max-cut, and sparsest cut.

2.3.1 Minimum cut (or min-cut)

Given:

- a graph $G = (V, E)$
- two distinct vertices $s, t \in V$. (s is often called the “source,” and t is often called the “sink” or the “target”)

Find: a cut S such that

- $s \in S$ and $t \in S^c$
- $|E(S, S^c)|$ is minimized.

(If we didn't have the first condition, we could simply take $S = \emptyset$.)

2.3.2 Maximum cut (or max-cut)

Given:

- a graph $G = (V, E)$

Find: a cut S such that

- $|E(S, S^c)|$ is maximized.

2.3.3 Sparsest cut

Given:

- a graph $G = (V, E)$

Find: a cut S such that

- $\frac{|E(S, S^c)|}{\min(|S|, |S^c|)}$ is minimized.

The point of the denominator is to try to balance the sizes of S and S^c . You are penalized if $|S|$ or $|S^c|$ is too small. You can think of sparsest cut as a discrete analogue of the isoperimetric problem.

Remark 2.2. You may wonder why we are looking at all of these when the class is called “Sparsest cut.” The reason is that if we jump straight into sparsest cut, we will need to introduce many new ideas at the same time:

- linear programming relaxations
- rounding
- approximation algorithms
- metric spaces

It is better to introduce these new ideas more gradually, so we do that by studying min-cut and max-cut first. By studying the min-cut problem, we will learn about linear programming relaxations and rounding. By studying the max-cut problem, we will learn about approximation algorithms. When we finally get to sparsest cut, we only need one more new thing (metric spaces) so it should be much more manageable.

2.4 Algorithms and running-time efficiency

This section is a very quick introduction to some relevant concepts in theoretic computer science.

Since our graphs are finite, we could solve the three problems above by considering all possible cuts. For each problem, the number of cuts we need to consider is at most the number of subsets of V , which is $2^{|V|}$. However, if V is large, then this is not very feasible. The time needed for this brute force approach grows exponentially in $|V|$. For that reason, we consider this algorithm to be inefficient.

Definition 2.3. An algorithm is said to be of *polynomial time* if its running time is upper bounded by a polynomial expression in the size of the input for the algorithm. This means that there exists constants $a, b \geq 0$ such that the following is true: For any input of size N , the number of basic operations taken by the algorithm is at most aN^b .

For this class, we will use the following informal terminology:

- An algorithm is “efficient” if it is of polynomial time. Otherwise, the algorithm is “inefficient.”
- A problem is “easy” if it has an efficient algorithm. Otherwise, the problem is “hard.”

Remark 2.4. Note that for us, the distinction between efficient and inefficient is polynomial vs non-polynomial. We don’t make a distinction between different degrees of polynomials. The functions N , N^2 and $10^{1000}N^{500}$ are all polynomials, so any of these bounds would be good enough to show an algorithm is polynomial-time.

If you have studied algorithms before, you have probably learned about sorting algorithms such as bubble-sort and merge-sort. Given a list of N numbers, bubble-sort has running time

$O(N^2)$, while merge-sort has running time $O(N \log N)$. Since both of these algorithms are of polynomial time, for the purposes of this class, we consider both of them to be efficient, even though $N \log N$ grows much more slowly than N^2 .

Theorem 2.5. *The min-cut problem can be solved by a polynomial time algorithm.*

Proof. We will prove this in the next few sections. Stay tuned! □

Theorem 2.6. *The max-cut and sparsest cut problems cannot be solved by polynomial time algorithms, unless $P = NP$.*

Proof. Proof omitted. □

Fun fact 2.7. Using computer science terminology, Theorem 2.5 says that the min-cut problem is in P , while Theorem 2.6 says that the max-cut and sparsest cut problems are NP-hard. In fact, max-cut and sparsest cut are NP-complete.

Technically, P and NP-hard are classes of decision problems (questions that ask for a “yes/no” answer), while the three cut problems are optimization problems. Each of these three problems has a corresponding decision version. For example, the decision version of min-cut asks the following. Given a graph $G = (V, E)$ and $s, t \in V$ and a number λ , is there a cut $S \subset V$ such that $s \in S$, $t \in S^c$, and $|E(S, S^c)| \leq \lambda$? When we say “min-cut is in P ” we really mean that the *decision version* of min-cut is in P .

Theorem 2.6 might seem like bad news for max-cut and sparsest cut, but this is not the end of the story. Later, we will talk about efficient approximation algorithms for these two problems.

2.5 Linear programming

https://en.wikipedia.org/wiki/Linear_programming

For this class, we only need one example of a problem that has a polynomial time algorithm, and that problem is the linear programming problem.

Suppose x_1, \dots, x_n are variables. A linear equation in x_1, \dots, x_n is something of the form

$$a_1x_1 + \dots + a_nx_n = b. \tag{2.4}$$

A linear inequality in x_1, \dots, x_n is something of the form

$$a_1x_1 + \dots + a_nx_n \leq b. \tag{2.5}$$

We do not allow strict inequalities.

A *linear programming problem* (a.k.a. *linear program* or *LP* for short) is to maximize or minimize an expression of the form $c_1x_1 + \dots + c_nx_n$ subject to constraints of the forms

(2.4) and (2.5). The expression that we are maximizing or minimizing is called the *objective function*.

Using vector notation can help with writing down the general form of a linear program. Recall that for two vectors $\vec{c} = (c_1, \dots, c_n)$ and $\vec{x} = (x_1, \dots, x_n)$, their dot product is $\vec{c} \cdot \vec{x} = c_1x_1 + \dots + c_nx_n$. The general form of a linear program is.

$$\begin{aligned}
 &\text{given: } \vec{c} = (c_1, \dots, c_n) \in \mathbb{R}^n \\
 &\quad \vec{a}_i = (a_{i,1}, \dots, a_{i,n}) \in \mathbb{R}^n \text{ for } i = 1, \dots, \ell \\
 &\quad b_i \in \mathbb{R} \text{ for } i = 1, \dots, \ell \\
 &\text{unknown: } \vec{x} = (x_1, \dots, x_n) \\
 &\text{maximize: } \vec{c} \cdot \vec{x} \\
 &\text{subject to: } \vec{a}_i \cdot \vec{x} \leq b_i \text{ for } i = 1, \dots, k \\
 &\quad \vec{a}_i \cdot \vec{x} = b_i \text{ for } i = k + 1, \dots, \ell
 \end{aligned} \tag{2.6}$$

We won't actually need to write things down this way in this class, but it is useful for some of the exercises.

Remark 2.8. Often, the term “linear programming problem” is defined with the additional requirement that all the variables are nonnegative. This can be a convenient condition to have (e.g., for some algorithms to work), but it's not necessary for the following reason: Every linear program of the form (2.6) can be converted into one with nonnegative variables. See Exercise 3.1 and Exercise 3.2 for more information.

Example 2.9. The following is a linear programming problem:

$$\begin{aligned}
 &\text{unknowns: } x_1, x_2 \\
 &\text{maximize: } 2x_1 + x_2 \\
 &\text{subject to: } x_1 \geq -2 \\
 &\quad x_2 \leq 1 \\
 &\quad x_1 - x_2 \leq 0
 \end{aligned} \tag{2.7}$$

The *feasible region* (i.e., the region defined by the constraints) is a triangle with vertices $(-2, -2)$, $(-2, 1)$, $(1, 1)$.

We are trying to maximize $2x_1 + x_2$. For different values of t , the lines $2x_1 + x_2 = t$ are all parallel to each other. (They all have slope -2 .) As we increase t , the lines move to the right. Maximizing $2x_1 + x_2$ is the same as finding how far we can move the line to the right and have it still intersect the triangle. The rightmost line that does this contains the point $(1, 1)$ and corresponds to $t = 3$. So the solution to the linear program is 3, and it is attained at $(x_1, x_2) = (1, 1)$.

The example Example 2.9 shows us several properties of all linear programs, which we state in the following lemmas. First recall the definition of convex.

Definition 2.10. A set $S \subset \mathbb{R}^n$ is *convex* if for all $a, b \in S$, the line segment joining a and b is also contained in S .

Lemma 2.11. *The feasible region in any linear programming problem will be a closed convex polytope in \mathbb{R}^n .*

Proof. The set of $x \in \mathbb{R}^n$ satisfying (2.4) is a $(n - 1)$ -dimensional plane in \mathbb{R}^n , which is a closed convex set. The set of $x \in \mathbb{R}^n$ satisfying (2.5) is a half-space in \mathbb{R}^n , which is also a closed convex set. The feasible region is the intersection of such sets. The intersection of convex sets is convex, and the intersection of closed sets is closed. \square

Lemma 2.12. *For any linear program, the maximum of the objective function, if it exists, must be attained at one of the vertices of the polytope. (The same is true for the minimum.)*

Proof. We omit the proof. The idea is the same as in Example 2.9. \square

Fun fact 2.13. The word “programming” in “linear programming” does not mean “writing a computer program.” Nowadays, it means “optimization.” Other examples of this usage are “dynamic programming,” “integer programming,” “semidefinite programming.”

The origin of this usage was from Dantzig in 1940s. He used “programming” to mean “planning” (e.g., like in “program of events”). The first uses of linear programming were to optimize planning problems in the US Air Force.

2.5.1 Algorithms

Solving a system of linear equations is easy. You can use row-reduction to get a polynomial-time algorithm. Solving a linear program is not as easy, but it can still be done in polynomial-time, as stated in the following theorem.

Theorem 2.14. *Linear programming can be solved with a polynomial time algorithm.*

Proof. Proof omitted. See Fun fact 2.15 for some information. \square

We will rely on Theorem 2.14 a lot in this class. It is fine if you do not know any algorithm to solve linear programming; we will not need that in this class.

Fun fact 2.15. The first example of a polynomial-time algorithm for linear programming is Khachiyan’s ellipsoid algorithm, introduced in 1974. It is not a practical algorithm. Another polynomial time algorithm, Karmarkar’s projective algorithm was introduced in 1984. This one was actually practical.

See https://en.wikipedia.org/wiki/Linear_programming#Algorithms for more information.

There is also something called the *simplex method* which in practice is very fast, but the worst-case performance is not polynomial-time.

Fun fact 2.16. There are many numerical libraries for solving linear programs. See for example <https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html>

2.6 A reformulation of min-cut

Our goal now is to show how to use linear programming to solve the min-cut problem. This will be accomplished in the next few sections.

Let $G = (V, E)$ be a graph. For $S \subset V$, let $\mathbf{1}_S : V \rightarrow \{0, 1\}$ be the *indicator function* (a.k.a. *characteristic function*) of S . That is,

$$\mathbf{1}_S(u) = \begin{cases} 1, & \text{if } u \in S \\ 0, & \text{if } u \notin S \end{cases} \quad (2.8)$$

Observe that if $u, v \in V$, then

$$|\mathbf{1}_S(u) - \mathbf{1}_S(v)| = \begin{cases} 1, & \text{if one of } u, v \text{ is in } S \text{ and the other is in } S^c \\ 0, & \text{if } u \text{ and } v \text{ are both in } S \text{ or both in } S^c \end{cases} \quad (2.9)$$

Therefore,

$$|E(S, S^c)| = \sum_{uv \in E} |\mathbf{1}_S(u) - \mathbf{1}_S(v)| \quad (2.10)$$

Thus, min-cut is the same as the following optimization problem:

$$\begin{aligned} &\text{unknown: } S \subset V \\ &\text{minimize: } \sum_{uv \in E} |\mathbf{1}_S(u) - \mathbf{1}_S(v)| \\ &\text{subject to: } s \in S, t \in S^c \end{aligned} \quad (\text{MC})$$

By changing notation, we get the following equivalent problem:

$$\begin{aligned} &\text{unknowns: } \{x_u\}_{u \in V} \\ &\text{minimize: } \sum_{uv \in E} |x_u - x_v| \\ &\text{subject to: } x_u \in \{0, 1\} \text{ for all } u \in V \\ &\quad \quad \quad x_s = 1, x_t = 0 \end{aligned} \quad (\text{MC}')$$

To see that (MC) and (MC') are really the same problem, note that choosing $x_u \in \{0, 1\}$ for each $u \in V$ is like choosing a subset $S \subset V$. The correspondence is given by $S = \{u \in V : x_u = 1\}$.

However, (MC') is not a linear programming problem. There are two issues

1. The constraint $x_u \in \{0, 1\}$ is not a linear constraint.
2. The objective function $\sum_{uv \in E} |x_u - x_v|$ is not linear.

We'll take care of these issues next.

2.7 The LP relaxation of min-cut – part 1

https://en.wikipedia.org/wiki/Max-flow_min-cut_theorem#Linear_program_formulation

Consider the following problem:

$$\begin{aligned}
 &\text{unknown: } f : V \rightarrow [0, 1] \\
 &\text{minimize: } \sum_{uv \in E} |f(u) - f(v)| \\
 &\text{subject to: } f(s) = 1, f(t) = 0
 \end{aligned}
 \tag{MC-LP}$$

or changing notation,

$$\begin{aligned}
 &\text{unknowns: } \{x_u\}_{u \in V} \\
 &\text{minimize: } \sum_{uv \in E} |x_u - x_v| \\
 &\text{subject to: } x_u \in [0, 1] \text{ for all } u \in V \\
 &\quad x_s = 1, x_t = 0
 \end{aligned}
 \tag{MC-LP'}$$

The difference between (MC-LP') and (MC') is that we replaced $\{0, 1\}$ with its convex hull $x_u \in [0, 1]$. Note that $x_u \in [0, 1]$ is the same as saying $0 \leq x_u \leq 1$, so the constraints in (MC-LP') are all linear. Making this kind of change is called a *convex relaxation*. The set $\{0, 1\}$ is not convex, so we replaced it with its convex hull $[0, 1]$.

However, in (MC-LP'), the objective function $\sum_{uv \in E} |x_u - x_v|$ is not linear. There's a clever trick to fix this, which we will discuss tomorrow. (See also Exercise 3.4 and Exercise 3.5.)

3 Day 1 exercises

Several of these exercises are about equivalent linear programs.

Definition 3.1. Two linear programs A and B are *equivalent* if:

1. The two linear programs A and B have the same optimal value.
2. If you know a way to set the unknowns in A to give the optimal value for A , then you can use this information to set the variables in B to give the optimal value for B , and vice versa.

Exercise 3.1. (🐜) Consider the following linear program.

$$\begin{aligned}
 &\text{unknowns: } p_1, n_1, p_2, n_2 \\
 &\text{maximize: } 2(p_1 - n_1) + (p_2 - n_2) \\
 &\text{subject to: } p_1 - n_1 \geq -2 \\
 &\qquad\qquad p_2 - n_2 \leq 1 \\
 &\qquad\qquad (p_1 - n_1) - (p_2 - n_2) \leq 0 \\
 &\qquad\qquad p_1, n_1, p_2, n_2 \geq 0
 \end{aligned} \tag{3.1}$$

Show that (2.7) (from Example 2.9) and (3.1) have the same maximum value. Can you give an argument that does not actually solve (3.1)?

See Exercise 3.2 for a generalization of this.

Exercise 3.2. (🐜🐜) Show that every linear program (as described in (2.6)) is equivalent (see Definition 3.1) to a linear program with nonnegative variables, i.e., one of the following form:

$$\begin{aligned}
 &\text{given: } \vec{c} = (c_1, \dots, c_n) \in \mathbb{R}^n \\
 &\qquad\qquad \vec{a}_i = (a_{i,1}, \dots, a_{i,n}) \in \mathbb{R}^n \text{ for } i = 1, \dots, \ell \\
 &\qquad\qquad b_i \in \mathbb{R} \text{ for } i = 1, \dots, \ell \\
 &\text{unknown: } \vec{x} = (x_1, \dots, x_n) \\
 &\text{maximize: } \vec{c} \cdot \vec{x} \\
 &\text{subject to: } \vec{a}_i \cdot \vec{x} \leq b_i \text{ for } i = 1, \dots, k \\
 &\qquad\qquad \vec{a}_i \cdot \vec{x} = b_i \text{ for } i = k + 1, \dots, \ell \\
 &\qquad\qquad \vec{x} \geq \vec{0}
 \end{aligned} \tag{3.2}$$

(Here $\vec{x} \geq \vec{0}$ means all the components of \vec{x} are nonnegative.)

Exercise 3.3. (🐜🐜) Show that every linear program (as described in (2.6)) is equivalent to one in *standard form*. In standard form, all the variables are nonnegative, and there are


no linear equality constraints:


$$\begin{aligned} \text{given: } & \vec{c} = (c_1, \dots, c_n) \in \mathbb{R}^n \\ & \vec{a}_i = (a_{i,1}, \dots, a_{i,n}) \in \mathbb{R}^n \text{ for } i = 1, \dots, k \\ & b_i \in \mathbb{R} \text{ for } i = 1, \dots, k \\ \text{unknown: } & \vec{x} = (x_1, \dots, x_n) \\ \text{maximize: } & \vec{c} \cdot \vec{x} \\ \text{subject to: } & \vec{a}_i \cdot \vec{x} \leq b_i \text{ for } i = 1, \dots, k \\ & \vec{x} \geq \vec{0} \end{aligned} \tag{3.3}$$

Exercise 3.4. () Consider the following minimization problem, which is not a linear program:

$$\begin{aligned} \text{unknowns: } & x_1, x_2 \\ \text{minimize: } & |2x_1 + x_2| \\ \text{subject to: } & x_1 \geq -2 \\ & x_2 \leq 1 \\ & x_1 - x_2 \leq 0 \end{aligned} \tag{3.4}$$

Can you find a linear program which is equivalent to it?

Exercise 3.5. () Same question as (3.4), but change the objective function to $2|x_1| + |x_2|$.

Exercise 3.6. () This question is about the max-flow problem. Let $G = (V, E)$ be a (finite, simple, undirected) graph.

For each $e \in E$, let $c_e > 0$ be a number, called the *capacity* of e . Also, let $s, t \in V$ be two distinct vertices.

In the max-flow problem, we want to “flow” as much quantity from s to t along the edges, with the following constraints:

1. The amount flowing through an edge e cannot exceed its capacity c_e .
2. Stuff cannot accumulate in any of the vertices other than s and t : If $u \in V \setminus \{s, t\}$, then the total amount flowing into u has to equal the two amount flowing out of u . (Thus, by conservation of “mass,” the amount flowing out of s has to equal the amount flowing into t .)

See Figure 2 for an example. Can you formulate this as a linear programming problem?

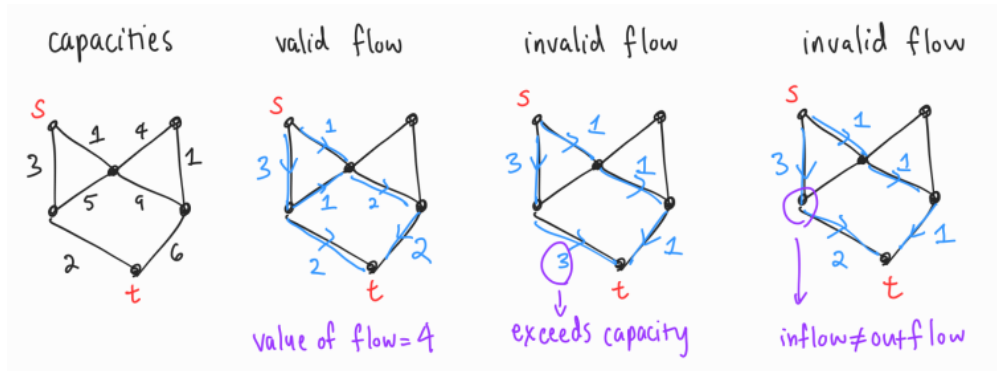


Figure 2: Example of a max-flow problem

4 Day 2

4.1 The LP relaxation of min-cut – part 2

Recall from yesterday that we are interested in (MC-LP'), but the objective function for that problem is not linear. There is a trick to fix this, which is illustrated in Example 4.1.

Example 4.1. Consider

$$\begin{aligned}
 &\text{unknowns: } x_1, x_2 \\
 &\text{minimize: } |x_1| + |x_2| \\
 &\text{subject to: [linear constraints on } x_1, x_2\text{]}
 \end{aligned} \tag{4.1}$$

This is equivalent to

$$\begin{aligned}
 &\text{unknowns: } x_1, x_2, y_1, y_2 \\
 &\text{minimize: } y_1 + y_2 \\
 &\text{subject to: } y_1 \geq |x_1| \\
 &\quad y_2 \geq |x_2| \\
 &\quad \text{[linear constraints on } x_1, x_2\text{]}
 \end{aligned} \tag{4.2}$$

which in turn is equivalent to

$$\begin{aligned}
 &\text{unknowns: } x_1, x_2, y_1, y_2 \\
 &\text{minimize: } y_1 + y_2 \\
 &\text{subject to: } y_1 \geq x_1 \\
 &\quad y_1 \geq -x_1 \\
 &\quad y_2 \geq x_2 \\
 &\quad y_2 \geq -x_2 \\
 &\quad \text{[linear constraints on } x_1, x_2\text{]}
 \end{aligned} \tag{4.3}$$

Thus, we have successfully converted (4.1) into a linear program.

Remark 4.2. If we change “minimize” to “maximize” in (4.1), the approach in Example 4.1 will not produce a linear program. You are asked about this in Exercise 5.1.

Following Example 4.1, the minimization problem (MC-LP') is equivalent to the following:

$$\begin{aligned}
 &\text{unknowns: } \{x_u\}_{u \in V} \cup \{x_{uv}\}_{uv \in E} \\
 &\text{minimize: } \sum_{uv \in E} x_{uv} \\
 &\text{subject to: } x_u \in [0, 1] \text{ for all } u \in V \\
 &\qquad x_s = 1, x_t = 0 \\
 &\qquad x_{uv} \geq x_u - x_v \text{ for all } u, v \in V \\
 &\qquad x_{uv} \geq x_v - x_u \text{ for all } u, v \in V
 \end{aligned} \tag{MC-LP''}$$

(MC-LP'') is actually a linear program! We refer to (MC-LP'') (or (MC-LP)) as the *linear programming relaxation* of min-cut.

Fun fact 4.3. The same trick used to go from (MC-LP') to (MC-LP'') can also be applied to show that (MC') is equivalent to the following

$$\begin{aligned}
 &\text{unknowns: } \{x_u\}_{u \in V} \cup \{x_{uv}\}_{uv \in E} \\
 &\text{minimize: } \sum_{uv \in E} x_{uv} \\
 &\text{subject to: } x_u \in [0, 1] \text{ for all } u \in V \\
 &\qquad x_s = 1, x_t = 0 \\
 &\qquad x_{uv} \geq x_u - x_v \text{ for all } u, v \in V \\
 &\qquad x_{uv} \geq x_v - x_u \text{ for all } u, v \in V \\
 &\qquad \{x_u\}_{u \in V} \cup \{x_{uv}\}_{uv \in E} \subset \mathbb{Z}
 \end{aligned} \tag{MC''}$$

Thus, (MC') would be a linear programming problem if it were not for the constraint that the unknowns are integers. This is an example of a *integer linear programming problem*. Integer linear programming is NP-hard (and in fact NP-complete).

4.2 From the LP relaxation back to min-cut, the easy way

We can use an efficient algorithm to find a function $f^* : V \rightarrow [0, 1]$ that minimizes (MC-LP). (Recall Theorem 2.14.) But (MC-LP) is no longer the min-cut problem. What do we do with this function f^* ?

Recall from Lemma 2.12 that the minimum of (MC-LP'') is attained at a vertex feasible region. Furthermore, there are efficient algorithms for LP that always output a vertex. Observe that for any vertex of the feasible region of (MC-LP''), we have $x_u \in \{0, 1\}$. So from f , we do in fact get a cut by setting $S = \{u \in V : f^*(u) = 1\}$. Furthermore, this cut minimizes (MC), so we are done!

Fun fact 4.4. For any linear program A , we can define a *dual* linear program B . There is something called the *strong duality theorem* which relates the solutions to A and B . The dual of min-cut is max-flow. (See Exercise 3.6 for the max-flow problem.) The max-flow min-cut theorem is a special case of the strong duality theorem. See Exercise 5.7 for more details.

4.3 Min-cut and its LP relaxation have the same minimum value

Let us forget about Section 4.2. Suppose we have found a function $f^* : V \rightarrow [0, 1]$ that minimizes (MC-LP). We will now use a more complicated approach to find a cut S that minimizes (MC). Why? This approach will help us a lot when we study sparsest cut. The method in Section 4.2 will not work for sparsest cut.

Recall the two optimization problems defined in (MC) and (MC-LP). Let their optimal values be denoted $\text{OPT}(\text{MC})$ and $\text{OPT}(\text{MC-LP})$ respectively. In this section we prove the following without using Section 4.2.

Theorem 4.5. $\text{OPT}(\text{MC-LP}) = \text{OPT}(\text{MC})$

A preliminary observation: For every $S \subset V$, the indicator function $\mathbb{1}_S$ is a function $V \rightarrow [0, 1]$, so $\text{OPT}(\text{MC-LP}) \leq \text{OPT}(\text{MC})$. Thus we only need to show

$$\text{OPT}(\text{MC-LP}) \geq \text{OPT}(\text{MC}). \quad (4.4)$$

The key observation is the following.

Lemma 4.6. *Let V be a finite set and let $f : V \rightarrow [0, 1]$ be any function. There are constants $\{c_S\}_{S \subset V} \subset [0, 1]$ such that $\sum_{S \subset V} c_S = 1$ and*

$$|f(u) - f(v)| = \sum_{S \subset V} c_S |\mathbb{1}_S(u) - \mathbb{1}_S(v)| \text{ for all } u, v \in V. \quad (4.5)$$

Remark 4.7. What is a good interpretation of Lemma 4.6? We'll come back to that when we discuss metric spaces. See Section 8.2.

For now, let's observe that this lemma does in fact seem relevant to proving Theorem 4.5. The expression $|f(u) - f(v)|$ shows up in the objective function of (MC-LP), and the expression $|\mathbb{1}_S(u) - \mathbb{1}_S(v)|$ shows up in the objective function of (MC). If we think of the expressions

$|f(u) - f(v)|$ and $|\mathbb{1}_S(u) - \mathbb{1}_S(v)|$ as functions of $(u, v) \in V \times V$, then Lemma 4.6 says the function $|f(u) - f(v)|$ is a weighted average (a.k.a. *convex combination*) of the functions $\{|\mathbb{1}_S(u) - \mathbb{1}_S(v)|\}_{S \subset V}$.

For fun, we give two proofs. They are really the same proof, just presented differently. The second proof uses integration to simplify some of the arguments.

First proof of Lemma 4.6. Fix a function $f : V \rightarrow [0, 1]$. For $x \in [0, 1]$, define

$$T(x) = f^{-1}([0, x]) = \{w \in V : f(w) < x\}. \quad (4.6)$$

A subset of V the form $T(x)$ is called a *threshold cut* with threshold x . (T stands for threshold.)

For any $u, v \in V$ and $x \in [0, 1]$, we have

$$|\mathbb{1}_{T(x)}(u) - \mathbb{1}_{T(x)}(v)| = \begin{cases} 1, & \text{if } \min\{f(u), f(v)\} < x \leq \max\{f(u), f(v)\} \\ 0, & \text{otherwise} \end{cases} \quad (4.7)$$

(This follows from (2.9) and the definition of $T(x)$.)

Let $0 = y_1 < y_2 < \dots < y_\ell = 1$ denote the elements of $\{f(w) : w \in V\} \cup \{0, 1\}$ in increasing order. Observe that if $x \in (y_{i-1}, y_i]$, then $T(x) = T(y_i)$.

For the rest of the proof, keep u and v fixed. Let $m, n \in \{1, \dots, \ell\}$ be such that $y_m = \min\{f(u), f(v)\}$ and $y_n = \max\{f(u), f(v)\}$. From (4.7),

$$|\mathbb{1}_{T(y_i)}(u) - \mathbb{1}_{T(y_i)}(v)| = \begin{cases} 1, & \text{if } m < i \leq n \\ 0, & \text{otherwise} \end{cases} \quad (4.8)$$

Thus, if we let

$$c_S = \begin{cases} y_i - y_{i-1}, & \text{if } S = T(y_i) \text{ for some } i \in \{2, \dots, \ell\} \\ 0, & \text{otherwise} \end{cases} \quad (4.9)$$

then we have

$$\sum_{S \subset V} c_S = \sum_{i=2}^{\ell} c_{T(y_i)} = \sum_{i=2}^{\ell} (y_{i+1} - y_i) = y_\ell - y_1 = 1 \quad (4.10)$$

and

$$\sum_{S \subset V} c_S |\mathbb{1}_S(u) - \mathbb{1}_S(v)| = \sum_{i=2}^{\ell} (y_{i+1} - y_i) |\mathbb{1}_{T(y_i)}(u) - \mathbb{1}_{T(y_i)}(v)| \quad (4.11)$$

$$= \sum_{i=m+1}^n (y_{i+1} - y_i) \quad (4.12)$$

$$= y_n - y_m \quad (4.13)$$

$$= |f(u) - f(v)| \quad (4.14)$$

which completes the proof. \square

Second proof of Lemma 4.6. The proof starts in the same way as the first proof by defining $T(x)$ and noting (4.7). From (4.7), we get

$$\int_0^1 |\mathbb{1}_{T(x)}(u) - \mathbb{1}_{T(x)}(v)| dx = \int_{\min\{f(u), f(v)\}}^{\max\{f(u), f(v)\}} 1 dx = |f(u) - f(v)| \quad (4.15)$$

For each $S \subset V$, consider

$$T^{-1}(S) = \{x \in [0, 1] : T(x) = S\} \quad (4.16)$$

Since V is finite, $T^{-1}(S)$ is a (possibly empty) interval. Note that if S is not a threshold cut, then $T^{-1}(S) = \emptyset$.

Let c_S be the length of the interval $T^{-1}(S)$. Since the sets $\{T^{-1}(S)\}_{S \subset V}$ are disjoint and $[0, 1] = \bigcup_{S \subset V} T^{-1}(S)$, we have $\sum_{S \subset V} c_S = 1$ and

$$\int_0^1 |\mathbb{1}_{T(x)}(u) - \mathbb{1}_{T(x)}(v)| dx = \sum_{S \subset V} \int_{T^{-1}(S)} |\mathbb{1}_{T(x)}(u) - \mathbb{1}_{T(x)}(v)| dx \quad (4.17)$$

$$= \sum_{S \subset V} \int_{T^{-1}(S)} |\mathbb{1}_S(u) - \mathbb{1}_S(v)| dx \quad (4.18)$$

$$= \sum_{S \subset V} c_S |\mathbb{1}_S(u) - \mathbb{1}_S(v)|. \quad (4.19)$$

Combining this with (4.15) completes the proof. \square

Remark 4.8. Exercise 5.2 is a generalization of Lemma 4.6 that we will use when studying sparsest cut. It is also a good way to check your understanding of the proof of Lemma 4.6.

Proof of Theorem 4.5. Let $f^* : V \rightarrow [0, 1]$ achieve the minimum in (MC-LP). Apply Lemma 4.6 to get constants $\{c_S\}_{S \subset V}$ such that (4.5) is satisfied for f^* . Since $f^*(s) = 1$ and $f^*(t) = 0$, by inspecting the proof of Lemma 4.6, we see that if $c_S > 0$, then S is a valid cut for (MC). Then

$$\text{OPT}(\text{MC}) \geq \text{OPT}(\text{MC-LP}) \quad (4.20)$$

$$= \sum_{uv \in E} |f^*(u) - f^*(v)| \quad (4.21)$$

$$= \sum_{uv \in E} \sum_{S \subset V} c_S |\mathbb{1}_S(u) - \mathbb{1}_S(v)| \quad (4.22)$$

$$= \sum_{S \subset V} c_S \sum_{uv \in E} |\mathbb{1}_S(u) - \mathbb{1}_S(v)| \quad (4.23)$$

$$\geq \sum_{S \subset V} c_S \text{OPT}(\text{MC}) \quad (4.24)$$

$$= \text{OPT}(\text{MC}) \quad (4.25)$$

which completes the proof. \square

We've shown that $\text{OPT}(\text{MC-LP}) = \text{OPT}(\text{MC})$. We are not done yet. From the minimizer f^* of (MC-LP), how do we get a subset $S \subset V$ that solves the min-cut problem? Again, we want a way that avoids Section 4.2, i.e., we should not assume that f^* takes values in $\{0, 1\}$. We will discuss this tomorrow. See also Exercise 5.5.

5 Day 2 exercises

The following exercises are especially recommended because they will be useful for upcoming classes: Exercise 5.2, Exercise 5.5

Exercise 5.1. (🐞) What goes wrong when you try to copy the method in Example 4.1 to turn the following into a linear program?

$$\begin{aligned} &\text{unknowns: } x_1, x_2 \\ &\text{MAXIMIZE: } |x_1| + |x_2| \\ &\text{subject to: [linear constraints on } x_1, x_2\text{]} \end{aligned} \tag{5.1}$$

Exercise 5.2. (🐞) Let V be a finite set and let $f : V \rightarrow \mathbb{R}$ be any function. Show that there are constants $\{c_S\}_{S \subset V} \subset [0, \infty)$ such that

$$|f(u) - f(v)| = \sum_{S \subset V} c_S |\mathbf{1}_S(u) - \mathbf{1}_S(v)| \text{ for all } u, v \in V. \tag{5.2}$$

Note: This “🐞” of this exercise is only accurate if you understand the proof of Lemma 4.6.

Exercise 5.3. (🐞🐞🐞) Here is an extension of Exercise 5.2. Let ℓ_1 be the set of infinite sequences of real numbers (x_1, x_2, \dots) such that $\sum_{i=1}^{\infty} |x_i|$ converges.

Let V be a finite set and let $f : V \rightarrow \ell_1$ be any function. (So in particular, $f(u) = (f_1(u), f_2(u), \dots)$ for some functions $f_i : V \rightarrow \mathbb{R}$.)

Show that there are constants $\{c_S\}_{S \subset V} \subset [0, \infty)$ such that

$$\sum_{i=1}^{\infty} |f_i(u) - f_i(v)| = \sum_{S \subset V} c_S |\mathbf{1}_S(u) - \mathbf{1}_S(v)| \text{ for all } u, v \in V. \tag{5.3}$$


Exercise 5.4. (🐞🐞🐞) Here is another extension of Exercise 5.2. Let $\mathcal{C}([a, b])$ be the set of continuous functions $[a, b] \rightarrow \mathbb{R}$.

Let V be a finite set and let $f : V \rightarrow \mathcal{C}([a, b])$ be any function. (Be careful! For each $u \in V$, $f(u)$ is a function $[a, b] \rightarrow \mathbb{R}$. This means that for each $u \in V$ and $t \in [a, b]$, we have $f(u)(t) \in \mathbb{R}$. Note that f itself is not a function $[a, b] \rightarrow \mathbb{R}$.)


Show that there are constants $\{c_S\}_{S \subset V} \subset [0, \infty)$ such that

$$\int_a^b |f(u)(t) - f(v)(t)| dt = \sum_{S \subset V} c_S |\mathbf{1}_S(u) - \mathbf{1}_S(v)| \text{ for all } u, v \in V. \tag{5.4}$$

Fun fact 5.1. If we use measure theory, we can state and prove a general result that contains Exercise 5.2, Exercise 5.3, and Exercise 5.4 as special cases.

Exercise 5.5. () Suppose f^* is a minimizer of (MC-LP). Using this, find (with proof) a subset $S \subset V$ that solves the min-cut problem. Do not use any of the facts from Section 4.2. In particular, do not assume that for all $u \in V$, $f^*(u) \in \{0, 1\}$.

Hint: Examine the proof of Theorem 4.5.


Exercise 5.6. () Here is a version of the min-cut problem for graphs with weights. Let $G = (V, E)$ be a graph. (As before, it is finite, simple, and undirected.)

For each $e \in E$, let $c_e > 0$ be a number, called the *capacity* of e . Also, let $s, t \in V$ be two distinct vertices.

For $S \subset V$, we define the capacity of the cut S to be

$$c(S, S^c) = \sum_{e \in E(S, S^c)} c_e \quad (5.5)$$

The min-cut problem for weighted graphs says: Find a cut $S \subset V$ such that $s \in S$, $t \in S^c$, and such that $c(S, S^c)$ is minimized. Can you come up with an LP relaxation for this problem?

Exercise 5.7. () Look up the definition of a dual linear program, e.g., https://en.wikipedia.org/wiki/Dual_linear_program. Show that your linear programs from Exercise 3.6 and Exercise 5.6 are in fact dual to each other.

Now look up the statement of the strong duality theorem, e.g., https://en.wikipedia.org/wiki/Dual_linear_program#Strong_duality. Use this to prove the *max-flow min-cut theorem*: Given a graph $G = (V, E)$, a source $s \in V$, a sink $t \in V$, and capacities $c_e > 0$, the minimum cut (as defined in Exercise 5.6) and the maximum flow (as defined in Exercise 3.6) are equal to each other.

6 Day 3

6.1 Rounding the LP-relaxation of min-cut

We've shown that $\text{OPT}(\text{MC-LP}) = \text{OPT}(\text{MC})$. We are not done yet. From the minimizer f^* of (MC-LP), how do we get a subset $S \subset V$ that solves the min-cut problem? Again, we want a way that avoids Section 4.2, i.e., we should not assume that f^* takes values in $\{0, 1\}$.

To find such a cut, let us examine our proof of Theorem 4.5 in more detail. Observe that from the chain of inequalities (4.20) starts and ends with $\text{OPT}(\text{MC})$, which means everything

in the chain is equal. In particular,

$$\sum_{S \subset V} c_S \sum_{uv \in E} |\mathbb{1}_S(u) - \mathbb{1}_S(v)| = \sum_{S \subset V} c_S \text{OPT}(\text{MC}), \quad (6.1)$$

so

$$c_S \sum_{uv \in E} |\mathbb{1}_S(u) - \mathbb{1}_S(v)| = c_S \text{OPT}(\text{MC}) \text{ for all } S \subset V, \quad (6.2)$$

so

$$c_S > 0 \implies \text{OPT}(\text{MC}) = \sum_{uv \in E} |\mathbb{1}_S(u) - \mathbb{1}_S(v)| \quad (6.3)$$

In other words, take any S with $c_S > 0$. That cut will achieve the minimum. From the proof of Lemma 4.6, we see that $c_S > 0$ iff S is a threshold cut, i.e., a subset of the form $\{u \in V : f^*(u) < x\}$ for some $x \in (0, 1)$.

6.2 Summary of our min-cut algorithm

Problem: We want to find a cut $S \subset V$ such that $\text{OPT}(\text{MC})$ is attained.

1. Consider the the LP-relaxation of $\text{OPT}(\text{MC})$, which we called $\text{OPT}(\text{MC-LP})$.
2. Find a function $f^* : V \rightarrow \mathbb{R}$ that achieves the minimum for $\text{OPT}(\text{MC-LP})$. (This can be done in polynomial-time by using an efficient linear programming algorithm. See Theorem 2.14.)
3. Let $S = \{u \in V : f^*(u) < \frac{1}{2}\}$. This achieves the minimum cut. (This step is called *rounding* the LP-relaxation. We proved in Section 6.1 why this choice of S works. Also, note that the $\frac{1}{2}$ can be replaced by any number strictly between 0 and 1.)

Remark 6.1. As we already pointed out in Section 4.2, for min-cut, the approach in Section 4.3 and Section 6.1 is unnecessarily complicated.

6.3 Max-cut

Recall the max-cut problem:

$$\begin{aligned} &\text{unknown: } S \subset V \\ &\text{maximize: } \sum_{uv \in E} |\mathbb{1}_S(u) - \mathbb{1}_S(v)| \end{aligned} \quad (\text{MAXCUT})$$

There are two changes from min-cut (MC). (1) The word “minimize” has been changed to “maximize.” (2) There is no longer a source s or a target t .

Remark 6.2. We cannot apply the methods of Section 2.7 and Section 4.1 to turn (MAXCUT) into a linear program. See Remark 4.2. If you do find a way to convert (MAXCUT) into a linear program, then by Theorem 2.6, you also prove that $P = NP!$

6.3.1 A randomized $\frac{1}{2}$ -approximation algorithm for max-cut

First we prove the following:

Lemma 6.3. For any graph $G = (V, E)$,

$$\frac{1}{2}|E| \leq \text{OPT}(\text{MAXCUT}) \leq |E| \quad (6.4)$$

Proof. It is clear (from the definitions) that $\text{OPT}(\text{MAXCUT}) \leq |E|$, so we only need to show $\text{OPT}(\text{MAXCUT}) \geq \frac{1}{2}|E|$.

Let $S \subset V$ chosen at random, uniformly over all $2^{|V|}$ subsets of V . Another way to think about it is that for all $x \in V$, $\Pr(x \in S) = \frac{1}{2}$, and these are independent.

For each edge $uv \in E$, $\Pr(uv \in E(S, S^c)) = \frac{1}{2}$. Thus, by linearity of expectation,

$$\text{expected value of } |E(S, S^c)| = \sum_{uv \in E} \Pr(uv \in E(S, S^c)) = \frac{1}{2}|E|. \quad (6.5)$$

Thus, for any graph $G = (V, E)$ there is a cut S that cuts at least half of the edges. (This technique of proof is known as the *probabilistic method*.) \square

The proof of Lemma 6.3 shows that if we select a random subset $S \subset V$, then “on average” we will not be off from the maximum cut by more than a factor of 2:

$$\frac{1}{2}\text{OPT}(\text{MAXCUT}) \leq \text{expected value of } |E(S, S^c)| \leq \text{OPT}(\text{MAXCUT}) \quad (6.6)$$

The example we gave above is a *randomized $\frac{1}{2}$ -approximation algorithm* for max-cut. Here is the definition of an approximation algorithm.

Definition 6.4. Suppose we have a maximization problem and the optimal value is OPT . Let APPROX be the value given by our algorithm. If we know that

$$\rho \cdot \text{OPT} \leq \text{APPROX} \leq \text{OPT} \quad (6.7)$$

then our algorithm is called a ρ -approximation algorithm

Remark 6.5. Sometimes, the term ρ -approximation algorithm is defined with $1/\rho$ in place of ρ in (6.7). If we use this convention, then the algorithm above is a randomized 2-approximation algorithm instead.

Fun fact 6.6. It is possible to *derandomize* the algorithm above. There are several ways to do this, and they each produce a deterministic polynomial time $\frac{1}{2}$ -approximation algorithm for max-cut. See Exercise 7.1. See https://en.wikipedia.org/wiki/Randomized_algorithm#Derandomization for more on derandomization.

Fun fact 6.7. The *Goemans–Williamson algorithm* is a ρ -approximation algorithm for max-cut, with ρ equal to

$$\frac{2}{\pi} \min_{0 \leq \theta \leq \pi} \frac{\theta}{1 - \cos \theta} \approx 0.878. \quad (6.8)$$

The proof (which we omit) uses a semidefinite programming relaxation of max-cut, randomized rounding, and some Euclidean geometry. See Exercise 7.2 and Exercise 7.3.

If $P \neq NP$, then there does not exist a polynomial time ρ -approximation algorithm for any $\rho > \frac{16}{17} \approx 0.941$. If the *unique games conjecture* is true, then there does not exist a polynomial time ρ -approximation algorithm for any ρ greater than (6.8).

There are many more things we can say about max-cut. However, that is not our focus, so we'll finally move on to sparsest cut.

6.4 Metrics and metric spaces

We will briefly take a break from the graphs and algorithms to talk about metric spaces.

Let X be any set. A function $d : X \times X \rightarrow \mathbb{R}$ is called a *metric* on X if it satisfies the following:

- positivity: $d(x, y) \geq 0$ for all $x, y \in X$. Furthermore, $d(x, y) = 0$ iff $x = y$.
- symmetry: $d(x, y) = d(y, x)$ for all $x, y \in X$.
- triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$ for all $x, y, z \in X$.

We say that (X, d) is a *metric space*. Sometimes we just write X if the metric is understood. (This is like how a group is a set with a binary operation, but people often write “Let G be a group.”)

Example 6.8. Take $X = \mathbb{R}$ and $d(x, y) = |x - y|$. Then (X, d) is a metric space. We often refer to this metric space as \mathbb{R} . (The metric is understood.)

Example 6.9. Take $X = \mathbb{R}^n$ and $d(x, y) = \|x - y\|_2 = (\sum_{i=1}^n |x_i - y_i|^2)^{1/2}$. Then (X, d) is a metric space. The metric d is the usual distance from Euclidean geometry; it is called the Euclidean metric or ℓ_2 metric on \mathbb{R}^n . We denote the metric space ℓ_2^n .

Example 6.10. Let X be any set. Define $d(x, y)$ to be 1 if $x \neq y$ and 0 if $x = y$. This is called the *discrete metric* on X .

Example 6.11. Let $G = (V, E)$ be a connected graph. For $u, v \in V$, let $d_G(u, v)$ be the length of the shortest path from u to v . This is a *graph metric*.

6.5 The spaces ℓ_1^n and ℓ_1

We already saw the ℓ_2 metric on \mathbb{R}^n . Here is another metric on \mathbb{R}^n .

$$d(x, y) = \|x - y\|_1 = \sum_{i=1}^n |x_i - y_i| \quad (6.9)$$

It is not hard to see that this is a metric. You are asked to do this in Exercise 7.6. This is called the ℓ_1 metric (or ℓ^1 metric or taxicab metric or Manhattan metric). The metric space is denoted ℓ_1^n .

We can let $n \rightarrow \infty$ in the following way. For an infinite sequence $x = (x_1, x_2, \dots)$, define $\|x\|_1 = \sum_{i=1}^{\infty} |x_i|$. Let X be the set of infinite sequences x such that $\|x\|_1 < \infty$. We define a metric on X by $d(x, y) = \|x - y\|_1 = \sum_{i=1}^{\infty} |x_i - y_i|$. This metric space is called ℓ_1 .

Fun fact 6.12. Here are even more metrics on \mathbb{R}^n . Define

$$\|x\|_p = \begin{cases} (\sum_{i=1}^n |x_i|^p)^{1/p} & \text{if } p \in (0, \infty) \\ \max_{i \in \{1, \dots, n\}} |x_i| & \text{if } p = \infty \end{cases} \quad (6.10)$$

We define

$$d_p(x, y) = \|x - y\|_p. \quad (6.11)$$

For $p \in [1, \infty]$, the function d_p is a metric on \mathbb{R}^n . This metric space is called ℓ_p^n . We can also define ℓ_p for infinite sequences in the analogous way. These spaces ℓ_p^n and ℓ_p are examples of *Banach spaces*. A Banach space is a complete normed vector space. They are central in the study of functional analysis. We will not say more about Banach spaces in this class. Also, we only need $p = 1$ in this class.

For $p = 1$ and $p = \infty$, it is not hard to see that d_p this is a metric. You are asked to do this in Exercise 7.6. For $p \in (1, \infty)$, the situation is a little harder. See Exercise 7.7.

6.6 Pseudometrics

The function d is called a *pseudometric* if it satisfies symmetry, triangle inequality, and

- nonnegativity: $d(x, y) \geq 0$ for all $x, y \in X$. Furthermore, if $x = y$, then $d(x, y) = 0$.

Remark 6.13. Warning: Some people use semimetric instead of pseudometric. But semimetric also has a different meaning in other contexts. We'll just stick to pseudometric.

Fun fact 6.14. According to Wikipedia, in addition to pseudometrics, there are also things called extended metrics, quasimetrics, metametrics, semimetrics, premetrics, and pseudo-quasimetrics! See [https://en.wikipedia.org/wiki/Metric_\(mathematics\)#Generalized_metrics](https://en.wikipedia.org/wiki/Metric_(mathematics)#Generalized_metrics)

The difference between metrics and pseudometrics is that for pseudometrics, it is fine to have some $x \neq y$ satisfying $d(x, y) = 0$. For metrics, this is not allowed. Why do we care about pseudometrics? We will talk about that later. See also Exercise 7.10.

7 Day 3 exercises

The following exercises are especially recommended because they will be useful for upcoming classes: Exercise 7.9, Exercise 7.10.

Exercise 7.1. (🐜🐜) Can you find a deterministic $\frac{1}{2}$ -approximation algorithm for max-cut that runs in polynomial time? (Hint: Consider a greedy algorithm.)

Exercise 7.2. (🐜) Let $G = (V, E)$ be a graph. Consider the following two optimization problems.

1. First problem:

$$\begin{aligned} \text{unknowns: } & \{x_u\}_{u \in V} \subset \mathbb{R} \\ \text{maximize: } & \sum_{uv \in E} \frac{1 - x_u x_v}{2} && \text{(MAXCUT')} \\ \text{subject to: } & x_u \in \{-1, 1\} \end{aligned}$$

2. Second problem: Let $n = |V|$.


$$\begin{aligned} \text{unknowns: } & \{\vec{x}_u\}_{u \in V} \subset \mathbb{R}^n \\ \text{maximize: } & \sum_{uv \in E} \frac{1 - \vec{x}_u \cdot \vec{x}_v}{2} && \text{(MAXCUT-SDP)} \\ \text{subject to: } & \|\vec{x}_u\|_2^2 = 1 \end{aligned}$$

Note that in (MAXCUT-SDP), we have a vector \vec{x}_u for each $u \in V$. Also $\vec{x}_u \cdot \vec{x}_v$ denotes the dot product, and $\|\vec{x}_u\|_2^2 = \vec{x}_u \cdot \vec{x}_u$.


Show that (MAXCUT') is equivalent to (MAXCUT) and that


$$\text{OPT}(\text{MAXCUT-SDP}) \geq \text{OPT}(\text{MAXCUT}') \tag{7.1}$$

Fun fact 7.1. (**MAXCUT'**) is an example of an *integer quadratic program*. Integer quadratic programs are hard. (**MAXCUT-SDP**) is an example of a *semidefinite program* (SDP). There are polynomial-time algorithms to solve SDPs.

Exercise 7.3. () Suppose we already have vectors $\{\vec{x}_u^*\}_{u \in V}$ that achieve the maximum in (**MAXCUT-SDP**). Using these vectors, give a randomized ρ -approximation algorithm for (**MAXCUT**), where ρ is (6.8). (Hint: You need a way to use $\{\vec{x}_u^*\}_{u \in V}$ to define a cut $S \subset V$. There will be some randomness involved in this choice.)


Exercise 7.4. () Let $G = (V, E)$ be a connected graph. When is the graph metric d_G the same as the discrete metric on V ?

Exercise 7.5. () Let $X = \mathbb{R}$. Is $d(x, y) = |x - y|^2$ a metric? Is $d(x, y) = |x - y|^{1/2}$ a metric?


Exercise 7.6. ()


1. Show that ℓ_1^n is a metric space.
2. Show that ℓ_∞^n is a metric space.

(In particular, you should check that the triangle inequality holds.)

Exercise 7.7. ()

1. Show that ℓ_2^n is a metric space. (Hint: Use the Cauchy–Schwarz inequality.)
2. For $p \in (1, \infty)$ show that ℓ_p^n is a metric space. (Hint: Use Hölder’s inequality.)

Exercise 7.8. () For $p \in (0, 1)$ and $n \geq 2$, show that ℓ_p^n is *not* a metric space.

Exercise 7.9. () Let (X_n, d) be a n -point space with the discrete metric. (In other words, $|X_n| = n$, and the metric d is defined as in Example 6.10.)

1. Can you find a map $f : X_4 \rightarrow \mathbb{R}$ such that $d(x, y) = |f(x) - f(y)|$ for all $x, y \in X_4$?
2. Can you find a map $f : X_4 \rightarrow \mathbb{R}^2$ such that $d(x, y) = \|f(x) - f(y)\|_2$ for all $x, y \in X_4$?
3. Can you find a map $f : X_4 \rightarrow \mathbb{R}^3$ such that $d(x, y) = \|f(x) - f(y)\|_2$ for all $x, y \in X_4$?
4. For which $m, n \in \mathbb{N}$ does there exist a map $f : X_m \rightarrow \mathbb{R}^n$ such that $d(x, y) = \|f(x) - f(y)\|_2$ for all $x, y \in X_m$?
5. Can you find a map $f : X_4 \rightarrow \mathbb{R}^2$ such that $d(x, y) = \|f(x) - f(y)\|_1$ for all $x, y \in X_4$?
6. Can you find a map $f : X_5 \rightarrow \mathbb{R}^2$ such that $d(x, y) = \|f(x) - f(y)\|_1$ for all $x, y \in X_5$?

7. Let $n \in \mathbb{N}$. What is the smallest number $D \geq 1$ such that there exists a map $f : X_n \rightarrow \mathbb{R}$ such that

$$|f(x) - f(y)| \leq d(x, y) \leq D \cdot |f(x) - f(y)| \text{ for all } x, y \in X_n? \quad (7.2)$$

Exercise 7.10. (🦋🦋) In the context of the problems we have studied so far, what are some advantages that pseudometrics have over metric spaces? (This is an open-ended question. I can think of several answers.)

Exercise 7.11. (🦋🦋) Let (X, d) be a pseudometric space.

1. Define a relation \sim on X by

$$x \sim y \iff d(x, y) = 0. \quad (7.3)$$

Show that this is an equivalence relation.

2. For $x \in X$, let $[x] \subset X$ denote the equivalence class containing x . Let \tilde{X} denote the set of equivalence classes. Define $\tilde{d} : \tilde{X} \times \tilde{X} \rightarrow \mathbb{R}$ by

$$\tilde{d}([x], [y]) = d(x, y). \quad (7.4)$$

Show that \tilde{d} is well-defined and that (\tilde{X}, \tilde{d}) is a metric space. This space is called the *metric space induced by the pseudometric space (X, d)* .

For the definitions of equivalence relation and equivalence class, see, e.g.,

- https://en.wikipedia.org/wiki/Equivalence_relation
- https://en.wikipedia.org/wiki/Equivalence_class

8 Day 4

8.1 Examples of pseudometrics

For a metric we need $d(x, y) > 0$ whenever $x \neq y$, but in linear programming we cannot have strict inequalities.

Here are some important examples of pseudometrics.

Example 8.1 (Cut pseudometric). If $S \subset X$, then $d(x, y) = |\mathbb{1}_S(x) - \mathbb{1}_S(y)|$ is a pseudometric. It is called a *cut pseudometric* on X .

Example 8.2 (Line pseudometric). Let X be any set and let $f : X \rightarrow \mathbb{R}$ be any function. Let $d(x, y) = |f(x) - f(y)|$. This is a pseudometric. It is called a *line pseudometric* on X because it is defined by mapping X into the real line \mathbb{R} .

Example 8.3 (ℓ_1 pseudometric). Let X be any set and let $f : X \rightarrow \ell_1$ be any function. Let $d(x, y) = \|f(x) - f(y)\|_1$. This is a pseudometric. It is called a ℓ_1 *pseudometric* on X because it is defined by mapping X into the space ℓ_1 .

8.2 Cut-cone representation of ℓ_1 pseudometrics

Cut pseudometrics and line pseudometrics already showed up in the min-cut problem; see the objective functions of (MC) and (MC-LP). Lemma 4.6 can be interpreted as saying: Suppose X is a finite set and d is a line pseudometric on X . Suppose further that the diameter of (X, d) is at most 1. (The *diameter* of a pseudometric space (X, d) is $\sup_{x, y \in X} d(x, y)$.) Then d can be written as a convex combination of cut pseudometrics on X .

Here is a slight generalization of Lemma 4.6. We changed the function from $V \rightarrow [0, 1]$ to $V \rightarrow \mathbb{R}$. As a result, we no longer have $\sum_{S \subset V} c_S = 1$. However, the rest (including the proof) remains unchanged.

Lemma 8.4 (Cut-cone representation of line pseudometrics, Exercise 5.2). *Let V be a finite set and let $f : V \rightarrow \mathbb{R}$ be any function. Then there are constants $\{c_S\}_{S \subset V} \subset [0, \infty)$ such that*

$$|f(u) - f(v)| = \sum_{S \subset V} c_S |\mathbf{1}_S(u) - \mathbf{1}_S(v)| \text{ for all } u, v \in V \quad (8.1)$$

Proof. This was Exercise 5.2. □

Instead of a *convex combination* (or weighted average), we now have a *conical combination*. See Example 8.5 for the reason for this terminology.

Example 8.5. Let $\vec{u} = (1, 0) \in \mathbb{R}^2$ and $\vec{v} = (0, 1) \in \mathbb{R}^2$.

1. A *linear combination* of \vec{u} and \vec{v} is a vector of the form $a\vec{u} + b\vec{v}$ for some $a, b \in \mathbb{R}$. The set of all linear combinations of \vec{u} and \vec{v} is the entire plane \mathbb{R}^2 .
2. A *convex combination* of \vec{u} and \vec{v} is a vector of the form $a\vec{u} + b\vec{v}$ for some $a, b \in [0, 1]$ with $a + b = 1$. The set of all convex combinations of \vec{u} and \vec{v} is the line segment joining \vec{u} and \vec{v} .
3. A *conical combination* of \vec{u} and \vec{v} is a vector of the form $a\vec{u} + b\vec{v}$ for some $a, b \geq 0$. The set of all conical combinations of \vec{u} and \vec{v} is the closed first quadrant $\{(x_1, x_2) : x_1 \geq 0, x_2 \geq 0\}$.

Now let's consider functions to ℓ_1 . (The following was also Exercise 5.3, but we give a proof here.)

Lemma 8.6 (Cut-cone representation of ℓ_1 pseudometrics). *Let V be a finite set and let $f : V \rightarrow \ell_1$ be any function. Then there are constants $\{c_S\}_{S \subset V} \subset [0, \infty)$ such that*

$$\|f(u) - f(v)\|_1 = \sum_{S \subset V} c_S |\mathbf{1}_S(u) - \mathbf{1}_S(v)| \text{ for all } u, v \in V \quad (8.2)$$

Proof. Recall that elements of ℓ_1 are infinite sequences. Thus we can write

$$f(u) = (f_1(u), f_2(u), \dots) \quad (8.3)$$

for some functions $f_i : V \rightarrow \mathbb{R}$.

For each i , apply Lemma 8.4 to f_i to get nonnegative constants $\{c_{i,S}\}_{S \subset V}$ such that

$$|f_i(u) - f_i(v)| = \sum_{S \subset V} c_{i,S} |\mathbf{1}_S(u) - \mathbf{1}_S(v)| \quad (8.4)$$

Then

$$\|f(u) - f(v)\|_1 = \sum_{i=1}^{\infty} |f_i(u) - f_i(v)| \quad (8.5)$$

$$= \sum_{i=1}^{\infty} \sum_{S \subset V} c_{i,S} |\mathbf{1}_S(u) - \mathbf{1}_S(v)| \quad (8.6)$$

$$= \sum_{S \subset V} \left(\sum_{i=1}^{\infty} c_{i,S} \right) |\mathbf{1}_S(u) - \mathbf{1}_S(v)| \quad (8.7)$$

so define $c_S = \sum_{i=1}^{\infty} c_{i,S}$.

(Some analysis remarks: We could change the order of summation in the argument above because all the terms are nonnegative. Also, since $\|f(u) - f(v)\|_1 < \infty$ for all $u, v \in V$, we know that $c_S < \infty$ for all $S \subset V$.) \square

Remark 8.7. In, we didn't care about the whole space ℓ_1 . We only cared about the image of f , which is a finite subset of ℓ_1 . It is a fact that every n -point subset of ℓ_1 can be embedded isometrically in ℓ_1^m for some $m \in \mathbb{N}$. (See Definition 9.1 for what this means and see Exercise 9.9 for a more precise statement.) A consequence of this fact is that we do not actually need to work with infinite sums in the proof of Lemma 8.6. (Of course, the convergence issues just get shifted over to Exercise 9.9.)

8.3 Sparsest cut

Given a graph $G = (V, E)$ and a cut $S \subset V$, define the *sparcity* of the cut to be

$$\text{spar}(S) = \frac{|E(S, S^c)|}{\min(|S|, |S^c|)} \quad (8.8)$$

The sparsest cut problem is to find a cut S that minimizes the sparsity. As stated in Theorem 2.6 this is NP-hard. Like with max-cut, we will give an approximation algorithm.

Definition 8.8. Suppose we have a minimization problem and the optimal value is OPT . Let APPROX be the value given by our algorithm. If we know that

$$\text{OPT} \leq \text{APPROX} \leq \rho \cdot \text{OPT} \tag{8.9}$$

then our algorithm is called a ρ -approximation algorithm

The difference between Definition 6.4 and Definition 8.8 is in the position of the ρ . This is because for minimization problems, we always have $\text{APPROX} \geq \text{OPT}$, while the inequality is reversed for maximization problems.

Here is what we will show about sparsest cut. This is the main theorem of the class.

Theorem 8.9. *There is a constant $C > 0$ such that the following is true. Given a graph $G = (V, E)$, we can use linear programming to find a cut with sparsity at most a factor $C \log |V|$ from the minimum sparsity. That is, if the minimum sparsity is OPT , then our algorithm will produce a cut with sparsity APPROX which satisfies*

$$\text{OPT} \leq \text{APPROX} \leq (C \log |V|) \text{OPT} \tag{8.10}$$

Theorem 8.9 asserts that there is an efficient $C \log |V|$ -approximation algorithm for sparsest cut. Unlike the approximation algorithm we saw for max-cut, the approximation ratio for the algorithm mentioned below depends on the size of the graph.

Remark 8.10. The constant C in Theorem 8.9 is an absolute constant. We could compute what the constant is if we wanted, but as is often the case in theoretical computer science and analysis, we care more about order of magnitude than about specific constants. We can “hide” the constant by using big- O notation. With this notation, Theorem 8.9 says that there is a $O(\log |V|)$ -approximation algorithm for sparsest cut.

Remark 8.11. The proof we will give for Theorem 8.9 was independently discovered by two groups of researchers: (1) Linial, London, Rabinovich (published in 1995), and (2) Aumann, Rabani (published in 1998).

8.4 A reformulation of sparsest cut

The first thing we do is take care of the denominator $\min(|S|, |S^c|)$. To get rid of the minimum, we use the following basic inequality (which, as seen in class, is clearly too advanced for me).

Lemma 8.12. *If $a, b > 0$ and $a + b = n$, then*

$$\frac{ab}{n} \leq \min(a, b) \leq \frac{2ab}{n}. \quad (8.11)$$

Proof. Use $ab = \min(a, b) \max(a, b)$ and $\frac{n}{2} \leq \max(a, b) \leq n$. \square

Since $|S| + |S^c| = |V|$, (8.12) implies

$$\frac{1}{2}|V| \cdot \frac{|E(S, S^c)|}{|S||S^c|} \leq \frac{|E(S, S^c)|}{\min(|S|, |S^c|)} \leq |V| \cdot \frac{|E(S, S^c)|}{|S||S^c|} \quad (8.12)$$

From (2.9), we get:

$$|E(S, S^c)| = \sum_{uv \in E} |\mathbf{1}_S(u) - \mathbf{1}_S(v)| \quad (8.13)$$

$$|S||S^c| = \frac{1}{2} \sum_{u, v \in V} |\mathbf{1}_S(u) - \mathbf{1}_S(v)|. \quad (8.14)$$

The sum in (8.14) is over all ordered pairs $(u, v) \in V \times V$. You are asked to prove (8.14) in Exercise 9.2.

We define the *uniform sparsity* to be

$$\text{uspar}(S) = \frac{\sum_{uv \in E} |\mathbf{1}_S(u) - \mathbf{1}_S(v)|}{\sum_{u, v \in V} |\mathbf{1}_S(u) - \mathbf{1}_S(v)|} = \frac{|E(S, S^c)|}{2|S||S^c|} \quad (8.15)$$

so that (8.12) implies

$$|V| \text{uspar}(S) \leq \text{spar}(S) \leq 2|V| \text{uspar}(S). \quad (8.16)$$

One consequence of (8.16) is that if you know the precisely minimum value of $\text{uspar}(S)$, then you know the minimum value of $\text{spar}(S)$ within a factor of 2.

Remark 8.13. See Exercise 9.3 for a generalization of sparsest cut, and for some explanation of the term “uniform” above.

Motivated by the above discussion, let’s consider the following minimization problem

$$\begin{aligned} &\text{unknowns: } S \subset V \\ &\text{minimize: } \frac{\sum_{uv \in E} |\mathbf{1}_S(u) - \mathbf{1}_S(v)|}{\sum_{u, v \in V} |\mathbf{1}_S(u) - \mathbf{1}_S(v)|} \end{aligned} \quad (\text{SC})$$

We’ll refer to (SC) to it as the sparsest cut problem. From now on, we can ignore the original version of sparsest cut given in Section 2.3.3. This is because a ρ -approximation algorithm for (SC) would give a 2ρ -approximation algorithm for the original Section 2.3.3 version of sparsest cut. Unlike in our analysis of max-cut, here we do not care about constant factors.

We cannot copy what we did with min-cut to get a LP relaxation for (SC) because the objective function here is more complicated. We’ll need to take another route, and this is where pseudometric spaces come in.

8.5 Three minimization problems related to sparsest cut

Consider the following three minimization problems. Note that the objective function is the same in all three. The only differences are in the types of pseudometrics that we allow.

1. In the first problem, we minimize over cut pseudometrics:

$$\begin{aligned}
 &\text{unknowns: } d : V \times V \rightarrow \mathbb{R} \\
 &\text{minimize: } \frac{\sum_{uv \in E} d(u, v)}{\sum_{u, v \in V} d(u, v)} \qquad (\text{SC-CUT}) \\
 &\text{subject to: } d \text{ is a cut pseudometric}
 \end{aligned}$$

Observe that (SC-CUT) is the same minimization question as the uniform sparsest cut problem (SC).

2. In the second problem, we minimize over ℓ_1 pseudometrics:

$$\begin{aligned}
 &\text{unknowns: } d : V \times V \rightarrow \mathbb{R} \\
 &\text{minimize: } \frac{\sum_{uv \in E} d(u, v)}{\sum_{u, v \in V} d(u, v)} \qquad (\text{SC-}\ell_1) \\
 &\text{subject to: } d \text{ is a } \ell_1 \text{ pseudometric}
 \end{aligned}$$

3. In the third problem, we place no further constraints on the pseudometric:

$$\begin{aligned}
 &\text{unknowns: } d : V \times V \rightarrow \mathbb{R} \\
 &\text{minimize: } \frac{\sum_{uv \in E} d(u, v)}{\sum_{u, v \in V} d(u, v)} \qquad (\text{SC-}\Psi\text{M}) \\
 &\text{subject to: } d \text{ is a pseudometric}
 \end{aligned}$$

(The symbol Ψ is the capital Greek letter psi. I use “ ΨM ” as an abbreviation for “pseudometric.”) (SC- ΨM) is known as the *Leighton–Rao LP relaxation* of sparsest cut. As the name suggests, (SC- ΨM) is actually a linear program. That is because it is equivalent to the following:

$$\begin{aligned}
 &\text{unknowns: } (d_{u,v})_{u,v \in V} \\
 &\text{minimize: } \sum_{uv \in E} d_{u,v} \\
 &\text{subject to: } \sum_{u,v \in V} d_{u,v} = 1 \\
 &\qquad d_{u,u} = 0 \text{ for all } u \in V \\
 &\qquad d_{u,v} \geq 0 \text{ for all } u, v \in V \\
 &\qquad d_{u,v} = d_{v,u} \text{ for all } u, v \in V \\
 &\qquad d_{u,w} \leq d_{u,v} + d_{v,w} \text{ for all } u, v, w \in V
 \end{aligned} \qquad (\text{SC-}\Psi\text{M}')$$

All the constraints are linear! The objective function is linear too!

Since every cut pseudometric is an ℓ_1 pseudometric and every ℓ_1 pseudometric is a pseudometric (Exercise 9.1), we have:

$$\text{OPT}(\text{SC-}\Psi\text{M}) \leq \text{OPT}(\text{SC-}\ell_1) \leq \text{OPT}(\text{SC-CUT}). \quad (8.17)$$

In fact, Lemma 8.6 implies

$$\text{OPT}(\text{SC-}\ell_1) = \text{OPT}(\text{SC-CUT}). \quad (8.18)$$

The argument is very similar to the proof of Theorem 4.5. You are asked to show this in Exercise 9.5.

Thanks to (8.18), we only need to study the relationship between $\text{OPT}(\text{SC-}\Psi\text{M})$ and $\text{OPT}(\text{SC-}\ell_1)$. We will get to this tomorrow. See also Exercise 9.6.

9 Day 4 exercises

The following exercises are especially recommended because they will be useful for upcoming classes: Exercise 9.5, Exercise 9.6

Exercise 9.1. (👉) Show that every cut pseudometric is a line pseudometric, and that every line pseudometric is an ℓ_1 pseudometric. (The purpose of this exercise is to make sure you understand the definitions.)

Exercise 9.2. (👉) For $S \subset V$, prove $|S||S^c| = \frac{1}{2} \sum_{u,v \in V} |\mathbb{1}_S(u) - \mathbb{1}_S(v)|$.

Exercise 9.3. (👉👉) Let $G = (V, E)$ be a graph. Let $C : E \rightarrow \mathbb{R}_{>0}$ and $D : V \times V \rightarrow \mathbb{R}_{\geq 0}$ be two functions. $C(e)$ is called the *capacity* of the edge e . $D(u, v)$ is called the *demand* associated to the ordered pair $(u, v) \in V \times V$. Define

$$\text{OPT} = \min_{\emptyset \subsetneq S \subsetneq V} \frac{\sum_{uv \in E} C(uv) |\mathbb{1}_S(u) - \mathbb{1}_S(v)|}{\sum_{u,v \in V} D(u,v) |\mathbb{1}_S(u) - \mathbb{1}_S(v)|} \quad (9.1)$$

The minimization problem (9.1) is called *sparsest cut with general demands*. (Note: *Sparsest cut with uniform demands* refers to the problem when $D(u, v) = 1$ for all $u, v \in V$. If in addition, when $C(e) = 1$ for all $e \in E$, then the fraction in the right-hand side of (9.1) is precisely the uniform sparsity (8.15).)

One motivation for the problem above is in studying a “multicommodity flow” problem. Consider the following situation. For $u, v \in V$, we would like to ship the quantity $D(u, v)$ of some “commodity” from u to v along the graph. Here, the direction matters.

However, there are two things making this difficult: (1) For each pair (u, v) , the type of commodity is different. (2) For each $e \in E$, the total amount of commodity (of all types) being transported across e (in both directions) cannot exceed $C(e)$.

Let $\lambda^* \geq 0$ be the largest number such that the following is true: There exists a way to ship commodities along the graph so that for any pair (u, v) , at least $\lambda^* \cdot D(u, v)$ of the commodity is being shipped from u to v . Show that

$$\lambda^* \leq \text{OPT} \tag{9.2}$$

Exercise 9.4. (🍷) Consider the max-flow and weighted min-cut problems from Exercise 3.6 and Exercise 5.6 respectively. How do these show up as a special case of Exercise 9.3? What does (9.2) say about the max-flow and min-cut problems? (Hint: It says something weaker than the max-flow min-cut theorem of Exercise 5.7.)

Exercise 9.5. (🍷🍷) Show $\text{OPT}(\text{SC-}\ell_1) = \text{OPT}(\text{SC-CUT})$. (Hint: It is helpful if you understand the proof of Theorem 4.5 well.)

Exercise 9.6. (🍷🍷🍷) We already know from (8.17) that $\text{OPT}(\text{SC-}\Psi\text{M}) \leq \text{OPT}(\text{SC-}\ell_1)$. What other kind of relation between $\text{OPT}(\text{SC-}\Psi\text{M})$ and $\text{OPT}(\text{SC-}\ell_1)$ should we be looking for? What kind of lemma would we need to show if we want to prove this kind of relation? This is an intentionally vague and open-ended question. (This is also the kind of question you ask yourself frequently when doing mathematical research.)

Definition 9.1. Let (X, d) be a metric space, let $p \in [1, \infty]$, let $n \in \mathbb{N}$. We say X *embeds isometrically into* ℓ_p^n if there is a map $f : X \rightarrow \ell_p^n$ such that

$$\|f(u) - f(v)\|_p = d(u, v) \text{ for all } u, v \in X. \tag{9.3}$$

(Note that the concept of isometric embeddings already appeared in Exercise 7.9.)

Exercise 9.7. (🍷🍷) Show that every n -point metric space embeds isometrically into ℓ_∞^{n-1} .

Exercise 9.8. (🍷) Show that if X is any n -point subset of ℓ_2 , then X embeds isometrically into ℓ_2^{n-1} . Here, the metric on X is the ℓ_2 metric.

Exercise 9.9. (🍷🍷🍷) Let $n \in \mathbb{N}$. Show that there exists a $N \in \mathbb{N}$ such that the following is true. If $p \in [1, \infty]$ and X is any n -point subset of ℓ_p , then X embeds isometrically into ℓ_p^N . Here, the metric on X is the ℓ_p metric. (Note: the statement is true with $N = \binom{n}{2}$.)

10 Day 5

10.1 Metric embeddings

We begin with the following definition.

Definition 10.1. Let (X, d_X) and (Y, d_Y) be metric spaces. We say $f : X \rightarrow Y$ is an *isometric embedding* if it satisfies

$$d_Y(f(u), f(v)) = d_X(u, v) \text{ for all } u, v \in X. \quad (10.1)$$

Example 10.2. A 3-point metric space with the discrete metric embeds isometrically into ℓ_2^2 . A 4-point metric space with the discrete metric embeds isometrically into ℓ_1^2 but not into ℓ_2^2 . This was part of Exercise 7.9.

Definition 10.3. Let (X, d) be a metric space, let $p \in [1, \infty]$, let $n \in \mathbb{N}$, and let $D \geq 1$. We say X *embeds into ℓ_p^n with distortion D* if there is a map $f : X \rightarrow \ell_p^n$ such that

$$\|f(u) - f(v)\|_p \leq d(u, v) \leq D \cdot \|f(u) - f(v)\|_p \text{ for all } u, v \in X. \quad (10.2)$$

We also say that f is an embedding of X into ℓ_p^n with distortion D . (There is an analogous definition with ℓ_p in place of ℓ_p^n .)

Remark 10.4. For metric spaces (X, d_X) and (Y, d_Y) , the definition of “ X embeds into Y with distortion D ” is not the most straightforward generalization of (10.2), but it is not too different. See Exercise 11.1.

Example 10.5. Let (X, d) be a metric space. Then the following statements are equivalent.

1. (X, d) embeds into ℓ_p with distortion 1.
2. (X, d) embeds isometrically into ℓ_p .
3. The metric d is an ℓ_p metric on X . (Example 8.3 generalizes to ℓ_p metrics.)

Example 10.6. Let X be a 3-point metric space with the discrete metric. Then X embeds into \mathbb{R} with distortion 2, and X cannot be embedded into \mathbb{R} with distortion D for any $D < 2$. (Note that \mathbb{R} is the same as ℓ_1^1 , or more generally, ℓ_p^1 for any $p \in [1, \infty]$.)

Now we connect embeddings of finite metric spaces into ℓ_1 to the LP relaxation of sparsest cut (SC- Ψ M).

Lemma 10.7. *Let $G = (V, E)$ be a graph. Suppose that every $|V|$ -point metric space embeds into ℓ_1 with distortion D . Then*

$$\text{OPT}(\text{SC-CUT}) \leq D \cdot \text{OPT}(\text{SC-}\Psi\text{M}) \quad (10.3)$$

Proof. Let $d^* : V \times V \rightarrow \mathbb{R}$ be a pseudometric that attains the minimum in $\text{OPT}(\text{SC-}\Psi\text{M})$. Use the hypothesis of the lemma to get a map $f : V \rightarrow \ell_1$ with distortion at most D . That is,

$$\|f(u) - f(v)\|_1 \leq d^*(u, v) \leq D \cdot \|f(u) - f(v)\|_1 \quad (10.4)$$

Then

$$\text{OPT}(\text{SC-}\Psi\text{M}) = \frac{\sum_{uv \in E} d^*(u, v)}{\sum_{u, v \in V} d^*(u, v)} \quad (10.5)$$

$$\geq \frac{\sum_{uv \in E} \|f(u) - f(v)\|_1}{D \sum_{u, v \in V} \|f(u) - f(v)\|_1} \quad (10.6)$$

$$\geq \frac{1}{D} \text{OPT}(\text{SC-}\ell_1) \quad (10.7)$$

$$= \frac{1}{D} \text{OPT}(\text{SC-CUT}) \quad (10.8)$$

where we used (8.18) in the last line. \square

So then the question becomes: How well can we embed pseudometric spaces into ℓ_1 ?

10.2 Rounding the Leighton–Rao LP

Let us delay addressing the question Here is what we have so far. Given a graph $G = (V, E)$, we can form the Leighton–Rao LP relaxation (SC- Ψ M). By running an efficient algorithm to solve (SC- Ψ M), we get a pseudometric $d^* : V \times V \rightarrow \mathbb{R}$.

Suppose we find a function $f : V \rightarrow \ell_p$ such that (10.4) is satisfied. (In the next section, we will talk about what we can take D to be.) Then we apply Lemma 8.6 to f to get constants $\{c_S\}_{S \subset V} \subset [0, \infty)$. By examining the proof of Lemma 8.6, we see that if $c_S > 0$, then there exists an i and an x such that $S = \{u \in V : f_i(u) < x\}$. Such a cut is called a *threshold cut*.

Furthermore, we can show the following. This is analogous to the procedure described in Section 6.1 for min-cut.

Lemma 10.8. *Let $G = (V, E)$ be a graph. Suppose that every $|V|$ -point metric space embeds into ℓ_1 with distortion D . Then there exists a threshold cut S such that*

$$\text{uspar}(S) \leq D \cdot \text{OPT}(\text{SC-CUT}) \quad (10.9)$$

Proof. You are asked to do this in Exercise 11.2. \square

10.3 Bourgain’s theorem

Theorem 10.9 (Bourgain’s theorem). *There is a constant $C > 0$ so that the following is true: Any finite metric space (X, d) embeds into ℓ_1 with distortion $C \log |X|$.*

We are interested in computations. If we inspect the proof of Bourgain’s theorem, we actually get the following.

Theorem 10.10 (Bourgain’s theorem, more detailed). *There is a constant $C > 0$ so that the following is true: Let (X, d) be an n -point metric space. Then there is an efficient randomized algorithm to compute an embedding of (X, d) into ℓ_1^N with distortion $C \log n$, where $N = \lfloor C(\log n)^2 \rfloor$.*

The proof is clever. It does not require anything too advanced. However, we do not have time to cover it this week.

Remark 10.11. Bourgain died of cancer in 2018. Terry Tao wrote a nice blog post with some stories about Bourgain <https://terrytao.wordpress.com/2018/12/29/jean-bourgain/>. Be sure to take a look at the photo of Bourgain’s paper.

10.4 Proof of the $O(\log |V|)$ -approximation algorithm

Now we are ready to prove the main theorem for this class!

Proof of Theorem 8.9. We give an efficient $C \log |V|$ -approximation algorithm for sparsest cut and explain why it works.

1. Starting with a graph $G = (V, E)$, form the Leighton–Rao LP relaxation (**SC- Ψ M**).
2. Solve (**SC- Ψ M**) with an efficient polynomial time algorithm. This gives a pseudometric d^* on V .
 - This step is possible because of Theorem 2.14.
3. Efficiently compute an embedding $f : (V, d^*) \rightarrow \ell_1^N$ with distortion $C \log |V|$, where $N \leq C(\log |V|)^2$.
 - This step is possible because of Bourgain’s theorem (Theorem 10.10).
4. Consider all threshold cuts, i.e., cuts of the form $S = \{u \in V : f_i(u) < x\}$ where $i \in \{1, \dots, N\}$ and $x \in \mathbb{R}$. Choose the threshold cut S^* that minimizes $\text{uspar}(S)$. This is the cut returned by our approximation algorithm.
 - We can apply Lemma 10.8 with $D = C \log |V|$ to conclude that

$$\text{uspar}(S^*) \leq C \log |V| \cdot \text{OPT}(\text{SC-CUT}). \quad (10.10)$$

- To see that this step runs in polynomial time, note the following: For each $i = 1, \dots, N$, there are at most $|V|$ threshold cuts. Thus, there are at most $N|V| \leq C|V|(\log |V|)^2$ threshold cuts. \square

Note that we relied on two theorems that we did not prove: (1) Theorem 2.14, the existence of a polynomial time algorithm for LPs, and (2) Theorem 10.10, Bourgain’s theorem on embeddings into ℓ_1 .

10.5 Sharpness of this approximation algorithm

By Lemma 10.7 and Theorem 10.9, there is an absolute constant $C > 0$ such that for any graph $G = (V, E)$,

$$\frac{\text{OPT}(\text{SC-CUT})}{\text{OPT}(\text{SC-}\Psi\text{M})} \leq C \log |V|. \quad (10.11)$$

In this section, we show that the $\log |V|$ factor in the Leighton–Rao LP relaxation cannot be improved.

Theorem 10.12. *There exist a constant $c > 0$ and a sequence of graphs $G_n = (V_n, E_n)$ such that $|V_n| \rightarrow \infty$ and*

$$\frac{\text{OPT}(\text{SC-CUT})}{\text{OPT}(\text{SC-}\Psi\text{M})} \geq c \log |V_n|. \quad (10.12)$$

A concise way to say the above results is: “The integrality gap of the Leighton–Rao LP relaxation of sparsest cut is $O(\log |V|)$.”

To prove Theorem 10.12, we need to find graphs where $\text{OPT}(\text{SC-CUT})$ is “large” and $\text{OPT}(\text{SC-}\Psi\text{M})$ is “small.” First, we make the following definition.

Definition 10.13. The *edge expansion* or *Cheeger constant* of $G = (V, E)$ is the quantity

$$h(G) = \min_{\emptyset \subsetneq S \subsetneq V} \text{spar}(S) = \min_{0 < |S| \leq \frac{1}{2}|V|} \text{spar}(S) = \min_{0 < |S| \leq \frac{1}{2}|V|} \frac{|E(S, S^c)|}{|S|}. \quad (10.13)$$

In particular, if $S \subset V$ and $0 < |S| \leq \frac{1}{2}|V|$, then

$$|E(S, S^c)| \geq h(G)|S| \quad (10.14)$$

Informally, if $h(G)$ is not too close to zero, given any subset $S \subset V$ that is not too large, there are “many” edges coming out of S .

It was convenient to use $\text{spar}(S)$ in Definition 10.13, but now we relate $h(G)$ back to (SC-CUT) , which was defined in terms of $\text{uspar}(S)$.

Lemma 10.14. *For any graph $G = (V, E)$, we have*

$$\text{OPT}(\text{SC-CUT}) \geq \frac{h(G)}{2|V|}. \quad (10.15)$$

Proof. We have

$$\text{OPT}(\text{SC-CUT}) = \min_{0 < |S| \leq \frac{1}{2}|V|} \frac{|E(S, S^c)|}{2|S||S^c|} \geq \frac{h(G)}{2|V|} \quad (10.16)$$

where in the inequality, we used the definition of $h(G)$ and the fact that $|S^c| \leq |V|$. \square

Definition 10.15. A graph $G = (V, E)$ is d -regular if each vertex is the endpoint of d edges.

Lemma 10.16 (d -regular implies $\text{OPT}(\text{SC-}\Psi\text{M})$ is small). *If $G = (V, E)$ is d -regular, then*

$$\text{OPT}(\text{SC-}\Psi\text{M}) \leq \frac{d}{|V|} \cdot \frac{1}{\lfloor \log_d(V/2) \rfloor} \quad (10.17)$$

Proof. Let $d_G : V \times V \rightarrow \mathbb{R}$ be the graph metric, as defined in Example 6.11. (Warning: In this proof the letter d is used both for d -regular and for the graph metric d_G .) Since

$$\text{OPT}(\text{SC-}\Psi\text{M}) \leq \frac{\sum_{uv \in E} d_G(u, v)}{\sum_{u, v \in V} d_G(u, v)} \quad (10.18)$$

we just need to give an upper bound for the numerator and a lower bound for the denominator. Let us try to bound both by an expression in $|V|$.

We can compute the numerator:

$$\sum_{uv \in E} d_G(u, v) = \sum_{uv \in E} 1 = |E| = \frac{d|V|}{2} \quad (10.19)$$

In the last line, we use the fact that G is d -regular and a double-counting argument.

Giving a lower bound on the denominator takes more work. We actually need to use the fact that G is d -regular. The informal idea is that since G is d -regular, there are “many” pairs of vertices that are “far” from each other. Now we make this idea precise.

For $u \in V$ and $r \in \{0, 1, \dots\}$, we have

$$|\{v \in V : d_G(u, v) = r\}| \leq d^r. \quad (10.20)$$

(To show this, use the fact that G is d -regular, and induct on r .) Therefore,

$$|\{v \in V : d_G(u, v) \leq r - 1\}| \leq 1 + d + d^2 + \dots + d^{r-1} < d^r. \quad (10.21)$$

Thus, if we let $r_0 = \lfloor \log_d \frac{|V|}{2} \rfloor$, then

$$|\{v \in V : d_G(u, v) \geq r_0\}| \geq |V| - d^{r_0} \geq |V| - \frac{|V|}{2} = \frac{|V|}{2} \quad (10.22)$$

This is true for each fixed $u \in V$, so

$$\sum_{u, v \in V} d_G(u, v) \geq \sum_{u \in V} r_0 \frac{|V|}{2} = r_0 \frac{|V|^2}{2} \quad (10.23)$$

Thus, using (10.18), (10.19), and (10.23), we have

$$\text{OPT}(\text{SC-}\Psi\text{M}) \leq \frac{\frac{d|V|}{2}}{r_0 \frac{|V|^2}{2}} = \frac{d}{r_0|V|} = \frac{d}{|V|} \cdot \frac{1}{\lfloor \log_d(|V|/2) \rfloor} \quad (10.24)$$

which completes the proof. \square

Lemma 10.14 tells us how to make $\text{OPT}(\text{SC-CUT})$ large, and Lemma 10.16 tells us a way to make $\text{OPT}(\text{SC-}\Psi\text{M})$ small. The following theorem tells us it is possible to do both at the same time.

Theorem 10.17 (Existence of constant-degree expander graphs). *There exists a number $d \in \mathbb{N}$, a constant $h_0 > 0$, and a sequence of finite graphs $G_n = (V_n, E_n)$ such that*

1. G_n is d -regular.
2. $h(G_n) \geq h_0$ for all n
3. $\lim_{n \rightarrow \infty} |V_n| = \infty$

Proof. Proof omitted. See Remark 10.18. □

Remark 10.18. Theorem 10.17 is true for any $d \geq 3$, but we only need it for a single d . It can be proved using the probabilistic method. There are also explicit constructions, but those are harder.

Using Lemma 10.14, Lemma 10.16, and Theorem 10.17 we can prove Theorem 10.12. See Exercise 11.3.

One consequence of Theorem 10.12 is that the $\log |X|$ in Bourgain's theorem Theorem 10.9 cannot be improved either.

Corollary 10.19 (Bourgain's theorem is sharp). *There exist a constant $c > 0$ and a sequence of metric spaces (X_n, d_n) such that $|X_n| \rightarrow \infty$ and (X_n, d_n) does not embed into ℓ_1 with distortion $c \log |X_n|$.*

Proof. You are asked to do this in Exercise 11.4. □

Remark 10.20. Bourgain's theorem Theorem 10.9 is about metric spaces. To show that it is sharp, we do not stay in the field of metric geometry. We use graph theory (existence of constant degree expanders) as well as theoretical computer science (the sparsest cut problem).

10.6 End of class remark

As stated in the introduction, this class is not intended to be a class on graph theory, theoretic computer science, or metric geometry. Instead, the main goal of this class is to see how these areas are related through the sparsest cut problem.

From here, there are many areas to explore. For example, we have used a non-trivial theorem from each of these areas without proof:

1. Theoretical computer science: Solvability of linear programming in polynomial time (Theorem 2.14)

2. Metric geometry: Bourgain’s theorem on embeddings of metric spaces (Theorem 10.9)
3. Graph theory: Existence of constant-degree expander graphs (Theorem 10.17)

10.7 Bonus: Better approximation algorithms for sparsest cut

We will not prove anything in this section. In the class, we focused only on sparsest cut with uniform demands, but the discussion above also applied to sparsest cut with general demands when we make the appropriate modifications. (See Exercise 9.3 for the relevant definitions.)

The following results make a distinction between uniform versus general demands.

Theorem 10.21 (Arora, Rao, Vazirani 2004). *There is a constant $C > 0$ such that the following is true. There is an efficient $C\sqrt{\log |V|}$ -approximation algorithm for sparsest cut with uniform demands.*

Theorem 10.22 (Arora, Lee, Naor 2005). *There is a constant $C > 0$ such that the following is true. There is an efficient $C\sqrt{\log |V|} \log \log |V|$ -approximation algorithm for sparsest cut with general demands.*

We give a few ideas of the proofs. Both Theorem 10.21 and Theorem 10.22 consider the following problem, called the *Goemans–Linial semidefinite programming relaxation of sparsest cut*. (It is stated below only for uniform demands, but the problem for general demands is analogous. Again, see Exercise 9.3.)

$$\begin{aligned}
 &\text{unknowns: } d : V \times V \rightarrow \mathbb{R} \\
 &\text{minimize: } \frac{\sum_{uv \in E} d(u, v)}{\sum_{u, v \in V} d(u, v)} \qquad \text{(SC-NEG)} \\
 &\text{subject to: } d \text{ is a negative type pseudometric}
 \end{aligned}$$

We will not define “semidefinite programming” or “negative type pseudometric,” but here are some facts.

Lemma 10.23. *Every ℓ_1 pseudometric is a negative type pseudometric.*

Lemma 10.24. (SC-NEG) *is a semidefinite program.*

Theorem 10.25 (Grötschel, Lovász, Schrijver 1981). *Semidefinite programming can be solved with a polynomial time algorithm.*

Theorem 10.26 (Arora, Rao, Vazirani 2004). *For sparsest cut with uniform demands, there is a constant $C > 0$ such that*

$$\frac{\text{OPT}(\text{SC-CUT})}{\text{OPT}(\text{SC-NEG})} \leq C\sqrt{\log |V|}. \qquad (10.25)$$

Theorem 10.27 (Arora, Lee, Naor 2005). *For sparsest cut with general demands, there is a constant $C > 0$ such that*

$$\frac{\text{OPT}(\text{SC-general-demands-CUT})}{\text{OPT}(\text{SC-general-demands-NEG})} \leq C \sqrt{\log |V|} \log \log |V|. \quad (10.26)$$

The proof of Theorem 10.26 uses structural results for negative type metric spaces. The proof of Theorem 10.27 uses the following embedding theorem. The distortion here is smaller than the distortion in Bourgain's theorem (Theorem 10.9) because here we only consider negative type metric spaces.

Theorem 10.28 (Arora, Lee, Naor 2005). *There is a constant $C > 0$ so that the following is true: Any finite negative type metric space (X, d) embeds into ℓ_1 with distortion $C \sqrt{\log |X|} \log \log |X|$.*

We do not know if the $\sqrt{\log |V|}$ in Theorem 10.26 is sharp. The current best lower bound is $e^{c\sqrt{\log \log |V|}}$.

We do not know if the $\sqrt{\log |V|} \log \log |V|$ in Theorem 10.27 is sharp either. The current best lower bound is $\sqrt{\log |V|}$, so the upper and lower bounds are not too far from each other. This lower bound was proved very recently, and is a consequence of the following theorem.

Theorem 10.29 (Naor, Young 2017). *There exists a $c > 0$ such that the following is true. Let $G = (V, E)$ be the Cayley graph of the discrete 5-dimensional Heisenberg group $\mathbb{H}_{\mathbb{Z}}^5$ with the standard generating set. Let $B_n \subset V$ be a ball of radius n in G . Then B_n (with the graph metric) is a negative type metric space, but it cannot be embedded into ℓ_1 with distortion $c\sqrt{\log |B_n|}$. As a consequence, for the graphs B_n , we have*

$$\frac{\text{OPT}(\text{SC-general-demands-CUT})}{\text{OPT}(\text{SC-general-demands-NEG})} \geq c\sqrt{\log |B_n|}. \quad (10.27)$$

Some topics in math relevant to the proof of Theorem 10.29 include: functional analysis, geometric group theory, harmonic analysis, sub-Riemannian geometry, geometric measure theory, ergodic theory, group representations, and metric differentiation. I will not make any attempt to explain the statement of Theorem 10.29 here. Have fun!

11 Day 5 exercises

Exercise 11.1. (🐶) Let (X, d_X) and (Y, d_Y) be metric spaces. We say $f : X \rightarrow Y$ is an *embedding with distortion D* if there are constants $0 < c_1 \leq c_2$ such that $c_1 c_2 \leq D$ and

$$c_1 \cdot d_Y(f(u), f(v)) \leq d_X(u, v) \leq c_2 \cdot d_Y(f(u), f(v)) \text{ for all } u, v \in X. \quad (11.1)$$

We say X *embeds into Y with distortion D* if there exists an f as in (11.1). Show that when (Y, d_Y) is ℓ_p or ℓ_p^n , then this definition is equivalent to Definition 10.3.

Exercise 11.2. (👉👉) Prove Lemma 10.8.

Exercise 11.3. (👉) Using Lemma 10.14, Lemma 10.16, and Theorem 10.17, prove Theorem 10.12. Apart from the lemmas, you only need to use some basic algebra.

Exercise 11.4. (👉) Using Theorem 10.12, prove Corollary 10.19.

12 Acknowledgments

I attended a computational and metric geometry class taught by Yury Makarychev in Winter 2019, which introduced me to the subject of this Mathcamp class. His course notes have also been very helpful: <https://home.ttic.edu/~yury/courses/geom2019/>

I am super thankful to Jaume de Dios Pont and Andrea Olivo for helping me prepare for this class. In the last week of June, I said to them “I am teaching a class about sparsest cut in two weeks. Would you like to hear me talk about it now? However, I have not prepared anything yet, and I have not thought about sparsest cut for over a year.” They said yes and were extremely helpful the whole time, as I struggled to recall the material and explain it to them. (I think this method is a fantastic away to review things you learned in the past. However, it requires the help of some very patient friends.)