Advanced Algorithms

Valerio Venanzi

Abstract

This document contains notes taken during the *Advanced Algorithms* course, held by Flavio Chierichetti during the academic year 2022-23.

Contents

1 Introduction

1.1 Categorization of algorithms

Algorithms can be categorized by a number of criteria:

- Whether the input is all available at the beggining or not
  - Offline
  - Online
- Whether some choices are randomized or not
  - Deterministic
  - Randomized
  - Both can be quantum algorithms
- How good the algorithm does on some inputs
  - Worst-case input
  - Average-case input
- Whether we distribute our algorithms execution among multiple processors or not
  - Single processor
  - Parallel/Distributed

1.2 Graphs

If S is a set and k is a non-negative integer, then

$$\binom{S}{k} = \{T \mid T \subseteq S \wedge |T| = k\}$$

The cardinality of such set is $|\binom{S}{2}| = \frac{|S|!}{k!(|S|-k)!} = \binom{|S|}{k}$

Many algorithms seen in this course solve problems on graphs. A graph is $G(V, E)$, where V is the set of vertices and E, the set of edges, is $E \subseteq \{\{u, v\} \mid u, v \in V \wedge u \neq v\} = \binom{V}{2}$.

1.3 Approximations

Most problems during the course are going to be approached in the following manner. We have a problem that is NP-Hard (or perhaps NP-Complete), so we don't know how to solve it efficiently. What we know how to do is to solve such problems efficiently but approximating the solution; that is, the solution proposed by our algorithm is worse than the optimal solution to the problem.

How do we denote such approximations? If a problem is a *maximization* problem (e.g. a set that has some property and its cardinality is as large as possible) then our approximation algorithm will return something that is smaller than the optimal. In this case we denote the approximation as a number smaller than 1, possibly a fraction. For instance, a $\frac{1}{2}$-approximation means that the solution is at least half the optimal solution (e.g. if the optimal is 10, then our solution is at least 5).

If a problem is a minimization problem, then we reason in an analogous way. Our algorithm outputs a solution larger than the optimal. For instance, a 2-approximation means that our solution is at most twice as large as the optimal one.

The approach that will be used in many cases is the following. Given a hard problem, we given an algorithm (possibly randomized) that approximates the solution in polynomial time. The algorithm is then analyzed to prove various properties that we are interested in. We ugually prove that the algorithm is an approximation, and how good of an approximation it is. In the case of randomized algorithms, it might be that the solution not only is an approximation, it might be just wrong; in those cases we are interested in proving that the probability of an incorrect solution is so small that it can be ignored.

1.4 Math and CS notions

There are some notions about mathematics or computer science, used during the course, that are not part of the program, but are strictly related to many of the things seen, and may be worth explaining quickly.

Such notions are explained at the end of this document, in section Math and CS notions.

2 Problems on graphs

2.1 Max-Cut problem

Input: undirected graph $G(V, E)$

Question: what is the bipartition $(S, V \setminus S)$ of V that maximizes $|Cut(S, V \setminus S)|$?

Where a *cut* is defined as follows:

$$Cut(S, V \setminus S) = \{e \mid e \in E \wedge e \cap S \neq \emptyset \wedge e \cap (V \setminus S) \neq \emptyset\} = \{e \mid e \in E \wedge |e \cap S| = 1\}$$

This problem is NP-Hard, thus we don't know how to solve it efficiently. There exists a $(0.878\dots)$-approximation based on Semi-Definite Programming. We now see a simple $\frac{1}{2}$-approximation.

Theorem 2.1. If S^* is an optimal solution to Max-Cut on $G(V, E)$, and S is the set returned by Random-Cut, then

$$\mathbb{E}[|Cut(S, V \setminus S)|] \geq \frac{1}{2}|Cut(S^*, V \setminus S^*)|$$

Lemma 2.2. Let S be the set returned by Random-Cut, then $\forall e \in E$, $Pr[e \in Cut(S, V \setminus S)] = \frac{1}{2}$.

Algorithm 1 Approximation of Max-Cut

```
procedure RANDOM-CUT($V, E$)
    $S \leftarrow \emptyset$
    for each $v \in V$ do
        flip an independent/fair coin $c_v$
        if $c_v$ is heads then
            $S \leftarrow S \cup \{v\}$
        end if
    end for
    return $S$
end procedure
```

Proof. By definition, $e \in Cut(S, V \setminus S)$ iff $|S \cap e| = 1$.

If $e = \{v, w\}$, $e \in Cut(S, V \setminus S)$ iff $((v \in S \wedge w \notin S) \vee (v \notin S \wedge w \in S))$.

Let $\xi_1 =$ "$v \in S \wedge w \notin S$" and $\xi_2 =$ "$v \notin S \wedge w \in S$".

$$\begin{aligned} Pr[\xi_1] &= Pr[c_v = \text{Heads} \wedge c_w = \text{Tails}] \overset{\text{(by independence of the coins)}}{=} \\ &= Pr[c_v = \text{Heads}] \cdot Pr[c_w = \text{Tails}] = \\ &= \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} \end{aligned}$$

With same reasoning, $Pr[\xi_2] = Pr[c_v = \text{Tails}] \cdot Pr[c_w = \text{Heads}] = \frac{1}{4}$.

$Pr[e \in Cut(S, V \setminus S)] = Pr[\xi_1 \vee \xi_2] = Pr[\xi_1] + Pr[\xi_2] - Pr[\xi_1 \wedge \xi_2] = \frac{1}{4} + \frac{1}{4} + 0 = \frac{1}{2}$ ☐

Lemma 2.3. $\forall T \subseteq V$, $Cut(T, V \setminus T) \subseteq E$, $|Cut(T, V \setminus T)| \leq E$.

Proof. Trivial. ☐

Corollary 2.3.1. Let S be the set of nodes returned by Random-Cut,

$$\mathbb{E}[|Cut(S, V \setminus S)|] = \frac{|E|}{2}$$

Proof. By Lemma 2.2, $\forall e \in E$ $Pr[e \in Cut(S, V \setminus S)] = \frac{1}{2}$.

$\mathbb{E}[|Cut(S, V \setminus S)|] = \sum_{e \in E} Pr[e \in Cut(S, V \setminus S)] = \sum_{e \in E} \frac{1}{2} = \frac{|E|}{2}$ ☐

Recall Theorem 2.1.

Proof. The proof follows from Lemma 2.2 and Corollary 2.3.1. ☐

We now want to exploit a technique often used to improve the expected output of a randomized algorithm. We run the algorithm not once, but multiple times, and get the best result.

Algorithm 2 Repetition of Random-Cut

```
for $i = 1$ to $t$ do
    run Random-Cut$(V, E)$ independently
    let $S_i, V \setminus S_i$ be the resulting cut
    let $C_i = |Cut(S, V \setminus S)|$
end for
return a largest cut
```

Note that, at the end of the algorithm, we don't ask to return *the* largest cut but *a* largest cut, since it could happen that there are multiple cuts with the same cardinality. In fact, it can even happen, in general, that all the cuts found have the same cardinality.

Now, for $i \in [t] = \{1, 2, \ldots, t\}$, let us define $N_i = |E| - C_i$ ($N_i \geq 0$). N_i would be our error, the number of edges we didn't cut.

Let $0 < \varepsilon < 1$.

$$
\begin{aligned}
Pr[N_i \geq (1+\varepsilon)\mathbb{E}[N_i]] \overset{\text{(by Markov inequality)}}{\leq} \frac{1}{1+\varepsilon} &= \\
&= \frac{1+\varepsilon}{1+\varepsilon} - \frac{\varepsilon}{1+\varepsilon} = \\
&= 1 - \frac{\varepsilon}{1+\varepsilon} \leq \\
&\leq 1 - \frac{\varepsilon}{2}
\end{aligned}
$$

Note that $|E| = 2\mathbb{E}[C_i]$.

$$
\begin{aligned}
Pr[N_i \geq (1+\varepsilon)\mathbb{E}[N_i]] &= \\
&= Pr[|E| - C_i \geq (1+\varepsilon)(|E| - \mathbb{E}[C_i])] = \\
&= Pr[|E| - (1+\varepsilon)|E| \geq \\
&\geq C_i - (1+\varepsilon)\mathbb{E}[C_i]] = Pr[-\varepsilon|[|E] \geq \\
&\geq C_i - (1+\varepsilon)\mathbb{E}[C_i]] = \\
&= Pr[-2\varepsilon\mathbb{E}[C_i] + (1+\varepsilon)\mathbb{E}[C_i] \geq C_i] = \\
&= Pr[C_i \leq (1-\varepsilon)\mathbb{E}[C_i]]
\end{aligned}
$$

$C_i \leq (1-\varepsilon)\mathbb{E}[C_i]$ is our "bad event".

$C_i > (1-\varepsilon)\mathbb{E}[C_i]$ is our "good event". $Pr[C_i > (1-\varepsilon)\mathbb{E}[C_i]] \geq \frac{\varepsilon}{2}$.

Suppose we run for $t = \lceil \frac{2}{3} \ln \frac{1}{\delta} \rceil$ (δ will be the probability of error).

$$Pr[\forall i \in [t], C_i \leq (1-\varepsilon)\mathbb{E}[C_i]] \overset{\text{(by ind. of the } t \text{ runs)}}{=}$$

$$= \prod_{i=1}^{t} Pr[C_i \leq (1-\varepsilon)\mathbb{E}[C_i]] \leq \prod_{i=1}^{t}(1-\frac{\varepsilon}{2}) =$$

$$= (1-\frac{\varepsilon}{2})^t \leq (e^{-\frac{\varepsilon}{2}})^t = e^{-\frac{\varepsilon}{2}t} \leq e^{-\frac{\varepsilon}{2}\frac{2}{\varepsilon}\ln\frac{1}{\delta} = e^{-\ln\frac{1}{\delta}}} = \delta$$

2.2 Vertex Cover problem

Input: undirected graph $G(V, E)$.

Output: $S \subseteq V$, of smallest cardinalityt, s.t. $\forall e \in E \; e \cap S \neq \emptyset$.

In other words, we want a subset of the vertices s.t. *every* edge of the graph touches at least one of the chosen vertices.

It is a *minimization* problem; note in fact that we could take every vertex (i.e. $S = V$) and this would clearly cover every edge.

It is an NP-Complete problem.

Algorithm 3 Approximation to VC

```
procedure MaximalMatching($V, E$)
    $S \leftarrow \emptyset$
    while $E \neq \emptyset$ do
        pick $e = \{u, v\} \in E$
        $S \leftarrow S \cup e$
        remove all the edges incident on $v$ and $w$ from $E$
    end while
    return $S$
end procedure
```

The way we solve this problem is by actually solving another problem; in fact, we find a *maximal matching*, that happens to be a VC.

Definition 2.1 (Matching)**.** A matching of the graph $G(V, E)$ is a subset $A \subseteq E$ s.t. $\forall \{e, e'\} \in \binom{A}{2}$, $e \cap e' = \emptyset$.

Definition 2.2 (Maximal matching)**.** A maximal matching of $G(V, E)$ is a subset $A \subseteq E$ s.t.

- it is a matching
- $\forall e \in E \setminus A$, $A \cup \{e\}$ is not a matching

Note that there is a distinction between *maximal* and *maximum* matching. A maximum matching is a matching with the largest possible cardinality. A maximal matching is a matching s.t. we can't add any more edges to it; so if we do so, the set is not a matching anymore.

So it could happen that a matching is maximal, but there exists another matching with larger cardinality.

Lemma 2.4. If $e_1, \dots, e_t$ are the edges selected by the MaximalMatching algorithms, then $\{e_1, \dots, e_t\}$ is a maximal matching.

Proof. For brevity, let us call $A = \{e_1, \dots, e_t\}$. Suppose by contradiction that A is not a maximal matching.

So, either A is not a matching at all, or is not maximal, so there exists an edge removed by the algorithm that should have been added to the set.

When the algorithms picks an edge e, it removes all the edges connected to both endpoints of e. This ensures that $\forall e_1, e_2 \in A$, $e_1 \cap e_2 = \emptyset$. So we know that the set is indeed a matching, and now we have to check if it is maximal.

If it's not maximal, then $\exists e \in E \setminus A$ that has been wrongfully removed by the algorithm. But, following the same reasoning used to show that A is a matching, we know that e shares some endpoints with at least one of the selected edges; i.e. $\exists e_A \in A$ s.t. $e \cap e_A \neq \emptyset$.

So, adding any of the edges removed by the algorithm, would "break" the matching. Thus A is also maximal. $\square$

Lemma 2.5. If MaximalMatching selects t edges, then $|S| = 2t$.

Proof. Trivial. $\square$

Lemma 2.6. Let A be any matching of $G(V, E)$. If S is a VC of $G(V, E)$, then $|S| \geq |A|$.

Proof. First note that a VC for $G(V, E)$ is also a VC for $G(V, B)$, $\forall B \subseteq E$.

Now, pick any $A \subseteq E$. Since, by assumption, S is a VC for $G(V, E)$, then S must also cover each edge in A; this S is a VC for $G(V, A)$.

The graph $G(V, A)$ can only have nodes of degree at most 1 (i.e. either 0 or 1). In fact, suppose A has a node of degree > 1, say 2; then A would have two edges that share a node, thus it wouldn't be a matching.

So, from the point of view of VC, any node in $G(V, A)$ can cover at most one edge, since $\forall v \in V$, $\deg_{G(V,A)}(v) \leq 1$.

Thus, a set of, say, k nodes, can cover $\leq k$ edges of $G(V, A)$. Now, $G(V, A)$ has $|A|$ edges, thus each VC of $G(V, A)$ has to have at least $|A|$ nodes. $\square$

Theorem 2.7. MaximalMatching returns a 2-approximation to VC.

Proof. Let A be the maximal matching produced by the MaximalMatching procedure (the one called $\{e_1, \dots, e_t\}$ in Lemma 2.4).

This solution contains $|S| = 2|A|$ nodes (Lemma 2.5).

Since A is a matching, if S^* is an optimal solution (smallest solution), we know by Lemma 2.6 that $|S^*| \geq |A|$.

$|S^*| \geq |A| = \frac{|S|}{2} \rightarrow |S| \leq 2|S^*|$. $\square$

There are reasons to believe that this is the best possible approximation. In fact, VC is *conjectured* to be NP-Hard to approximate to $2 - \varepsilon$ ($\forall \varepsilon > 0$ constant).

Although, good news, VC has a good property that lets us "easily" (under some conditions) find a solution. It is **fixed parameter tractable**. If $G(V, E)$ has a VC of k nodes, then an optimal VC of $G(V, E)$ can be found in time $n^c \cdot f(k)$; specifically, there's an algorithm that finds a VC in time $O(n^2 \cdot 2^k)$.

So, if the optimal solution is small enough (i.e. $\log n$), then this search only takes polynomial time.

The algorithm makes use of Induced subgraph.

Algorithm 4 Search of a VC

```
procedure VC($G(V, E), k$)
    if $E = \emptyset$ then
        return true
    end if
    if $k = 0$ then
        return false
    else
        fix $\{u, v\} \in E$
        if VC($G[V \setminus \{u\}], k - 1$) then
            return true
        end if
        if VC($G[V \setminus \{v\}], k - 1$) then
            return true
        end if
    end if
    return false
end procedure
```

Lemma 2.8. $\mathrm{VC}(G(V, E), k)$ takes time $O(n^2 \cdot 2^k)$.

Proof. By induction, running $\mathrm{VC}(G(V, E), l)$ causes at most $2^l - 1$ calls to the function VC.

Base case ($l = 0$). Only the first call to $\mathrm{VC}(\cdot, 0)$ will be generated. $2^{l-1} - 1 = 2^1 - 1 = 1$.

Inductive step ($l + 1$). Assume the claim holds for l. The number of calls generated by $\mathrm{VC}(\cdot, l + 1)$ is equal to no more than twice the numner of calls generated by $\mathrm{VC}(\cdot, l)$+1. The total number of calls, by induction, is then $\leq 1 + 2(2^{l+1} - 1) = 1 + 2^{l+2} - 2 = 2^{l+2} - 1$.

Each single casll takes time $O(n^2)$; there a few checkes that can be done in constant time, we have to fix an edge, and we have to build the induced subgraph (which take $O(n^2)$). Note that, fixing an edge might take $O(n^2)$; worst case we use an adjacency matrix and we visit it until we find a fixable edge.

Thus, the claim follows. □

Lemma 2.9. $\mathrm{VC}(G(V, E), k) ==$ true iff $G(V, E)$ has a VC of size k.

Exercise 2.9.1. The proof of above lemma is left as an exercise

3 Mathematical programming

We can divide mathematical programs in:

- Linear Programming (LP)
- Semi-Definite Programming (SDP)

A mathematical program contains:

- varibles $x_1, \dots, x_n$ ($x_i \in \mathbb{R}$ or $x_i \in \mathbb{Q}$)
- objective function $f(x_1, \dots, x_n)$ (either to minimize or to maximize)
- constraint functions $g_i(x_1, \dots, x_n) \leq b_i$ (or $g_i(x_1, \dots, x_n) \geq b_i$), for $i = 1, \dots, n$

In a linear program, the objective function f is linear and each constraint function g_i is linear as well.

When we force each variable to be an integer, the progam is called **Integer Program**.

Definition 3.1 (Feasible solution)**.** A solution is said to be feasible if it satisfied all constraints.

3.1 Some examples

$$\begin{cases} \max x_1 + x_2 \\ x_1 \geq 3 \\ x_2 \geq 5 \\ x_1 \leq 2.5 \end{cases} \tag{1}$$

This program clearly has no solutions, since the constraints $x_1 \geq 3$ and $x_1 \leq 2.5$ can't be both satisfies ad the same time.

$$\begin{cases} \max x_1 + x_2 \\ x_2 \geq 5 \\ x_1 \leq 2.5 \end{cases} \tag{2}$$

There are infinitely many solutions, and the "optimal value" of this LP is unbounded (∞).

$$\begin{cases} \max x_1 + x_2 \\ 2x_1 + x_2 \leq 10 \\ x_1 \geq 0 x_2 \geq 0 \end{cases} \tag{3}$$

We can set $x_1 = 0$ and $x_2 = 10$. All the constraints are satisfied and our objective function would have value 10.

3.2 Hardness of LPs

If $f(x_1, \dots, x_n) = \sum_{j=1}^{n} c_j x_j$ and $g_i(x_1, \dots, x_n) = \sum_{j=1}^{n} a_{ij} x_j$, then if each coefficient c_j and a_{ij} and each term b_i is a rational number, representable with at most t bits, then the LP can be optimized in time $O(n \cdot m \cdot t)^c$, for come constant $c > 0$.

IPs are instead harder to solve than LPs. In general, we can't solve an IP in polynomial time. What can be done, and will be done later for some problems, is to give an IP program for a problem, that will be tranformed into an LP, easier to solve.

3.3 Solution for Vertex Cover

We now see a pratical use of LPs to solve the Vertex Cover problem.

Let our set of variables be $\{x_v \mid v \in V\}$. We define our IP, for a graph $G(V, E)$ to be the following:

$$\begin{cases} \min \sum_{v \in V} x_v \\ x_u + x_v \geq 1 & \forall \{u, v\} \in E \\ x_v \in \{0, 1\} & \forall v \in V \end{cases} \tag{4}$$

Solving this IP would give us an optimal VC.

Let $x^*_{v_1}, x^*_{v_2}, \dots, x^*_{v_n}$ be an optimal solution to the IP. Define $S^* = \{v \mid v \in V \wedge x^*_v = 1\}$.

Theorem 3.1. *S^* is an optimal VC of $G(V, E)$.*

Lemma 3.2. *S^* is a vertex cover.*

Proof. S^* being a VC means that $\forall \{u, v\} \in E$, $u \in S^*$ or $v \in S^*$ or both (i.e. $u, v \in S^*$). Given that $x^*_u + x^*_v \geq 1$, $\forall \{u, v\} \in E$, and that $x^*_u, x^*_v \in \{0, 1\}$, at least one of x^*_u and x^*_v is equal to 1. Thus $u \in S^*$ or $v \in S^*$. $\square$

Lemma 3.3. *If S is a VC, then there exists a feasible solution $\{x_v\}_{v \in V}$ to the IP, s.t. $\sum_{v \in V} x_v = |S|$.*

Proof. $x_v = 1 \leftrightarrow v \in S$. $\square$

Corollary 3.3.1. *An optimal solution to the IP has value equal to*

$$\min_{\substack{S \subseteq V \\ S \text{ is a VC}}} |S|$$

Recall that IPs can't be solve, in general, in polytime, while LPs can. So we *relax* the IP for VC (see equation 4) into an LP.

$$\begin{cases} \min \sum_{v \in V} x_v \\ x_u + x_v \geq 1 & \forall \{u, v\} \in E \\ 0 \leq x_v \leq 1 & \forall v \in V \end{cases} \tag{5}$$

Note the the two programs are almost the same, except that, during the relaxation, we've allowed each variable from being in $\{0,1\}$, to being in the range $[0,1]$. Albeit this is not how IPs are always transformed into LPs, it is indeed quite often the case, and it is the way in which most (if not all) of the IPs during this course will be transformed into LPs.

It is easy transforming the IP into a VC solution: for a vertex v, $x_v = 1$ iff v is in the vertex cover. In the LP we have fractional values, so how do we choose vertices? We apply a **rounding rule**: an algorithm that, given an LP solution, outputs a VC set.

The rounding rule for this problem is quite simple. We define

$$S = \{v \mid x_v \geq \frac{1}{2} \wedge v \in V\}$$

Lemma 3.4. *S is a vertex cover.*

Proof. Let $\{u,v\} \in E$. The LP contains the constraint $x_u + x_v \geq 1$. Given that our solution guarantees that $x_u + x_v \geq 1$ (the LP solution is feasible by definition), $x_u + x_v \geq 1 \rightarrow \max(x_u, x_v) \geq \frac{x_u+x_v}{2} \geq \frac{1}{2}$.

Thus, $u \in S$, or $v \in S$ (or both), thus the edge $\{u,v\}$ is covered. □

Lemma 3.5. *$|S| \leq 2\sum_{v\in V} x_v$.*

Proof. $|S| = \sum_{v\in S} 1 \overset{(v\in S \rightarrow x_v \geq \frac{1}{2})}{\leq} \sum_{v\in S} 2x_v = 2\sum_{v\in S} x_v \overset{(x_v \geq 0)}{\leq} 2\sum_{v\in V} x_v$ □

Recall that the LP minimizes $\sum_{v\in V} x_v$. Let $\{x_v^*\}_{v\in V}$ be an optimal solution to the LP. Let S^* be the result of the application of our rounding rule; i.e. $S^* = \{v \mid v \in V \wedge x_v^* \geq \frac{1}{2}\}$.

$$|S^*| \overset{(\text{Lemma } 3.5)}{\leq} 2\sum_{v\in V} x_v^* = 2LP^* \overset{(\text{LP relaxation of IP})}{\leq} 2IP^* = 2 \min_{\substack{S\subseteq V \\ S \text{ is a VC}}} |S| = 2OPT$$

If Lemma 3.5 could be improved (to, say, $|S| \leq 1.9\sum_{v\in V} x_v$), then the approximation ration would be directly improved.

Let us define the **integrality gap** to be the ratio between the optimal IP solution and the optimal LP solution. That is,

$$IG = \frac{IP^*}{LP^*}$$

Now, let us consider a particular graph s.t. we can use it show an upper bound on the integrality gap. This approach will be used also in the future: build a special instance of a problem, s.t. the instance can be used to show lower or upper bound to the optimality of approximations.

Consider the graph $G(V,E) = K_3$ (i.e. the complete graph with 3 vertices).

Lemma 3.6. Each optimal solution for $G(V,E)$ contains two nodes.

Lemma 3.7. $x_1 = x_2 = x_3 = \frac{1}{2}$ is a feasible solution of value $\frac{3}{2}$.

Proof. $\forall \{i,j\} \in E$, $x_i + x_j = 2\frac{1}{2} = 1 \geq 1$. Thus each constraint is satisfied.

The objective value is $3\frac{1}{2} = \frac{3}{2}$. □

Lemma 3.8. There exists no feasible LP solution of value $< \frac{3}{2}$.

Proof. For $\{i,j\} \in E$, $x_i + x_j \geq 1$.

$2\sum_{v \in V} x_v = 2x_1 + 2x_2 + 2x_3 = (x_1 + x_2) + (x_2 + x_3) + (x_1 + x_3) \overset{\text{feasibility}}{\geq} 1 + 1 + 1 = 3$.

Thus $\sum_{v \in V} \geq \frac{3}{2}$. □

Corollary 3.8.1. $LP^* = \frac{3}{2}$ and $IP^* = 2$.

Thus, for K_3, $IG = \frac{IP^*}{LP^*} = \frac{2}{\frac{3}{2}} = \frac{4}{3}$. So $IG \leq 2$.

Exercise 3.8.1. Show that $\forall \varepsilon > 0$, $\exists G(V,E)$ s.t. $\dfrac{OPT_{VC}(G(V,E))}{LP^*(G(V,E))} \geq 2 - \varepsilon$.

Hints:

- K_t, large t
- Prove the optimal VC for K_t has $t-1$ nodes
- What is the optimal LP solution for K_t?

3.4 Set Cover problem

Input: $U = [n] = \{1, \ldots, n\}$, $S_1, \ldots, S_m \subseteq U$.

We call the input U, meaning our *universe* set.

Question: what is the size of a smallest class of subsets $T \subseteq [m]$ s.t. $\bigcup_{j \in T} S_j = U$?

Let us see an example. Let $U = [4] = \{1,2,3,4\}$, and let $S_1 = \{1,2\}$, $S_2 = \{2,3\}$, $S_3 = \{3,4\}$. A solution, even optimal, is $S_1 \cup S_3 = [4]$.

Set Cover (SC) is NP-Hard. We consider an LP solution to approximate it. We first see an IP problem, translated into an LP.

Let the IP be

$$\begin{cases} \min \sum_{j=1}^{m} x_j & (x_j = 1 \text{ if } S_j \text{ is in the solution}) \\ \sum_{j \in [m] \text{ s.t. } i \in S_j} x_j \geq 1 & \forall i \in [n] \\ x_j \in \{0,1\} & \forall j \in [m] \end{cases} \tag{6}$$

The LP relaxation is

$$\begin{cases} \min \sum_{j=1}^{m} x_j \\ \sum_{j \in [m] \text{ s.t. } i \in S_j} x_j \geq 1 & \forall i \in [n] \\ 0 \leq x_j \leq 1 & \forall j \in [m] \end{cases} \tag{7}$$

The constraint $\sum_{j\in[m] \text{ s.t. } i\in S_j} x_j \geq 1$ means that each element of $[n]$ has to be picked at least once.

Observation 3.8.1. *$LP^* \leq IP^*$*

Proof. The LP is a relaxation to the IP (the constraint of the LP are less stringent than those of the IP). Thus each IP solution is also an LP solution. □

We now need a rounding rule, to converto the LP solution into a solution for Set Cover. This rounding rule is a randomized procedure, less trivial than the one used for VC. Many rouding rules are actually randomized, like this one.

Algorithm 5 Randomized rounding technique

```
  $A \leftarrow \emptyset$
  let $x^*$ be an optimal LP solution
  for $k = 1$ to $\lceil 2\ln n \rceil$ do
      for $j = 1$ to $m$ do
          flip an independent coin with head probability $x_j^*$
          if the coin is heads then
              $A \leftarrow A \cup \{S_j\}$
          end if
      end for
  end for
```

The first for loop is a repetition of "phases". It represents how many times do we want to repeat the subprocedure in the second loop, the one that loops over sets and adds them to the solution.

As it is going to be clearer later on, since this procedure is randomized, it is not always the case that the output A is indeed a set cover.

The number of times we repeat the first loop actually defines a sort of trade off. The more times we repeat it, the higher the probability that A is a set cover, but also the higher the probability that we put too many sets in the solution, leading to a solution worse than the optimal.

Lemma 3.9. *Consider a generic iteration k of the outer loop. Let p_i be the probability in iteration k that at least on of the sets containing i gets added to A. Then $p_i \geq 1 - \frac{1}{e} \approx 0.633\ldots$.*

Proof. To prove this lemma we are going to use that fact that $1 - x \leq e^{-x}$.

$Pr[i \text{ not covered in iteration } k] = \prod_{j\in[m]:i\in S_j}(1 - x_j^*) \leq \prod_{j\in[m]:i\in S_j} e^{-x_j^*} = e^{-\sum_{j\in[m]:i\in S_j}} \overset{\text{(feasibility of solution)}}{\leq} e^{-1}$.

Thus $Pr[i \text{ covered in iteration } k] \geq 1 - \frac{1}{e}$. □

Lemma 3.10. *Pick $i \in [n]$. $Pr[i \text{ not covered by ant set in } A] \leq \frac{1}{n^2}$.*

Proof. $Pr[i \text{ not covered by any set in } A] = \prod_{k=1}^{\lceil 2\ln n\rceil} Pr[i \text{ not covered in iteration } k] \leq \prod_{k=1}^{\lceil 2\ln n\rceil} \frac{1}{e} = e^{-\lceil 2\ln n\rceil} = n^{-2}$ □

Lemma 3.11. $Pr[A \text{ is a set cover}] \geq 1 - \frac{1}{n}$

Proof. $Pr[A \text{ is not a set cover}] \overset{\text{(UB)}}{\leq} \sum_{i=1}^{n} Pr[i \text{ not covered by } A] \overset{\text{(Lemma 3.10)}}{\leq} \sum_{i=1}^{n} \frac{1}{n^2} = \frac{n}{n^2} = \frac{1}{n}$ □

Lemma 3.12. $\mathbb{E}[|A|] \leq \lceil 2 \ln n \rceil LP^* \leq \lceil 2 \ln n \rceil OPT_{SC}$

Proof. Fix an iteration k of the outer loop. Let A_k be the class of sets added to A in iteration k.

$\mathbb{E}[|A_k|] \overset{\text{(linearity of } \mathbb{E})}{=} \sum_{i=1}^{m} x_j^* = LP^* \leq OPT_{SC}$

$A = A_1 \cup A_2 \cup \dots \cup A_k \cup \dots \cup A_{\lceil 2 \ln n \rceil}$.

$|A| \leq \sum_{i=1}^{\lceil 2 \ln n \rceil} |A_i|$. Then $\mathbb{E}[|A|] \leq \sum_{i=1}^{\lceil 2 \ln n \rceil} \mathbb{E}[|A|] = \sum_{i=1}^{\lceil 2 \ln n \rceil} LP^* \leq \lceil 2 \ln n \rceil LP^* \leq \lceil 2 \ln n \rceil OPT_{SC}$. □

So, let us consider the following statement

$$? \overset{\exists \text{ instance}}{\leq} \frac{OPT_{SC}}{LP^*_{SC}} \overset{\forall \text{ instance}}{\leq} \lceil 2 \ln n \rceil$$

If we manage to find an instance with an IG smaller than $\frac{OPT_{SC}}{LP^*_{SC}}$, then we have proved a lower bound on the integrality gap of set cover.

We know of such an instance, that we shall see in a moment. As it is the case, the following will be proved

$$\frac{1}{4 \ln 2} \ln n \overset{\exists \text{ instance}}{\leq} \frac{OPT_{SC}}{LP^*_{SC}} \overset{\forall \text{ instance}}{\leq} \lceil 2 \ln n \rceil$$

An improvement, albeit a small one.

Let us consider the following universe set, $E = \{e_A \mid A \in \binom{[q]}{\frac{q}{2}},\ \text{for some even } q \geq 2\}$.

For $n = |E| = \binom{q}{\frac{q}{2}} = \Theta(\frac{2^q}{\sqrt{q}}) \Rightarrow q \geq \log n - \Theta(\log \log n)$.

$\forall i \in [q]$, let us define $S_i = \{e_A \mid e_A \in E \wedge i \in A\}$ and $C = \{S_i \mid i \in [q]\}$. $|C| = q \overset{\Delta}{=} m$.

Let us see an example. Given $q = 4$:

- $E = \{e_{12}, e_{13}, e_{14}, e_{23}, e_{24}, e_{34}\}$
- $S_1 = \{e_{12}, e_{13}, e_{14}\}$
- $S_2 = \{e_{12}, e_{23}, e_{24}\}$
- $S_3 = \{e_{13}, e_{23}, e_{34}\}$
- $S_4 = \{e_{14}, e_{24}, e_{34}\}$

Lemma 3.13. $LP^*_{SC} \leq 2$

Proof. Consider the LP solution $_1 = x_2 = \cdots = x_q = \frac{2}{q}$, with $q = n$. This solution has value $\sum_{i=1}^{q} x_i = q\frac{2}{q} = 2$.

The generic LP constraint is $\sum_{s_j : e_a \in S_j} x_j \geq 1$, $\forall e_a \in E$.

$\sum_{s_j : e_a \in S_j} x_j = \sum_{s_j : e_a \in S_j} \frac{2}{q} = |A|\frac{2}{q} = \frac{q}{2}\frac{2}{q} = 1 \geq 1$. Thus the solution is feasible. □

Lemma 3.14. $OPT_{SC} \geq \frac{1}{2}\log_2 n - O(\log\log n)$

Proof. Assume by contradiction that the sets $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$, for $k \leq \frac{q}{2}$, cover each element of the instance. Consider also the set $T = [q] \setminus \{i_1, \ldots, i_k\}$. Then $\exists A \subseteq T$ s.t. $|A| = \frac{q}{2}$ (because $|T| \geq q - k \geq \frac{q}{2}$), then $e_A \in E$.

The element e_A is not covered by $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$, because $A \cap \{i_1, \ldots, i_k\} = \emptyset$. Then $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$ is not a set cover if $k \leq \frac{q}{2}$.

It follows that the minimum set cover contains $\geq \frac{q}{2} + 1$ sets. □

Thus $IG = \dfrac{OPT_{SC}}{LP^*_{SC}} \overset{\text{(on some instances)}}{\geq} \dfrac{\frac{1}{2}\log_2 n - O(\log\log n)}{2} \approx \frac{1}{4}\log_2 n$

3.5 Densest subgraph problem

Input: $G(V, E)$

Question: what is the subset $S \subseteq V$ having maximum **density**?

We define density as follows:

$$\rho(S) = \frac{|E(S)|}{|S|}$$

Where $E(S) = \{\{u, v\} \mid \{u, v\} \in E \wedge u, v \in S\}$.

A variation of the problem is the following. For k-densest subgraph we want $S \subseteq V$, $|S| = k$, s.t. $|E(S)|$ is as large as possible.

To solve this problem, we give an LP program.

$$\begin{cases} \max \sum_{\{i,j\} \in E} x_{\{i,j\}} & \\ x_{\{i,j\}} \leq y_i & \forall \{i,j\} \in E \\ x_{\{i,j\}} \leq y_j & \forall \{i,j\} \in E \\ \sum_{i \in V} y_i \leq 1 & \\ x_{\{i,j\}} \geq 0 & \forall \{i,j\} \in E \\ y_i \geq 0 & \forall i \in V \end{cases} \tag{8}$$

Lemma 3.15. For any $G(V, E)$, $\forall S \subseteq V$, $\exists$ feasible solution to the LP, having value $\geq \frac{|E(S)|}{|S|} = \rho(S)$.

Proof. For $i \in S$, set $y_i = \frac{1}{|S|}$. For $i \in V \setminus S$, set $y_i = 0$. For each $\{i,j\} \in E$, set

$$x_{\{i,j\}} = \begin{cases} \frac{1}{|S|} & \text{if } \{i,j\} \in E(S) \\ 0 & \text{o/w} \end{cases} \tag{9}$$

The constraint $\sum_{i \in V} y_i \leq 1$ is satisfied, given that $\sum_{i \in V} y_i = \sum_{i \in S} y_i + \sum_{i \in V \setminus S} y_i = \sum_{i \in S} \frac{1}{|S|} + \sum_{i \in V \setminus S} 0 = \frac{|S|}{|S|} = 1$.

Take any $\{i,j\} \in E(S)$. We have $x_{\{i,j\}} = \frac{1}{|S|}$. But, if $\{i,j\} \in E(S)$, then $i,j \in S$. Thus $y_i = y_j = \frac{1}{|S|}$. Then, $\forall \{i,j\} \in E(S)$, the constraints $x_{\{i,j\}} \leq y_i$ and $x_{\{i,j\}} \leq y_j$ are both satisfied.

If instead $\{i,j\} \notin E(S)$, then $x_{\{i,j\}} = 0$. Thus $x_{\{i,j\}} \leq y_i$ and $x_{\{i,j\}} \leq y_j$ are both satisfied by $y_i, y_j \geq 0$.

Thus, our solution is feasible.

The value of the solution is:

$$\sum_{\{i,j\} \in E} x_{\{i,j\}} = \sum_{\{i,j\} \in E(S)} x_{\{i,j\}} = \sum_{\{i,j\} \in E(S)} \frac{1}{|S|} = \frac{|E(S)|}{|S|} = \rho(S)$$

□

Lemma 3.16. For any feasible LP solution of value v, $\exists S \subseteq V$ s.t. $\rho(S) \geq v$.

Proof. Let Y', X' be a feasible LP solution of value v. Let Y, X be the solution to the LP s.t.

- $y_i = y'_i$, $\forall i \in V$
- $x_{\{i,j\}} = \min(y_i, y_j)$, $\forall \{i,j\} \in E$

The Y, X solution is feasible, indeed:

- $\sum_{i \in V} y_i = \sum_{i \in V} y'_i \leq 1$
- $\forall \{i,j\} \in E$, it holds $x_{\{i,j\}} \leq y_i$ and $x_{\{i,j\}} \leq y_j$

We have to provide a set S having density at least v. We are going to provide a number of S's, one of which will have the desired property. $\forall r \geq 0$, let us consider

$$S(r) = \{i \mid y_i \geq r\}$$

$$E(r) = \{\{i,j\} \mid x_{\{i,j\}} \geq r\}$$

We can observe that $\{i,j\} \in E(r) \leftrightarrow i \in S(r)$ and $j \in S(r)$. The proof is left as an exercise.

We claim that $\exists r \geq 0$ s.t. $\rho(S(r)) = \dfrac{|E(r)|}{|S(r)|} \geq \sum_{\{i,j\} \in E} x_{\{i,j\}} = v$.

Proof of claim. We define a permutation π of the nodes s.t.

$$0 \leq y_{\pi(1)} \leq y_{\pi(2)} \leq \cdots \leq y_{\pi(n-1)} \leq y_{\pi(n)} \leq 1$$

$$\begin{aligned}
&\int_0^1 |E(r)|dr = \\
&\quad = \int_0^{y_{\pi(1)}} |S(r)|dr + \int_{y_{\pi(1)}}^{y_{\pi(2)}} |S(r)|dr + \dots + \int_{y_{\pi(n)}}^{1} |S(r)|dr = \\
&\quad = \int_0^{y_{\pi(1)}} n\, dr + \int_{y_{\pi(1)}}^{y_{\pi(2)}} (n-1)dr + \dots + \int_{y_{\pi(n-1)}}^{y_{\pi(n)}} 1dr + \int_{y_{\pi(n)}}^{1} 0dr = \\
&\quad = (y_{\pi(1)} - 0)n + (y_{\pi(2)} - y_{\pi(1)})(n-1) + \dots + (y_{\pi(n)} - y_{\pi(n-1)})1 + 0 = \\
&\quad = y_{\pi(1)}(n - (n-1)) + y_{\pi(2)}((n-1) - (n-2)) + \dots + y_{\pi(n)}(1-0) = \\
&\quad = \sum_{i=1}^{n} y_{\pi(i)} = \sum_{i=1}^{n} y_i \le 1
\end{aligned}$$

$\int_0^1 |E(r)|dr = \sum_{\{i,j\}\in E} x_{\{i,j\}} = \sum_{\{i,j\}\in E} \min(y_i, y_j) = \sum_{\{i,j\}\in E} \min(y'_i, y'_j) \geq \sum_{\{i,j\}\in E} x'_{\{i,j\}} = v$

Then, $\int_0^1 |S(r)|dr \le 1$ and $\int_0^1 |E(r)|dr \ge v$.

By contradiction, supopose that $\forall r \geq 0$, $\dfrac{|E(r)|}{|S(r)|} < v$. Then $|E(r)| < v|S(r)|$, $\forall r \ge 0$. Thus $v \le \sum_{\{i,j\}\in E} x_{\{i,j\}} = \int_0^1 |E(r)|dr < \int_0^1 v|S(r)|dr = v \int_0^1 |S(r)|dr \le v \cdot 1 = v$.

Since $v \not< v$, we have a contradiction. Thus $\exists r \ge 0$ s.t. $|E(r)| \ge v|S(r)| \to \rho(S(r)) \ge v$. □

Thus, we can get an optimal (densest) subgraph (one of $S(y_1), \dots, S(y_n)$ will do). We can solve the LP in polytime, and we can transform the LP solution in an optimal subgraph. Is this fast enough? No, the LP has $\ge |E|$ variables.

Corollary 3.16.1. $LP^* = OPT_{DS}$

Proof. By Lemma 3.15, $LP^* \ge OPT_{DS}$. By Lemma 3.16, $LP^* \le OPT_{DS}$. □

The LP for Densest Subgraph has $\Theta(n^2)$ variables, so albeit being polynomially solvable, the number of variables is not linear. For n very large (e.g. $10^8 \dots 10^9$ nodes), n^2 is too slow.

We now see a greedy algorithm, due to M. Charikar, to approximate Densest Subgraph.

Exercise 3.16.1. Prove that Greedy does not always return an optimal solution.

Hint: construct a graph s.t. the algorithm returns a non optimal solution.

Definition 3.2 (Orientation)**.** An orientation ϕ of the edges of an undirected graph $G(V, E)$ is a function that assigns to each $e \in E$ one of its endpoints

$$\phi : E \to V$$

Algorithm 6 Charikar algorithm

```
procedure GREEDY($G(V,E)$)
    $S \leftarrow V$
    for $i = 1, 2, \dots, n$ do
        let $v_i$ be a node having minimum degree in $G[S_{i-1}]$
        $S_i \leftarrow S_{i-1} \setminus \{v_i\}$
    end for
    return a set $S_i$ having larget density $\rho(S_i)$
end procedure
```

Note that $\phi(e) \in e$.

Definition 3.3. Given an orientation ϕ of $G(V,E)$, let $d_\phi(v)$ be the in-degree of $v \in V$, according to ϕ.

Observation 3.16.1. If ϕ is an orientation of $G(V,E)$, then $\sum_{v \in V} d_\phi(v) = |E|$.

Definition 3.4. The max-degree of ϕ is $\Delta_\phi = \max_{v \in V} d_\phi(v)$.

Lemma 3.17. $\max_{\emptyset \subset S \subseteq V} \rho(S) = \max_{\emptyset \subset S \subseteq V} \dfrac{|E(S)|}{|S|} \leq \Delta_\phi, \forall \phi$.

Proof. Let $\emptyset \subset S \subseteq V$ be given. Each $\{u,v\} \in E(S)$ is going to be oriented towards on of its endpoints; that is, towards a node of S. So $|E(S)| \leq \sum_{v \in S} d_\phi(v) \leq \sum_{v \in S} \Delta_\phi = |S|\Delta\phi$.

Thus, $\dfrac{|E(S)|}{|S|} \leq \Delta_\phi$. $\square$

While running the greedy algorithms, we *build* an orientation. Whenever node v_i (the i^{th} node to be removed) is removed from the graph (which, at that point, has node set S_{i-1}) we orient each edge of $G[S_{i-1}]$ that is incident on v_i towards v_i. The resulting orientation is called ϕ_{GR}.

Observe that, with orientation ϕ_{GR}, $d_{\phi_{GR}}(v)$ equals the degree of v in the graph, when v is about to be removed.

Lemma 3.18. Let M be the maximum $\rho(S_i)$ value; i.e. $M = \max_{i=0,\dots,n-1} \rho(S_i)$. (Note that M is the value of the greedy solution).

Then, $\Delta_{\phi_{GR}} \leq 2M$.

Proof. For $i = 1, 2, \dots, n$, v_i is a node of minumum degree in $G[S_{i-1}]$. In general, the minumum degree in a graph is not larger than the average degree of the same graph. Thus

$$d_{\phi_{GR}}(v_i) = \min_{v \in S_{i-1}} d_{S_{i-1}}(v) \leq \operatorname*{avg}_{v \in S_{i-1}} d_{S_{i-1}}(v) =$$
$$= \frac{\sum_{v \in S_{i-1}} d_{S_{i-1}}(v)}{|S_{i-1}|} = \frac{2|E(S_{i-1})|}{|S_{i-1}|} = 2\rho(S_{i-1})$$

Then

$$\Delta_{\phi_{GR}} = \max_{v_i \in V} d_{\phi_{GR}}(v_i) = \max_{i=1,\ldots,n} d_{\phi_{GR}}(v_i) \leq \max_{i=1,\ldots,n} 2\rho(S_{i-1}) = 2M$$

□

Corollary 3.18.1. Greedy returns a 2-approximation to Densest Subgraph.

Proof.

$$M \geq \frac{\Delta_{\phi_{GR}}}{2} \geq \frac{\max_{\emptyset \subset S \subseteq V} \rho(S)}{2}$$

Thus the greedy solution is not worse than a 2-approximation. □

3.6 Primal and Dual

We can define an LP, in general, as a **primal** in the following way:

$$\begin{cases} \max c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ (y_1)a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\ (y_2)a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\ \vdots \\ (y_2)a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \\ x_1, \ldots, x_n \geq 0 \end{cases} \tag{10}$$

The **dual** of the above LP is:

$$\begin{cases} \min b_1y_1 + b_2y_2 + \cdots + b_my_m \\ (x_1)a_{11}y_1 + a_{21}y_2 + \cdots + a_{m1}y_m \geq c_1 \\ (x_2)a_{12}y_1 + a_{22}y_2 + \cdots + a_{m2}y_m \geq c_2 \\ \vdots \\ (x_n)a_{1n}y_1 + a_{2n}y_2 + \cdots + a_{mn}y_m \geq c_n \\ y_1, \ldots, y_m \geq 0 \end{cases} \tag{11}$$

The fact that a primal is always defined as a maximization problem, and a dual as a minimization problem, is conventional.

Now, we can also define these programs in matrix form.

If

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \tag{12}$$

$$b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} \tag{13}$$

$$c = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \tag{14}$$

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \tag{15}$$

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} \tag{16}$$

Then the **primal** is

$$\begin{cases} \max c^T X \\ AX \leq b \\ X \geq 0 \end{cases} \tag{17}$$

And the **dual** is

$$\begin{cases} \min b^T Y \\ A^T Y \geq c \\ Y \geq 0 \end{cases} \tag{18}$$

Recall that, if N and M are matrices, then $(NM)^T = M^T N^T$.

Theorem 3.19 (Weak Duality)**.** If

- X is a feasible solution to the primal
- Y is a feasible solution to the dual

Then

$$Primal(C) = c^T X \leq b^T Y = Dual(Y)$$

Proof.

$$c^T X = X^T c \leq X^T A^T Y = (AX)^T Y \leq b^T Y$$

□

Recall Densest subgraph problem LP formulation. Let us see it as a Primal LP (in "standard form").

$$\begin{cases} \max \sum_{\{i,j\} \in E} x_{\{i,j\}} & \\ (y_{ij}) : x_{\{i,j\}} - x_i \leq 0 & \forall \{i,j\} \in E \\ (y_{ji}) : x : \{i,j\} - x_j \leq 0 & \forall \{i,j\} \in E \\ (y^*) : \sum_{i \in V} x_i \leq 1 & \\ x_{\{i,j\}} \geq 0 & \forall \{i,j\} \in E \\ x_i \geq 0 & \forall i \in V \end{cases} \tag{19}$$

Let us now consider the equivalent Dual LP.

$$\begin{cases} \min y^* \\ (x_{\{i,j\}}) : y_{ij} + y_{ji} \geq 1 \\ (x_i) : y^* - \sum_{j:\{i,j\}\in E} y_{ij} \geq 0 \\ y^* \geq 0 \\ y_{ij}, y_{ji} \geq 0 \end{cases} \tag{20}$$

Our greedy 2-approximtion for DS is esentially a dual LP solution: set $y^* = \Delta_{\phi_{GR}}$ and

$$y_{ij} = \begin{cases} 1 & \text{if } \{i,j\} \text{ is directed towards } i \text{ in } \phi_{GR} \\ 0 & \text{o/w} \end{cases} \tag{21}$$

For the $x_{\{i,j\}}$, $y_{ij} + y_{ji} \geq 1$ is satisfied, because it is either $1+0 \geq 1$ or $0+1 \geq 1$. For x_i, $y^* \geq \sum_{j:\{i,j\}\in E} y_{ij}$. $\Delta_{\phi_{GR}} = y^* \geq \sum_{j:\{i,j\}\in E} y_{ij} = deg_{\phi_{GR}}(i)$.

Then, each constraint is satisfied by our solution, of value $\Delta_{\phi_{GR}}$.

$$\frac{\Delta_{\phi_{GR}}}{2} \leq \text{greedy solution} \leq PRIMAL^* \leq DUAL^* \leq \Delta_{\phi_{GR}}$$

Now, recall that Solution for Vertex Cover is a *minimization* problem. We saw its dual:

$$\begin{cases} \min \sum_{v\in V} x_v \\ y_{\{u,v\}} : x_u + x_v \geq 1 \\ x_v \geq 0 \end{cases} \tag{22}$$

Let us now see its primal:

$$\begin{cases} \max \sum_{\{u,v\}\in E} y_{\{u,v\}} \\ x_v : \sum_{u:\{u,v\}\in E} y_{\{u,v\}} \leq 1 \\ y_{\{u,v\}} \geq 0 \end{cases} \tag{23}$$

For all maximal matching $M \subseteq E$. For the dual, we set

$$x_u = \begin{cases} 1 & \text{if } e \in M \text{ s.t. } u \in E \\ 0 & \text{o/w} \end{cases} \tag{24}$$

Is this dual solution feasible? Yes. By contradiction, if $x_u + x_v < 1$, for some $\{u,v\} \in E$, we could add $\{u,v\}$ to M. Then M wouldn't be maximal. What's the value of the solution? It is $\sum_{v\in V} x_v = 2|M|$.

Instead, for the primal, we set

$$y_{\{u,v\}} = \begin{cases} 1 & \text{if } \{u,v\} \in M \\ 0 & \text{o/w} \end{cases} \tag{25}$$

In the algorithm, if there exists and edge e, we add both endpoints to the solution, and then we remove them from the graph. THe solution is feasible because M is a matching.

$$|M| \leq PRIMAL^* \leq DUAL^* \leq 2|M|$$

3.7 Greedy algorithms for Densest Subgraph

Going back to Densest subgraph problem, we want to improve the greedy algorithm, by parallelizing it and possibly make it so the number of iterations is reduced.

```
procedure PARALLEL_GREEDY($G(V, E)$)
    $S_0 \leftarrow V$
    $i \leftarrow 0$
    while $S_i \neq \emptyset$ do
        $A_i \leftarrow \{v \mid v \in S_i \wedge \deg_{S_i}(v) = \min_{u \in S_i} \deg_{S_i}(u)\}$
        $S_{i+1} \leftarrow S_i \setminus A_i$
        $i \leftarrow i + 1$
    end while
    return a set $S_i$ having larget density $\rho(S_i)$
end procedure
```

There are instances in which this algorithm saves many iterations, as well as instances in which it doesn't save much time. Let us think for example of a graph in which each component is two vertices connected together; it is easy to see that the graph that the algorithm would stop after only 1 iteration. Let us now consider a path; the algorithm, at each iteration, would remove both endpoints of the path, thus requiring $\approx \frac{n}{2}$ iterations, if n is the number of nodes.

So we need a better algorithm. See the algorithm 7.

Algorithm 7 Greedy average

```
procedure GREEDY_AVG($G(V, E)$)
    $S_0 \leftarrow V$
    $i \leftarrow 0$
    while $S_i \neq \emptyset$ do
        let $v_i$ be a node s.t. $\deg_{S_i}(v_i) \leq \text{avg}_{v \in S_i} \deg_{S_i}(v)$
        $S_{i+1} \leftarrow S_i \setminus \{v_i\}$
        $i \leftarrow i + 1$
    end while
    return a set $S_i$ having larget density $\rho(S_i)$
end procedure
```

Does this solve the problem? Not really. Consider a path with a triangle in one endpoint. For such a graph the algorithm requires $\approx n$ iterations.

See the alorithm 8

Algorithm 8 Greedy average

```
procedure GREEDY_AVG($G(V, E)$)
    $S_0 \leftarrow V$
    $i \leftarrow 0$
    while $S_i \neq \emptyset$ do
        $A_i \leftarrow \{v_i \mid \deg_{S_i}(v_i) \leq \operatorname{avg}_{v \in S_i} \deg_{S_i}(v)\}$
        $S_{i+1} \leftarrow S_i \setminus A_i$
        $i \leftarrow i + 1$
    end while
    return a set $S_i$ having larget density $\rho(S_i)$
end procedure
```

Still no solution. A path is once once again an instance in which this algortihm doesn't save much time. Though this is not a bad idea per se, so why not try to take a little more than the average?

Algorithm 9 Greedy ε (Bahmani, Kumar, Vassilvitskii, 2012)

```
procedure GREEDY$_\varepsilon$($G(V, E)$)
    $S_0 \leftarrow V$
    $i \leftarrow 0$
    while $S_i \neq \emptyset$ do
        $A_i \leftarrow \{v_i \mid v_i \in S_i \wedge \deg_{S_i}(v_i) \leq (1 + \varepsilon) \operatorname{avg}_{v \in S_i} \deg_{S_i}(v)\}$
        $S_{i+1} \leftarrow S_i \setminus A_i$
        $i \leftarrow i + 1$
    end while
    return a set $S_i$ having larget density $\rho(S_i)$
end procedure
```

Now we want to prove two things about this algorithm:

- $GREEDY_\varepsilon$ gives a good approximation to DS (Lemma 3.20)
- $GREEDY_\varepsilon$ runs quickly (Lemma 3.21)

Lemma 3.20. $GREEDY_\varepsilon$ returns a $2(1 + \varepsilon)$-approximation, $\forall \varepsilon > 0$.

Proof. Let $S^* \neq \emptyset$ be an optimal solution (then, wlog, $|S^*| \geq 2$).

We claim that $\forall v \in S^*$, $\deg_{S^*}(v) \geq \rho(S^*) \triangleq \dfrac{|E(S^*)|}{|S^*|}$.

$$\begin{aligned}\rho(S^*) = \frac{|E(S^*)|}{|S^*|} &\overset{\text{(by optimality of } S^*)}{\geq} \rho(S^* \setminus \{v\}) = \\ &= \frac{|E(S^* \setminus \{v\})|}{|S^* \setminus \{v\}|} = \frac{|E(S^*)| - \deg_{S^*}(v)}{|S^*| - 1}\end{aligned}$$

So $\dfrac{|E(S^*)| - \deg_{S^*}(v)}{|S^*| - 1} \leq \dfrac{|E(S^*)|}{|S^*|}$.

This implies that $|E(S^*)| - \deg_{S^*}(v) \leq \frac{|S^*|-1}{|S^*|}|E(S^*)| = (1 - \frac{1}{|S^*|})|E(S^*)|$.

Thus $|E(S^*)| - \deg_{S^*} \leq |E(S^*)| - \frac{|E(S^*)|}{|S^*|}$.

So $\frac{|E(S^*)|}{|S^*|} \leq \deg_{S^*}(v)$. And this proves the first claim.

Our second claim is that at least 1 node isremoved in each iteration (this is more technical, note in fact that, if it wasn't like that, the algorithm might run forever).

$$\min_{v \in S_i} \deg_{S_i}(v) \leq \operatorname*{avg}_{v \in S_i} \deg_{S_i}(v) \leq (1+\varepsilon) \operatorname*{avg}_{v \in S_i} \deg_{S_i}(v)$$

Thus $v \in A_i$ (i.e. if $|S_i| \geq 1 \rightarrow |A_i| \geq 1$).

Consider the first iteration i s.t. at least one node of S^* is removed from S_i. That is, let i be the smallest integer s.t. $A_i \cap S^* \neq \emptyset$.

Let $v \in A_i \cap S^*$. Since i is the first iteration in which $A_i \cap S^* \neq \emptyset$, it must be that $S_i \supseteq S^*$.

$$\begin{aligned}\rho(S^*) &\overset{\text{(claim 1)}}{\leq} \deg_{S^*}(v) \overset{(S_i \supseteq S^*)}{\leq} \deg_{S_i}(v) \overset{\text{(greedy choice)}}{\leq} \\ &\leq (1+\varepsilon) \operatorname*{avg}_{u \in S_i} \deg_{S_i}(u) = (1+\varepsilon)\frac{\sum_{u \in S_i} \deg_{S_i}(u)}{|S_i|} = \\ &= (1+\varepsilon)\frac{2|E(S_i)|}{|S_i|} = 2(1+\varepsilon)\rho(S_i)\end{aligned}$$

Thus $\rho(S_i) \geq \frac{1}{2(1+\varepsilon)}\rho(S^*)$.

So $GREEDY_\varepsilon$ returns something no worse than a $2(1+\varepsilon)$-approximation. $\square$

Lemma 3.21. *$GREEDY_\varepsilon$ iterates for at most $O(\frac{\log n}{\varepsilon})$ times, $\forall \varepsilon > 0$.*

Proof. Fix an iteration i.

$$\begin{aligned}2|E(S_i)| &= \sum_{v \in S_i} \deg_{S_i}(v) = \sum_{v_i \in A_i} \deg_{A_i}(v_i) + \sum_{v_i \in S_i \setminus A_i} \deg_{S_i}(v_i) \geq \\ &\geq 0 + \sum_{v_i \in S_i \setminus A_i} \deg_{S_i}(v_i) > \sum_{v \in S_i \setminus A_i} (\operatorname*{avg}_{u \in S_i} \deg_{S_i}(u))(1+\varepsilon) = \\ &= \sum_{v \in S_i \setminus A_i} (2\frac{|E(S_i)|}{|S_i|}(1+\varepsilon)) = |S_i \setminus A_i| 2\frac{|E(S_i)|}{|S_i|}(1+\varepsilon) \overset{(A_i \subseteq S_i)}{=} \\ &= (|S_i| - |A_i|)2\frac{|E(S_i)|}{|S_i|}(1+\varepsilon)\end{aligned}$$

Thus,

$$\begin{aligned}2|E(S_i)| > (|S_i| - |A_i|)2\frac{|E(S_i)|}{|S_i|}(1+\varepsilon) = (1 - \frac{|A_i|}{|S_i|})2|E(S_i)|(1+\varepsilon) = \\ = (1 - \frac{|A_i|}{|S_i|})2|E(S_i)| + (1 - \frac{|A_i|}{|S_i|})2|E(S_i)|\varepsilon = \\ = 2|E(S_i)| - 2|A_i|\frac{|E(S_i)|}{|S_i|} + (1 - \frac{|A_i|}{|S_i|})2|E(S_i)|\varepsilon\end{aligned}$$

$1 > 1 - \frac{|A_i|}{|S_i|} + (1 - \frac{|A_i|}{|S_i|})\varepsilon$. Then $|S_i| > |S_i| - |A_i| + (|S_i| - |A_i|)\varepsilon$.

$(1+\varepsilon)|A_i| \geq \varepsilon|S_i| \to |A_i| \geq \frac{\varepsilon}{1+\varepsilon}|S_i|$.

Now, recall $|A_i| = |S_i| - |S_{i+1}|$. Then $|S_i| - |S_{i+1}| \geq \frac{\varepsilon}{1+\varepsilon}|S_i|$.

$|S_i|(1 - \frac{\varepsilon}{1+\varepsilon}) \geq |S_{i+1}|$.

Thus,

$$|S_{i+1}| \leq \frac{1}{1+\varepsilon}|S_i|$$

Then,

$$\begin{aligned}&|S_0| = n \\ &|S_1| \leq \frac{1}{1+\varepsilon}|S_0| = (1+\varepsilon)^{-1} \cdot n \\ &|S_2| \leq \frac{1}{1+\varepsilon}|S_1| = (1+\varepsilon)^{-2} \cdot n \\ &\vdots \\ &|S_i| \leq \frac{1}{1+\varepsilon}|S_{i-1}| = (1+\varepsilon)^{-i} \cdot n\end{aligned}$$

Consider $i^* = \lceil \log_{1+\varepsilon} \rceil$. The procedure either ends before iteration i^*, or $|S_{i^*}| \leq (1+\varepsilon)^{-i^*} \cdot n \leq (1+\varepsilon)^{-\log_{1+\varepsilon} n} \cdot n = \frac{1}{n}n = 1$.

The number of iterations, then, is at most $i^* = O(\log_{1+\varepsilon} n) = O(\frac{\log n}{\log(1+\varepsilon)}) = O(\frac{\log n}{\varepsilon})$. □

3.8 Max-Cut

Recall Max-Cut problem. Given a graph $G(V, E)$, find a subset $S \subseteq V$ s.t. $|Cut(S, V \setminus S)|$ is maximized. Where

$$Cut(S, V \setminus S) = \{e \mid e \in E \wedge |e \cap S| = 1\}$$

What could be an LP for Max-Cut? Let us try the following program

$$\begin{cases} \max \sum_{\{i,j\}\in E} x_{\{i,j\}} \\ 0 \le x_{\{i,j\}} & \forall \{i,j\} \in E \end{cases} \tag{26}$$

It is easy to see that this is not a good LP. It has an unbounded integrality gap; $\forall \tau > 0$, $\forall e \in E$, set $x_e = \tau$.

So we modify slightly above LP.

$$\begin{cases} \max \sum_{\{i,j\}\in E} x_{\{i,j\}} \\ 0 \le x_{\{i,j\}} \le 1 & \forall \{i,j\} \in E \end{cases} \tag{27}$$

What is the IG of this LP? $LP^* = |E|$ (set $x_{\{i,j\}} = 1$, $\forall \{i,j\} \in E$). So, $\forall G(V,E)$, the Max-Cut optimum value lies within $\frac{|E|}{2}$ and $|E|$. Thus $IG \le 2$.

A 2-approximating LP is trivial. Can we do better?

Let us now take the LP just defined, and add an inequality for triangles subgraphs (i.e. the graph K_3).

$$\begin{cases} \max \sum_{\{i,j\}\in E} x_{\{i,j\}} \\ x_{\{i,j\}} + x_{\{j,k\}} + x_{\{i,k\}} \le 2 & \forall \{i,j,k\} \in \binom{V}{3} \text{ s.t. } \{i,j\},\{j,k\},\{i,k\} \in E \\ 0 \le x_{\{i,j\}} \le 1 & \forall \{i,j\} \in E \end{cases} \tag{28}$$

Suppose that $S, V \setminus S$ is an integral solution to Max-cut. Then, the value of this solution is $|Cut(S, V \setminus S)|$.

Given $S, V \setminus S$, let us set

$$x_{\{i,j\}} = \begin{cases} 1 & \text{if } |\{i,j\} \cap S| = 1 \\ 0 & \text{o/w} \end{cases} \tag{29}$$

Then $\sum_{\{i,j\}\in E} x_{\{i,j\}} = |Cut(S, V \setminus S)|$.

Is the fractional solution feasible? Let i, j, k be a triangle in $G(V,E)$.

If $|\{i,j,k\} \cap S| \in \{0,3\}$, then $x_{\{i,j\}} + x_{\{j,k\}} + x_{\{i,k\}} = 0 \le 2$.

If $|\{i,j,k\} \cap S| \in \{1,2\}$, then $x_{\{i,j\}} + x_{\{j,k\}} + x_{\{i,k\}} = 2 \le 2$.

In any case all the constraints are satisfied, thus the fractional solution is feasible.

Let us now consider another LP, where we add an additional inequality (to the one already added for triangles). We add the *triangle inequality.*

$$\begin{cases} \max \sum_{\{i,j\}\in E} x_{\{i,j\}} \\ x_{\{i,j\}} + x_{\{j,k\}} + x_{\{i,k\}} \le 2 & \forall \text{triangle } \{i,j,k\} \\ x_{\{i,j\}} + x_{\{j,k\}} \ge x_{\{i,k\}} & \forall \text{triangle } \{i,j,k\} \\ 0 \le x_{\{i,j\}} \le 1 & \forall \{i,j\} \in E \end{cases} \tag{30}$$

One should prove that this ia a relaxation of Max-cut.

In the 90s the following has been proved (TODO check if it actually has been proven in the 90s). If one takes the original Max-Cut LP, and adds to it at most $n^{O(1)}$ constraints, each involving $O(1)$ variables (while still guaranteeing that the LP is a relaxation), then the integrality gap of the LP is at least $2-\varepsilon$, $\forall \varepsilon > 0$.

Goemans-Williamson came up with an algorithm based on mathematical programming (but not on LPs) which gives a $0.878\ldots$-approximation to Max-Cut.

We now see a different kind of mathematical program for solving Max-Cut. A **Quadratic Program** (QP).

$$\begin{cases} \max \sum_{\{i,j\}\in E} \frac{1-x_i x_j}{2} & \\ x_i^2 = 1 & \forall i \in V \\ x_i \in \mathbb{R} & \forall i \in V \end{cases} \tag{31}$$

Note that

$$\begin{cases} x_i^2 \le 1 \\ x_i^2 \ge 1 \end{cases} \iff \begin{cases} x_i^2 \le 1 \\ -x_i^2 \le -1 \end{cases} \tag{32}$$

It follows that $x_i \in \{1,-1\}$, $\forall i \in V$.

If i and j are s.t. $x_i = x_j$ then $\dfrac{1-x_i x_j}{2} = \dfrac{1-1}{2} = 0$.

If i and j are s.t. $x_i \neq x_j$ then $\{x_i, x_j\} = \{1,-1\}$. Thus $\dfrac{1-x_i x_j}{2} = \dfrac{1-(-1)}{2} = 1$.

Lemma 3.22. Max-Cut $OPT = QP^*$.

Unfortunately, quadratic programs, in general, are NP-Hard.

3.9 Semi-Definite Programming (SDP)

$$\begin{cases} \max \sum_{1\le i\le j\le n} c_{ij} \cdot \underline{v}_i \cdot \underline{v}_j & \\ \sum_{1\le i\le j\le n} a_{ijk} \cdot \underline{v}_i \cdot \underline{v}_j \le b_k & \forall k = 1,\ldots,m \\ \underline{v}_i \in \mathbb{R}^n & \forall i = 1,\ldots,n \end{cases} \tag{33}$$

Where

$$\underline{v}_i \cdot \underline{v}_j = \sum_{k=1}^{t} v_i(k) \cdot v_j(k)$$

Also, the dimensionality of $\mathbb{R}^n$ (i.e. n) has to be $\Omega(n)$, where n is the number of variables.

The optimal solution to a SDP can be approximated to an additive $\forall \varepsilon > 0$ in time polynomial in $n \log \frac{1}{\varepsilon}$.

3.10 Max-Cut SDP

Recall the QP for Max-Cut, we wanna relax it into an SDP.

$$\text{QP: } \begin{cases} \max \sum_{\{i,j\}\in E} \frac{1-x_i x_j}{2} & \\ x_i^2 = 1 & \forall i \in V \\ x_i \in \mathbb{R} & \forall i \in V \end{cases} \overset{\text{relax}}{\Longrightarrow} \text{SDP: } \begin{cases} \max_{\{i,j\}\in E} \frac{1-\underline{v}_i \cdot \underline{v}_j}{2} & \\ \underline{v}_i \cdot \underline{v}_i = 1 & \forall i \in \mathbb{R} \\ \underline{v}_i \in \mathbb{R}^n & \forall i \in \mathbb{R} \end{cases} \tag{34}$$

The relaxation if quite natural, there are no complicated reasonings. The x_i variables in the QP can be seen as vectors of size 1. We then transform them in vectors of size n in the SDP.

Lemma 3.23. *$SDP^* \geq QP^* (= OPT)$.*

Proof. Let $x_1, x_2, \ldots, x_n$ be a feasible solution to QP.

Set

$$\underline{v}_i = \begin{pmatrix} x_i \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \forall i \in [n] \tag{35}$$

We now want to prove that:

- this SDP solution is feasible
- the value of the SDP is not smaller than the QP one

Let us prove the first claim. $\forall i \in [n]$,

$$\underline{v}_i \cdot \underline{v}_i = \sum_{j=1}^{n} \underline{v}_i(j) \cdot \underline{v}_i(j) = \sum_{j=1}^{n} \underline{v}_i(j)^2 = \underline{v}_i(j)^2 = x_i^2 \overset{\text{feasibility of QP}}{=} 1$$

Then $\underline{v}_i \cdot \underline{v}_i \geq 1$. Thus the SDP solution is feasible.

Let us now prove the second claim.

$$\sum_{\{i,j\}\in E} \frac{1 - \underline{b}_i \cdot \underline{v}_j}{2} = \sum_{\{i,j\}\in E} \frac{1 - x_i x_j}{2}$$

So, not only $SDP \geq QP$, but actually $SDP = QP$. □

Corollary 3.23.1. *$SDP^* \geq OPT$* (Max-Cut's optimal).

Can we bound the integrality gap? Can we find a constant c s.t. $OPT \geq c \cdot SOP^*$

To do so, we are going to see an important rounding algorithm, by *Goemans-Williamson.*

For the definition of an hypersphere see Hypersphere.

Algorithm 10 Goemans-Williamson procedure

```
procedure GW($G(V, E)$)
    solve Max-Cut SDP, obtaining vectors $\underline{v}_1, \dots, \underline{v}_n$ (for $n = |V|$)
    sample a UAR vector $\underline{Y}$ on the $n$-dimensional hypersphere $H_n$
    $S \leftarrow \{i \mid \underline{v}_i \cdot \underline{Y} \geq 0\}$
    return $(S, V \setminus S)$
end procedure
```

First of all, let us consider any of the $\underline{v}_i$ vectors of the SDP solution. Because of the constraint $\underline{v}_i \cdot \underline{v}_i = 1$, all the $\underline{v}_i$ vectors are on the hypersphere, by construction.

The high level idea is the following. Let us consider the simple case of H_2 (i.e. a circle), though the idea holds for any dimension. Let $\underline{v}$ and $\underline{w}$ be any two vectors.

Suppose $\|\underline{v}\|_2 = \|\underline{w}\|_2 = 1$, then:

- $\|\underline{v}\|_2^2 = \|\underline{w}\|_2^2 = 1^2 = 1$
- $\underline{v} \cdot \underline{v} = \underline{w} \cdot \underline{w} = 1$
- $\underline{v} \cdot \underline{w} = \cos(\theta_{\underline{v},\underline{w}})$ (see *Cosine similarity*)

Let us divide the circle in two equal parts, with a straight line. This straight line would be our UAR vector $\underline{Y}$. The two parts of the circle will be our cut of the graph: if two vectors lie on the same half of the circle, then they are on the same part of the cut. Equivalently, if two vectors are divided by $\underline{Y}$, then they are divided by the cut.

This is why we look at the cosine:

- smaller angle $\rightarrow$ bigger cosine ($\cos 0° = 1$) $\rightarrow$ higher probability that two vectors are not divided
- bigger angle $\rightarrow$ smaller cosine ($\cos 180° = -1$) $\rightarrow$ higher probability that two vectors are divided

Let us see some cases:

- $\theta_{\underline{v},\underline{w}} = 0° \rightarrow Pr[\underline{v}, \underline{w} \text{ both go in } S] = 1$
- $\theta_{\underline{v},\underline{w}} = 180° \rightarrow Pr[\underline{v}, \underline{w} \text{ both go in } S] = 0$
- $\theta_{\underline{v},\underline{w}} = \frac{\pi}{2}° \rightarrow Pr[\underline{v}, \underline{w} \text{ both go in } S] = \frac{1}{2}$

Lemma 3.24. Let $\{i, j\} \in E$, then $Pr[|\{i, j\} \cap S| = 1] = Pr[\{i, j\} \text{ is cut}] = \frac{\theta_{\underline{v}_i,\underline{v}_j}}{\pi}$.

Proof (sketch). Consider a ”plane” (a 2-D flat) passing through the origin

$$O = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

And consider two vectors $\underline{v}_i, \underline{v}_j$.

Let $\theta_{\underline{v}_i,\underline{v}_j}$ be the angle between the two vectors. Then the angle between the projections of the vectors remains $\theta_{\underline{v}_i,j\underline{v}_j}$.

Since $\underline{Y}$ is sampled UAR on the hypersphere, the projection of $\underline{Y}$ on the plane will be a UAR vector/direction, starting from O. Note that the projection of $\underline{Y}$ might be a direction, because it could be too short.

$$\begin{aligned}
Pr[\{i,j\} \text{ is cut}] &= Pr[(i \in S \wedge j \notin S) \vee (i \notin S \wedge \in S)] = \\
&= Pr[i \in S \wedge j \notin S] + Pr[i \notin S \wedge j \in S] = \\
&= Pr[\underline{v}_i \cdot \underline{Y} \geq 0 \wedge \underline{v}_j \cdot \underline{Y} < 0] + Pr[\underline{v}_i \cdot \underline{Y} < 0 \wedge \underline{v}_j \cdot \underline{Y} \geq 0] = \\
&= \frac{\theta_{ij}}{2\pi} + \frac{\theta_{ij}}{2\pi} = \frac{\theta_{ij}}{\pi}
\end{aligned}$$

□

Observe that GW reduces to our original UAR-cut solution if $\theta_{ij} = \frac{\pi}{2}, \forall \{i,j\} \in \binom{V}{2}$. In particular, GW uses the SDP solution to induce controlled bias on the random-cut algorithm.

Theorem 3.25. If $(S, V \setminus S)$ is the cut returned by GW, then

$$\mathbb{E}[|Cut(S, V \setminus S)|] \geq \alpha_{GW} SDP^* (\geq \alpha_{GW} OPT)$$

For $\alpha_{GW} = \dfrac{2}{\pi} \min_{x \in (-1,1)} \dfrac{\arccos(x)}{1-x} \approx 0.878\ldots.$

Proof.

$$\begin{aligned}
\mathbb{E}[|Cut(S, V \setminus S)|] &= \sum_{\{i,j\} \in E} Pr[\{i,j\} \text{ cut by } S] \overset{Lemma\ 3.24}{=} \\
&= \sum_{\{i,j\} \in E} \frac{\theta_{\underline{v}_i,\underline{v}_j}}{\pi} = \sum_{\{i,j\} \in E} \frac{\arccos(\underline{v}_i, \underline{v}_j)}{\pi}
\end{aligned}$$

Now, suppose α is s.t. $\dfrac{\arccos(x)}{\pi} \geq \alpha(\frac{1}{2} - \frac{x}{2})$, for each $x \in (-1,1)$.

Then $\mathbb{E}[|Cut(, V \setminus S)|] \geq \alpha \sum_{\{i,j\} \in E} \frac{1 - \underline{v}_i \cdot \underline{v}_j}{2} = \alpha \cdot SDP^*$.

What is the larget α for which $\dfrac{\arccos(x)}{\pi} \geq \alpha(\frac{1}{2} - \frac{x}{2})$ holds $\forall x \in (-1,1)$?

$\forall x \in (-1,1)$, $\alpha \frac{1-x}{2} \leq \frac{\arccos(x)}{\pi} \rightarrow \alpha(1-x) \leq \frac{2}{\pi} \arccos(x) \rightarrow \alpha \leq \frac{2}{\pi} \frac{\arccos(x)}{1-x}$

Then,

$$\alpha = \alpha_{GW} \triangleq \min_{x \in (-1,1)} \left(\frac{2}{\pi} \frac{\arccos(x)}{1-x}\right) \approx 0.878\ldots$$

□

So, we proved that

$$0.878\dots = \alpha_{GW} \leq IG(\text{Max-Cut SDP}) \leq ?$$

The G-W analysis is tight. $\forall \varepsilon > 0$, $\exists$ graph $G(V, E)$ s.t. the IG of the SDP Max-relaxation is no better than $\alpha_{GW} + \varepsilon$.

We now prove a weaker claim. We consider a graph instance which has an IG slightly worse than $0.878\dots$.

Consider the graph C_5 (i.e. a cycle of length 5), as seen in figure 1.

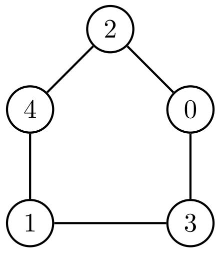

Figure 1: Graph C_5

Let our cut be $S = \{0, 1, 4\}$.

We cut 4 edges. Thus $OPT \geq 4$.

Lemma 3.26. If k is odd, then it is impossible to cut all the edges of C_k (i.e. cycle on k nodes).

Proof. We can try to do cut all the edges by picking nodes with the following procedure. Follow the cycle, alternating nodes to pick; i.e. we pick one and skip the following, then pick the following and skip the one after, and continue to do so.

Because the cycle is of odd length, then at the end of the cycle we must pick two adjacent nodes, and the edge that connects these two nodes won't be cut. $\square$

Note that this is the same reason why we can't have a perfect matching on an odd cycle.

So, as a corollary, we get that, in C_5, $OPT < |E| = 5$.

Thus $4 \leq OPT < 5 \rightarrow OPT = 4$.

Now that we have the optimal value, let us try to solve the SDP for this graph. Consider the following solution. $\forall i \in \{0, 1, \dots, 4\}$, set:

$$\underline{\mathrm{v}}_i = (\cos(i\frac{2\pi}{5}), \sin(i\frac{2\pi}{5}), 0, 0, 0)$$

For this solution to be feasible we must have that $\forall i \in V, \underline{\mathrm{v}}_i \cdot \underline{\mathrm{v}}_i = 1$. Before showing it, recall that $\forall x, \cos^2 x + \sin^2 x = 1$.

$$\begin{aligned}\underline{v}_i \cdot \underline{v}_i &= \|\underline{v}_i\|_2^2 = \\ &= \sum_{j=1}^{5} \underline{v}_i(j) \cdot \underline{v}_i(j) = \sum_{j=1}^{2} \underline{v}_i(j) \cdot \underline{v}_i(j) = \\ &= \cos^2(i\frac{2\pi}{5}) + \sin^2(i\frac{2\pi}{5}) = 1\end{aligned}$$

Thus the solution is feasible. Let us now calculate the value of such solution.

$$\begin{aligned}&SDP = \\ &\quad = \sum_{\{i,j\}\in E} \frac{1 - \underline{v}_i \cdot \underline{v}_i}{2} = \\ &\quad = \frac{1-\underline{v}_0\cdot\underline{v}_2}{2} + \frac{1-\underline{v}_2\cdot\underline{v}_4}{2} + \frac{1-\underline{v}_4\cdot\underline{v}_1}{2} + \frac{1-\underline{v}_1\cdot\underline{v}_3}{2} + \frac{1-\underline{v}_3\cdot\underline{v}_0}{2} = \\ &\quad = \frac{1-\cos(\theta_{02}) + 1-\cos(\theta_{24}) + 1-\cos(\theta_{41}) + 1-\cos(\theta_{13}) + 1-\cos(\theta_{30})}{2} = \\ &\quad = \frac{1-\cos(\frac{4\pi}{5})5}{2} = \frac{5}{2}(1-\cos(\frac{4\pi}{5})) \approx 4.522\ldots\end{aligned}$$

Thus, the IG is no better than

$$0.878\cdots = \alpha_{GW} \overset{\forall G}{\leq} IG \overset{\exists G}{\leq} \frac{4}{4.522\ldots} = 0.884\ldots$$

The first inequality (i.e. $\alpha_{GW} \leq IG$) is actually tight, but tightness is harder to prove.

So, for Max-Cut we have a $0.878\ldots$ approximation that cannot be improved by modifying the random rounding algorithm. Can the approximation be improved at all? There are reasons to believe that $0.878\ldots$ is the best one we can obtain in polynomial time.

4 Unique Game Conjecture

4.1 Unique Label Cover

It is an optimization problem.

We are given a *bipartite graph* $G((V,W),E)$ (see Bipartite graph), and given a set of labels $[m] = \{1,2,\ldots,m\}$, and $\forall e = \{v,w\} \in E$ a permutation $\pi_{vw} : [m] \to [m]$.

Find an assignment of labels to the vertices, $\phi : V \cup W \to [m]$, s.t. the set S of satisfied edges is as large as possible.

$$S = \{v,w\} \in E \mid v \in V \wedge w \in W \wedge \pi_{vw}(\phi(v)) = \phi(w)$$

4.2 Unique Games Conjecture

Formulated by Khot in 2002.

For each $\varepsilon > 0$, it is NP-Hard to determine if, in a Unique Label Cover problem, with $m = m(\varepsilon)$ labels, if:

- Each assignment of labels (each solution) satisfies no more than $\varepsilon|E|$ edges, or
- There exists an assignment (a solution) which satisfies at least $(1-\varepsilon)|E|$ edges

From this conjecture, came a bunch of theorems stating that, if the UGC holds, than some problem is NP-Hard to approximate better than a specific value. Let us see a couple of examples.

Theorem 4.1. If the UGC holds, then $\forall\varepsilon > 0$, it is NP-Hard to approximate Max-Cut to $\alpha_{GW} - \varepsilon$.

Theorem 4.2. If the UGC holds, then $\forall\varepsilon > 0$, is it NP-Hard to approximate Vertex-Cover to $2 - \varepsilon$.

4.3 Constraint Satisfaction Problems (CSPs)

CSPs of a constant arity, on a constant-sized alphabet, can be optimally approximated via a SDP-based algorithm in polytime, if the UGC holds.

4.4 Sparsest Cut problem

Sparsest Cut is not a CSP.

Given an undirected graph $G(V, E)$, find a subset $\emptyset \subset S \subset V$ that minimizes

$$\Psi(S) = \frac{|Cut(S, V \setminus S)|}{|S| \cdot |V \setminus S|}$$

Let us consider the Sparsest Cut problem for **Erdős-Renyi random graphs**. To sample an E-R r.g., one can use the algorithm 11.

Algorithm 11 E-R random graph $G(n,p)$

```
$S \leftarrow [n]$
$E \leftarrow \emptyset$
for each $\{i,j\} \in \binom{V}{2}$ do
    flip an iid coin with head probability $p$
    if coin is heads then
        $E \leftarrow E \cup \{i,j\}$
    end if
end for
return $G(V,E)$
```

$$\begin{aligned}
&\mathbb{E}[|Cut(S, V \setminus S)|] = \\
&\qquad = \sum_{i \in S} \sum_{j \in V \setminus S} Pr[\{i, j\} \in E] = \\
&\qquad = \sum_{i \in S} \sum_{j \in V \setminus S} p = |S| \cdot |V \setminus S| \cdot p
\end{aligned}$$

Thus, $\mathbb{E}_{G \sim G(n,p)}[\Psi(S)] = \frac{\mathbb{E}[|Cut(S, V \setminus S)|]}{|S| \cdot |V \setminus S|} = \frac{|S| \cdot |V \setminus S| \cdot p}{|S| \cdot |V \setminus S|} = p$

We have concluded that

$$0 \leq |Cut(S, V \setminus S)| \leq |S| \cdot |V \setminus S|$$

This implies that

$$\Psi(S) \in [0, 1]$$

Observe that, if $\Psi(S) = 0 \to S$ is disconnected from $V \setminus S$.

Our *objective function* is

$$\Psi^* = \min_{\emptyset \subset S \subset V} \Psi(S)$$

Leighton-Rao algorithm for Sparsest-Cut:

- Based on an LP
- Geometry plays a pivotal role in its analysis
- The algorithm is, essentially, a series of geometric reductions

5 Metrics

Definition 5.1 (Metric)**.** A metric d on a set S is a function $d : S \times S \to \mathbb{R}$, s.t.:

1. $d(X, Y) = d(Y, X)$, $\forall X, Y \in S$
2. $d(X, X) = 0$, $\forall X \in S$
3. $d(X, Y) + d(Y, Z) \geq d(X, Z)$, $\forall X, Y, Z \in S$ (Triangle inequality)

Definition 5.2 (Elemetary-cut metric)**.** The elementary-cut metric $d_T : S \times S \to \mathbb{R}$, for $T \subseteq S$, is defined as

$$d_T(X, Y) = \begin{cases} 1 & \text{if } |\{X, Y\} \in T| = 1 \\ 0 & \text{o/w} \end{cases} \tag{36}$$

Lemma 5.1. *If $d_T : S \times S \to \mathbb{R}$ is an elementary-cut metric, then d_T is a metric.*

Proof. Axioms 1. and 2. are trivial to verify.

Let $X, Y, Z \in S$:

- if $X, Y, Z \in T$, then $0 = d_T(X,Y) + d_T(Y,Z) \geq d_T(X,Z) = 0$
- if $X, Y, Z \notin T$, then $0 = d_T(X,Y) + d_T(Y,Z) \geq d_T(X,Z) = 0$

It remains a third case, itself subdivided in three other subcases.

Assume $\exists A \in \{T, S \setminus T\}$ s.t. exactly one of X, Y, Z is in A; i.e. we divide X, Y and Z in the two partitions of S.

- if $X \in A$, then $1 = d_T(X,Y) + d_T(Y,Z) \geq d_T(X,Z) = 1$
- if $Z \in A$, then $1 = d_T(X,Y) + d_T(Y,Z) \geq d_T(X,Z) = 1$
- if $Y \in A$, then $2 = d_T(X,Y) + d_T(Y,Z) \geq d_T(X,Z) = 0$

□

We want to argue that the Sparsest Cut problem is an optimization problem, over elementary-cut metrics. That is, the goal is to minimize $\phi(d_T)$, where $d_T : V \times V \to \mathbb{R}$ is an elementary-cut metric over V, and

$$\phi(d_T) = \frac{\sum_{\{x,y\} \in E} d_T(X,Y)}{\sum_{\{x,y\} \in \binom{V}{2}} d_T(X,Y)}$$

Lemma 5.2. If d_T is an elementary-cut metric over V, then

$$\phi(d_T) = \frac{|Cut(S, V \setminus S)|}{|T| \cdot |V \setminus T|}$$

Let P be any predicate. $[P]$ has value 1 is P is true, and 0 if P is false.

Proof.

$$\begin{aligned} \phi(d_T) &= \\ &= \frac{\sum_{\{x,y\} \in E} d_T(X,Y)}{\sum_{\{x,y\} \in \binom{V}{2}} d_T(X,Y)} = \frac{\sum_{\{x,y\} \in E} [X \in T \text{ xor } Y \in T]}{\sum_{\{x,y\} \in \binom{V}{2}} [X \in T \text{ xor } Y \in T]} = \\ &= \frac{|Cut(T, V \setminus T)|}{|T| \cdot |V \setminus T|} \end{aligned}$$

□

Definition 5.3 (Cut metric)**.** Let $d_{T_1}, d_{T_2}, \ldots, d_{T_k} : V \times V \to \mathbb{R}$ be elementary cut metric, and let $\lambda_1, \lambda_2, \ldots, \lambda_k > 0$, then $d : V \times V \to \mathbb{R}$ defined as

$$d(X,Y) = \sum_{i=1}^{k} (\lambda_i \cdot d_{T_i}(X,Y))$$

is a cut metric.

Cut metric are less "integral" (more "fractional"/flexible) than elementary cut metrics.

Lemma 5.3. Let d be a cut metric with cuts $T_1, T_2, \ldots, T_k$, then

$$\phi(d) \geq \min_{i \in [k]} \phi(d_i)$$

Then, minimizing over cut metrics is equivalent to minimizing over elementary-cut metrics.

Proof. Since d is a cut metric with cuts $T_1, \ldots, T_k$, $\exists \lambda_1, \lambda_2, \ldots, \lambda_k > 0$ s.t. $d(X,Y) = \sum_{i=1}^k (\lambda_i \cdot d_{T_i}(X,Y))$, $\forall X, Y \in V$.

Then,

$$\phi(d) = \frac{\sum_{\{u,v\} \in E} d(u,v)}{\sum_{\{u,v\} \in \binom{V}{2}} d(u,v)} = \frac{\sum_{\{u,v\} \in E} \sum_{i=1}^k (\lambda_i \cdot d_{T_i}(u,v))}{\sum_{\{u,v\} \in \binom{V}{2}} \sum_{i=1}^k (\lambda_i \cdot d_{T_i}(u,v))} =$$
$$= \frac{\sum_{i=1}^k \sum_{\{u,v\} \in E} (\lambda_i \cdot d_{T_i}(u,v))}{\sum_{i=1}^k \sum_{\{u,v\} \in \binom{V}{2}} (\lambda_i \cdot d_{T_i}(u,v))}$$

We now claim that, if $a_i, b_i > 0$, $\forall i \in [k]$, then

$$\frac{\sum_{i=1}^k a_i}{\sum_{i=1}^k b_i} \geq \min_{i \in [k]} \frac{a_i}{b_i}$$

Let $\rho = \min_{i \in [k]} \dfrac{a_i}{b_i}$.

Then $\dfrac{\sum_{i=1}^k a_i}{\sum_{i=1}^k b_i} = \dfrac{\sum_{i=1}^k b_i \frac{a_i}{b_i}}{\sum_{i=1}^k b_i} \geq \dfrac{\sum_{i=1}^k b_i \rho}{\sum_{i=1}^k b_i} = \rho$

Thus the claim is proved.

Now,

$$\phi(d) \geq \min_{i \in [k]} \frac{\sum_{\{u,v\} \in E} (\lambda_i \cdot d_{T_i}(u,v))}{\sum_{\{u,v\} \in \binom{V}{2}} (\lambda_i \cdot d_{T_i}(u,v))} = \min_{i \in [k]} \phi(d_{T_i})$$

□

Corollary 5.3.1.

$$\min_{\substack{d \\ d \text{ a cut metric}}} \phi(d) = \min_{\substack{d_T \\ d_T \text{ an elem. cut metric}}} \phi(d_T) = \Psi^*$$

Proof. Follows from Lemma 5.2 and Lemma 5.3. □

Exercise 5.3.1. Prove that each cut metric d is a metric.

Definition 5.4 (Normed metric)**.** Given $\underline{X}, \underline{Y} \in \mathbb{R}^d$,

$$\ell_p(\underline{X}, \underline{Y}) = \sqrt[p]{\sum_{t=1}^{d} |X(t) - Y(t)|^p}$$

Theorem 5.4. ℓ_p is a metric over $\mathbb{R}^d$, $\forall d \geq 1$, $\forall p \geq 1$.

We define ℓ_∞ as

$$\ell_\infty(\underline{X}, \underline{Y}) = \max_{t \in [d]} |X(t) - Y(t)| = \lim_{p \to \infty} \ell_p(\underline{X}, \underline{Y})$$

We are interested in some specific metrics. Consider ℓ_1, called *Manhattan distance*, which has some nice properties. It is, essentially, just a linear function with absolute values.

$$\ell_1(\underline{X}, \underline{Y}) = |\underline{X} - \underline{Y}|_1 = \sum_{t=1}^{d} |X(t) - Y(t)|$$

Lemma 5.5. If $d : V \times V \to \mathbb{R}$ is a cut metric over k cuts, then $f : V \to \mathbb{R}^k$ s.t. $d(X, Y) = |f(X) - f(Y)|_1$, $\forall X, Y \in V$.

If $X \subseteq \mathbb{R}^d$ is finite, then the $l - 1$-metric over X can be represented by a cut metric over $k = (|X| - 1)d$ cuts.

Proof. If d is a cut metric, $\exists \lambda_1, \dots, \lambda_k > 0$, and $\emptyset \subset T_1, \dots, T_k \subset V$ s.t. $\forall \{u, v\} \in \binom{V}{2}$

$$d(u, v) = \sum_{i=1}^{k} (\lambda_i d_{T_i}(u, v))$$

$\forall u \in V$, define $\underline{X}_u \in \mathbb{R}^k$ ($\underline{X}_u = f(u)$) as follows:

$$\underline{X}_u(i) = \begin{cases} \lambda_i & \text{if } u \in T_i \\ 0 & \text{o/w} \end{cases} \tag{37}$$

Let $u, v \in V$, then

$$\begin{aligned}
|\underline{X}_u - \underline{X}_v|_1 &= \sum_{i=1}^{k} |\underline{X}_u(i) - \underline{X}_v(i)| = \sum_{i=1}^{k} ([|\{u, v\} \cap T| = 1] \cdot \lambda_i) = \\
&= \sum_{i=1}^{k} (d_{T_i}(u, v) \cdot \lambda_i) = \\
&= d(u, v)
\end{aligned}$$

Now, suppose that we have $X \subseteq \mathbb{R}$. Let us start, for semplicity, with $d = 1$.

Suppose that the points that we want to embed in a cut metric are $(X_1), (X_2), \dots, (X_n)$.

Let $\pi : [n] \to [n]$ be a permutation s.t. $X_{\pi(1)} \leq X_{\pi(2)} \leq \cdots \leq X_{\pi(n)}$. For each $j = 1, \ldots, n-1$, let S_j be defined as

$$S_j = \{\pi(1), \pi(2), \ldots, \pi(j)\}$$

Moreover, $\lambda_j = X_{\pi(j+1)} - X_{\pi(j)}$. Then $\lambda_j \geq 0$, since $X_{\pi(j+1)} \geq X_{\pi(j)}$.

For each $i < j$, it hold that

$$\begin{aligned}
\lambda_i + \lambda_{i+1} + \cdots + \lambda_{j-1} = &\sum_{t=1}^{n-1} (d_{S_t}(\pi(i), \pi(j)) \cdot \lambda_t) = \\
&= (X_{\pi(j)} - X_{\pi(j-1)}) + \\
&+ (X_{\pi(j-1)} - X_{\pi(j-2)}) + \\
&\vdots \\
&+ (X_{\pi(i+1)} - X_{\pi(i)}) = \\
&= X_{\pi(j)} - X_{\pi(i)} = |X_{\pi(j)} - X_{\pi(i)}|_1
\end{aligned}$$

Thus, an ℓ_1-metric X, over $d = 1$ dimension, can be represented by a cut metric (over $|X| - 1$ cuts). In general, an ℓ_1-metric over d dimensions is equal to the sum of d ℓ_1-metrics over 1 dimension each.

Thus any ℓ_1-metric X over d dimensions can be emebedded into a cut metric over $d(|X| - 1)$ cuts. $\square$

Corollary 5.5.1.

$$\min_{\substack{d \\ d \text{ a } \ell_1\text{-metric}}} \phi(d) = \min_{\substack{d \\ d \text{ a cut metric}}} \phi(d) = \min_{\substack{d_T \\ d_T \text{ an elem. cut metric}}} \phi(d_T) = \min_{\emptyset \subset S \subset V} \Psi(S)$$

Observe that

$$\begin{cases} \min \phi(d) \\ d \text{ an } \ell_1\text{-metric} \end{cases} \iff \begin{cases} \min \sum_{\{u,v\} \in E} d(u,v) \\ d \text{ an } \ell_1\text{-metric} \\ \sum_{\{u,v\} \in \binom{V}{2}} d(u,v) = 1 \end{cases} \tag{38}$$

Theorem 5.6 (Bourgain, Linial et al.)**.** Any metric on n points can be embedded into ℓ_1 (with $d = O(\log^2 n)$ dimensions) with "distortion" $O(\log n)$. Furthermore, the embedding can be obtained in polynomial time.

Definition 5.5 (Distortion)**.** If d and d' are metric over V, then if $\exists \alpha, \beta \geq 1$ s.t. $\forall x, y \in V$

$$\frac{d(x,y)}{\alpha} \leq d'(x,y) \leq \beta \cdot d(x,y)$$

then the distortion of the pair d, d' is not larget than $\alpha \cdot \beta$.

5.1 Leighton-Rao algorithm

We now see an algorithm for approximating Sparsest, based on LP.

Procedure:

1. Solve the metric LP

$$\begin{cases} \min \sum_{\{u,v\}\in E} X_{\{u,v\}} & \\ X_{\{u,v\}} + X_{\{v,w\}} \geq X_{\{u,w\}} & \forall u,v,w \in V \\ \sum_{\{u,v\}\in\binom{V}{2}} X_{\{u,v\}} = 1 & \\ X_{\{u,v\}} \geq 0 & \forall \{u,v\} \in \binom{V}{2} \end{cases} \tag{39}$$

2. Let d be the resulting metric
3. Apply Bourgain's theorem on d, to obtain a ℓ_1-metric d' that distorts d by at most $O(\log n)$
4. Apply the algorithm of Lemma 5.5 on d', to obtain a cut metric d'' equal to d' (i.e. no distortion)
5. For each cut T_i of d'', compute $\phi(d_{T_i})$
6. Return $d''' = \operatorname{argmin}_{T_i} \phi(d_{T_i})$

Theorem 5.7. This algorithm return a $O(\log n)$ approximation to Sparset cut.

Proof. d minimizes the metric LP, then

$$LP^* = \min_{\substack{\underline{\mathrm{d}} \\ \underline{\mathrm{d}} \text{ a cut-metric}}} \phi(\underline{\mathrm{d}}) = \min_{\substack{\underline{\mathrm{d}} \\ \underline{\mathrm{d}} \text{ a } \ell_1\text{-metric}}} \phi(\underline{\mathrm{d}})$$

d' distorts d by at most $O(\log n)$: $\phi(d') \leq \phi(d) \cdot O(\log n) \leq OPT \cdot O(\log n)$.

Moreover, $\phi(d'') = \phi(d') \leq OPT \cdot O(\log n)$.

Finally, Lemma 5.3 guarantees that $\phi(d''') \leq \phi(d'') \leq OPT \cdot O(\log n)$. □

Lemma 5.8. Each ℓ_1-metric can be embedded into a cut metric with no distortion (i.e. distortion 1). Each cut-metric can be embedded into ℓ_1 with no distortion.

That is,

$$\ell_1 \sim \text{cut metrics}$$

Theorem 5.9 (Bourgain)**.** Each metric on n points can be embedded into ℓ_2 with distortion $O(\log n)$.

This seems to indicate that ℓ_2 is "weaker" than genearl metrics. Let us see an example of it.

Definition 5.6. For a connected undirected graph $G(V,E)$, the shortest path metric on G is a metric over V s.t. $\forall u,v \in V$, $d_G(u,v)$ is the number of edges of a shortest path from u to v in G.

Observation 5.9.1. The shortest path metric satisfies the triangle inequality.

Consider the graph C_4, as in figure 2.

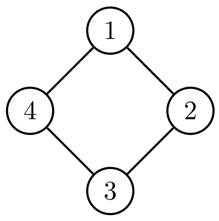

Figure 2: Graph C_4

$1 = d_G(1,2) = d_G(2,3) = d_G(3,4) = d_G(4,1)$.

$2 = d_G(1,3) = d_G(2,4)$.

If $G = C_4$, then d_G can be embedded in ℓ_1 with no distortion. Embedding shown in figure 3.

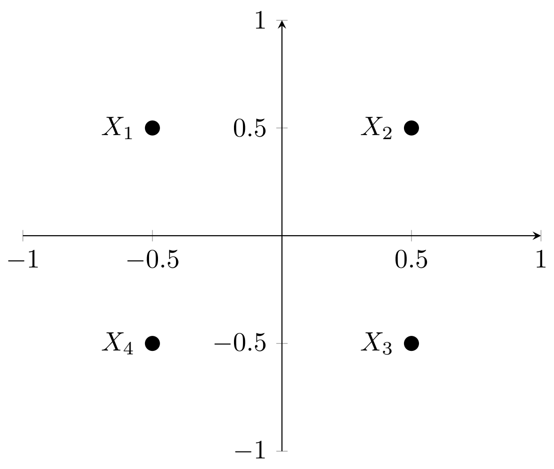

Figure 3: C_4 embedding into ℓ_1

Recall that

$$|X - Y|_1 = \sum_{i=1}^{d} |X_i - Y_i|$$

Then,

$$\begin{aligned}
&|X_1 - X_2|_1 = |X_2 - X_3|_1 = |X_3 - X_4|_1 = |X_1 - X_4|_1 = 1 \\
&|X_1 - X_3|_1 = |X_2 - X_4|_1 = 1 + 1 = 2
\end{aligned}$$

Thus, $f(i) = X_i$ is an embedding with no distortion.

Say that now we want to embed C_4 into ℓ_2. The embedding can be represented like the ℓ_1 embedding (i.e. see figure 3). The difference with ℓ_1, as it will be clear in a moment, is that points for which the shortest path is a diagonal, have a different distances in ℓ_2 and d_G (whereas in ℓ_1 and d_G it was the same).

Recall that

$$|X - Y|_2 = \sqrt{\sum_{i=1}^{d}(X_i - Y_i)^2}$$

Then,

$$|X_1 - X_2|_2 = |X_2 - X_3|_2 = |X_3 - X_4|_2 = |X_1 - X_4|_2 = 1$$
$$|X_1 - X_3|_2 = |X_2 - X_4|_2 = \sqrt{1^2 + 1^2} = \sqrt{2}$$

On the edges we have a distance of 1. On the diagonals we have a distance of $\frac{2}{\sqrt{2}} = \sqrt{2}$. Thus, the distortion of this embedding is $\sqrt{2}$.

Question: what is the minimum distortion of C_4 into ℓ_2?

Theorem 5.10. The C_4-metric cannot be embedded into ℓ_2 with distortion $< \sqrt{2}$.

Proof. Let $f(i) = Y_i$ be an embedding for $i \in [4]$, and for $Y_i \in \mathbb{R}^d$.

Let $M = \max(|Y_1 - Y_2|_2, |Y_2 - Y_3|_2, |Y_3 - Y_4|_2, |Y_1 - Y_4|_2)$ be the maximum edge length.

Let $m = \min(|Y_1 - Y_3|_2, |Y_2 - Y_4|_2)$ be the minimum diagonal length.

Lemma: $\forall Y_1, Y_2, Y_3, Y_4 \in \mathbb{R}^d$, $|Y_1 - Y_3|_2^2 + |Y_2 - Y_4|_2^2 \leq |Y_1 - Y_2|_2^2 + |Y_2 - Y_3|_2^2 + |Y_3 - Y_4|_2^2 + |Y_1 - Y_4|_2^2$. This is the "Short diagonals lemma", and its proof will be presented after the proof of this theorem.

But then, the followings hold

$$2m^2 \leq |Y_1 - Y_3|_2^2 + |Y_2 - Y_4|_2^2$$

$$4M^2 \geq |Y_1 - Y2|_2^2 + |Y_2 - Y_3|_2^2 + |Y_3 - Y_4|_2^2 + |Y_1 - Y_4|_2^2$$

Then,

$$2m^2 \leq 4M^2 \rightarrow (\frac{m}{M})^2 \leq 2 \rightarrow \frac{m}{M} \leq \sqrt{2} \rightarrow m \leq M\sqrt{2}$$

But we wanted $m = 2M$, thus the distortion has to be at least $\frac{2}{\sqrt{2}} = \sqrt{2}$. $\square$

Lemma 5.11. $\forall Y_1, Y_2, Y_3, Y_4 \in \mathbb{R}^d$,

$$|Y_1 - Y_3|_2^2 + |Y_2 - Y_4|_2^2 \leq |Y_1 - Y_2|_2^2 + |Y_2 - Y_3|_2^2 + |Y_3 - Y_4|_2^2 + |Y_1 - Y_4|_2^2$$

Proof. Recall that

$$|Y_i - Y_j|_2^2 = \sum_t (Y_i(t) - Y_j(t))^2$$

We will prove our inequality on each coordinate independently. Consider a coordinate t, and let $z_i = Y_i(t) \in \mathbb{R}$.

Claim: $(z_1 - z_2)^2 + (z_2 - z_3)^2 + (z_3 - z_4)^2 + (z_1 - z_4)^2 \geq (z_1 - z_3)^2 + (z_2 - z_4)^2$.

Sum of Squares (SOS) Proof: $(z_1 - z_2)^2 + (z_2 - z_3)^2 + (z_3 - z_4)^2 + (z_4 - z_1)^2 - (z_1 - z_3)^2 - (z_2 - z_4)^2 = (z_1 - z_2 - z_3 - z_4)^2 \geq 0$ ☐

5.2 Max-cover problem

Recall the problem Set Cover problem. We now see a variation of it.

Input: $C = \{S \mid S \subseteq [n]\} = \{S_1, \dots, S_m\}$.

Goal: find $C' \subseteq \binom{C}{k}$ s.t. $|\bigcup_{S \in C'} S|$ is as large as possible.

In algorithm 12 a greedy algorithm for solving Max-cover.

Algorithm 12 Greedy for Max-cover

```
procedure GREEDY($(C, k)$)
    $C' \leftarrow \emptyset$
    $T_0 \leftarrow \emptyset$
    for $i = 1, 2, \dots, k$ do
        assign to each $S \in C \setminus C'$ the score $sc_{T_{i-1}}(S) = |S \setminus T_{i-1}|$
        let $S_i \in C \setminus C'$ be a set of largest $sc_{T_{i-1}}$ score
        $C' \leftarrow C' \cup \{S_i\}$
        $T_i \leftarrow T_{i-1} \cup S_i$
    end for
    return $C'$
end procedure
```

Recall that $1 - \frac{1}{e} \approx 0.63\dots$.

Theorem 5.12. Greedy returns a $(1 - \frac{1}{e})$-approximation to Max-cover.

Proof. Let $C^* = \{S_1^*, S_2^*, \dots, S_k^*\} \subseteq C$ be an optimal solution.

Also, let $X^* = \bigcup_{i=1}^k S_i^*$ be the set of elements covered by C^*. Then OPT $= |X^*|$.

The algorithm picks sets $S_1, \dots, S_k$. Let s_i be the score of S_i when S_i was picked (note that the score of a set S_i can change during iterations, as more elements are covered). Then $s_i = |S_i \setminus T_{i-1}|$, where $T_{i-1} = \bigcup_{j=1}^{i-1} S_j$.

Also, let $t_i = |T_i| = \bigcup_{j=1}^{i} S_j$.

Let $u_i = |X^*| - t_i$.

Thus, we have defined the following values:

- s_i is the number of "new" elements covered by S_i

- t_i is the total number of elements covered by $S_1, \ldots, S_i$
- u_i is the advantage of the optimal solution over $S_1, \ldots, S_i$

Then, $t_0 = 0$, and $t_k = |T_k|$ is the value of C' (the solution returned by the greedy procedure). Also $u_0 = |X^*| - t_0 = \text{OPT}$.

We want to prove that $t_k \geq (1 - \frac{1}{e})u_0$.

Claim: $s_i \geq \frac{u_{i-1}}{k}$, $\forall i \in \{1, \ldots, k\}$.

Proof of the claim. The set $S = S_i$ chosen by greedy maximizes $|S \setminus T_{i-1}|$. At least one set $S_j^* \in C^*$ must cover at least $\frac{u_{i-1}}{k} = \frac{|X^*| - |T_{i-1}|}{k}$ elements of $X^* \setminus T_{i-1}$.

If it was not the case, it would be impossible for C^* to cover all of $X^* \setminus T_{i-1}$. In fact, the set has cardinality u_{i-1}, and $|C^*| = k$. Also C^* has to cover all of $X^* \setminus T_{i-1}$, since the latter is a subset of X^*.

Then, $\exists S_j^* \in C^*$ s.t. $sc_{T_{i-1}}(S_j^*) = |S_j^* \backslash T_{i-1}| \geq \frac{u_{i-1}}{k}$. Then, $sc_{T_{i-1}}(S_i) \overset{\text{(greedy choice)}}{\geq} sc_{T_{i-1}}(S_j^*) \geq \frac{u_{i-1}}{k}$, and $s_i = sc_{T_{i-1}}(S_i)$. And this proves the claim.

Claim: $u_i \leq (1 - \frac{1}{k})^i |X^*|$, for $i \in \{0, \ldots, k\}$.

By induction on i.

Base case ($i = 0$). $u_0 = |X^*|$.

Inductive case. Assume the claim holds until $i \leq k - 1$; we want to prove it for $i + 1$.

$$\begin{aligned}
u_{i+1} &= |X^*| - t_{i+1} = |X^*| - |T_{i+1}| = |X^*| - |\bigcup_{j=1}^{i+1} S_j| = \\
&= |X^*| - |\bigcup_{j=1}^{i} S_j| - |S_{i+1} \setminus \bigcup_{i=1}^{i} S_j| = |X^*| - |T_i| - |S_{i+1} \setminus T_i| = \\
&= |X^*| - |T_i| - s_{i+1} = u_i - s_{i+1} \overset{\text{previous claim}}{\leq} u_i - \frac{u_i}{k} = (1 - \frac{1}{k})u_i \leq \\
&\leq (1 - \frac{1}{k}) - (1 - \frac{1}{k})^i |X^*| = (1 - \frac{1}{k})^{i+1} |X^*|
\end{aligned}$$

And this proves the claim.

Then, $u_k \leq (1 - \frac{1}{k})^k |X^*| \leq (1 - \frac{1}{e})|X^*| \leq \frac{1}{e}|X^*|$. But, $u_k = |X^*| - |T_k|$, then $|X^*| - |T_k| = u_k \leq \frac{1}{e}|X^*|$, then $(1 - \frac{1}{e})|X^*| \leq |T_k| = t_k$.

□

6 Set functions

6.1 Overview

Given $S \subseteq [n]$, a set function is a function $f : 2^{[n]} \to \mathbb{R}$, where $f(S)$ is the value of f at S.

Set functions can be divided in three kinds.

Submodular functions, $\forall S, T \subseteq [n]$:

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$$

Modular functions, $\forall S, T \subseteq [n]$:

$$f(S) + f(T) = f(S \cup T) + f(S \cap T)$$

Also, a function is modular iff $\exists$ weights $w_1, \dots, w_n, z$ s.t. $f(S) = z + \sum_{i \in S} w_i$.

Supermodular functions, $\forall S, T \subseteq [n]$:

$$f(S) + f(T) \leq f(S \cup T) + f(S \cap T)$$

Let us consider the following set function. Let $C = \{A_1, \dots, A_n\}$, where $A_i \subseteq [t]$. Let our "covering" function be

$$f(S) = |\bigcup_{j \in S} A_j|$$

for $S \subseteq [n]$.

Theorem 6.1. The function f, as just defined, is submodular.

Proof. Observe that $f(S) = \sum_{i \in [t]} [\exists j \in S : i \in A_j] = \sum_{i=1}^{t} f_i(S)$, where

$$f_i(S) = \begin{cases} 1 & \text{if } \exists j \in S \text{ s.t. } i \in A_j \\ 0 & \text{o/w} \end{cases} \tag{40}$$

We claim that, $\forall i \in [t]$, f_i is submodular. To prove it, we need to prove that $f_i(S) + f_i(T) \geq f_i(S \cup T) + f_i(S \cap T)$, $\forall S, T \subseteq [n]$.

Since the range of f_i is (a subset of) $\{0, 1\}$, if $f_i(S \cup T) = f_i(S \cap T) = 0$, then the inequality holds trivially.

Moreover, f_i is monotone increasing: if $S \subseteq T$, then $f(S) \leq f(T)$. If $f_i(S) = 1$, then $\exists j \in S$ s.t. $i \in A_j$. Since $j \in S \subseteq T$, then it must be that $f_i(T) = 1$.

Thus we only need to consider two cases:

a. $f_i(S \cup T) = f_i(S \cap T) = 1$

b. $f_i(S \cap T) = 0$ and $f_i(S \cup T) = 1$

If a. holds, then $\exists j \in S \cap T$ s.t. $i \in A_j$, but then $j \in S$ and $j \in T$, thus $f_i(S) = f_i(T) = 1$

if b. holds, then $\exists j \in S \cup T$ s.t. $i \in A_j$, thus $f_i(S \cup T) = 1$. Either $j \in S$ or $j \in T$, and not both. Thus $f_i(S) + f_i(T) = 1$.

Thus, we have proved that $\forall i \in [t]$, f_i is submodular. To finish the proof we have to show that the sum of submodular functions is submodular.

$f(S) = \sum_{i=1}^{t} f_i(S) \cdot$

Let $S, T \subseteq [n]$, then

$$\begin{aligned} f(S) + f(T) = \sum_{i=1}^{t} (f_i(S) + f_i(T)) &\overset{f_i \text{ submodular}}{\geq} \\ &\geq \sum_{i=1}^{t} (f_i(S \cup T) + f_i(S \cap T)) = \\ &= \sum_{i=1}^{t} f_i(S \cup T) + \sum_{i=1}^{t} f_i(S \cap T) = \\ &= f(S \cup T) + f(S \cap T) \end{aligned}$$

□

6.2 Diminishing return

We now study Max-cover as being a special case of submodular functions.

Submodular functions are also studied in economics, where they are given a different name.

Definition 6.1 (Diminishing return)**.** Given $S \subseteq [n]$ and $x \in [n] \setminus S$, we define the return of x at S to be

$$\Delta_f(x \mid S) = f(S \cup \{x\}) - f(S)$$

That is, how much do I gain by adding x to S.

A function f has the diminishing return property if, $\forall S \subseteq T$, $\forall x \in [n] \setminus T$:

$$\Delta_f(x \mid S) \geq \Delta_f(x \mid T)$$

Lemma 6.2. If f is submodular then f has the diminishing return property.

Proof. We'd like to prove that $\forall A \subseteq B$, $\forall x \in [n] \setminus B$, $\Delta_f(x \mid A) \geq \Delta_f(x \mid B)$.

Let us define $S = A \cup \{x\}$, $T = B$.

By submodularity of f, $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$. Thus $f(A \cup \{x\}) + f(B) \geq f(A \cup \{x\} \cup B) + f((A \cup \{x\}) \cap B)$.

$f(A \cup \{x\}) + f(B) \geq f(B \cup \{x\}) + f(A)$

$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$

Then, $\Delta_f(x \mid A) \geq \Delta_f(x \mid B)$. □

Exercise 6.2.1. The converse of previous lemma also holds. Prove that if f has the diminishing return property, then f is submodular.

We aim to maximize a submodular f under a "cardinality constraint". In Max-cover, we wanted to maximize $f(S)$ under $|S| = k$.

In algorithm 13

Algorithm 13 Submodular functions framework for Max-cover

```
procedure GREEDY$_f(X, k)$
    $S_0 \leftarrow \emptyset$
    for $i = 0, 1, \dots, k-1$ do
        let $x_{i+1} \in X$ be a maximizer of $f(S_i \cup \{x_{i+1}\})$
        $S_{i+1} \leftarrow S_i \cup \{x_{i+1}\}$
    end for
    return $S_k$
end procedure
```

Theorem 6.3. If f is submodular, monotone and non-negative, then GREEDY$_f(X, k)$ returns a $(1 - \frac{1}{e})$-approximation to $\max_{S \in \binom{X}{k}} f(S)$.

Proof. Let $S^* = \{X_1^*, \dots, X_k^*\}$ be an optimal solution.

Then $f(S^*) = \max_{S \in \binom{X}{k}} f(S)$.

$\forall i = 0, \dots, k-1$:

$$
\begin{aligned}
f(S^*) \overset{\text{monotonicity}}{\leq} \; & f(S^* \cup S_i) = \\
& = f(\{X_1^*, \dots, X_k^*\} \cup S_i) - f(\{X_1^*, \dots, X_{k-1}^*\} \cup S_i) + f(\{X_1^*, \dots, X_{k-1}^*\} \cup S_i) - \\
& - f(\{X_1^*, \dots, X_{k-2}^*\} \cup S_i) + f(\{X_1^*, \dots, X_{k-2}^*\} \cup S_i) - \dots - f(\{X_1^*, X_2^*\} \cup S_i) + \\
& + f(\{X_1^*, X_2^*\} \cup S_i) - f(X_1^* \cup S_i) + f(X_1^* \cup S_i) - f(S_i) + f(S_i) = \\
& = \Delta_f(X_k^* \mid S_i \cup \{X_1^*, \dots, X_{k-1}^*\}) + \\
& + \Delta_f(X_{k-1}^* \mid S_i \cup \{X_1^*, \dots, X_{k-2}^*\}) + \\
& \vdots \\
& + \Delta_f(X_2^* \mid S_i \cup \{X_1^*\}) + \\
& + \Delta_f(X_1^* \mid S_i) + \\
& + f(S_i) = \\
& = f(S_i) + \sum_{j=1}^{k} \Delta_f(X_j^* \mid S_i \cup \{X_1^*, \dots, X_{j-1}^*\}) \overset{\text{diminish. return}}{\leq} \\
& \leq f(S_i) + \sum_{j=1}^{k} \Delta_f(X_j^* \mid S_i) \overset{\text{greedy choice}}{\leq} \\
& \leq f(S_i) + \sum_{j=1}^{k} \Delta_f(X_{i+1} \mid S_i) = f(S_i) + k \cdot \Delta_f(X_{i+1} \mid S_i) = \\
& = f(S_i) + k(f(S_i \cup \{X_{i+1}\}) - f(S_i))
\end{aligned}
$$

Then, $f(S^*) \leq f(S_i) + k(f(S_i \cup \{X_{i+1}\}) - f(S_i))$.

Let us define $\delta_i = f(S^*) - f(S_i)$.

$f(S^* - f(S_i)) \leq k(f(S_{i+1}) - f(S_i))$

$\delta_i \leq k(-f(S^*) + f(S_{i+1}) - f(S_i) + f(S^*))$

$\delta_i \leq k(\delta_i - \delta_{i+1})$

$k\delta_{i+1} \leq (k-1)\delta$

Then, $\forall i = 0, \dots, k-1$, $\delta_{i+1} \leq (1 - \frac{1}{k})\delta_i$.

Finally,

$$\begin{aligned}
&\delta_0 = f(S^*) - f(S_0) = f(S^*) - f(\emptyset) \overset{\text{non-neg.}}{\leq} f(S^*) \\
&\delta_1 \leq (1 - \frac{1}{k})\delta_0 \leq (1 - \frac{1}{k})f(S^*) \\
&\delta_2 \leq (1 - \frac{1}{k})\delta_1 \leq (1 - \frac{1}{k})^2 f(S^*) \\
&\vdots \\
&\delta_k \leq f(S^*) - f(S_k) \leq (1 - \frac{1}{k})^k f(S^*) \leq \frac{1}{e} f(S^*)
\end{aligned}$$

Then, $f(S_k) \geq (1 - \frac{1}{e})f(S^*)$. $\square$

Let us now see some problems on submodular functions. Let $f : 2^{[n]} \to \mathbb{R}$ be submodular.

Problem: assuming that f is monotone increasing and submodular, then

$$\max_{S \subseteq [n] : |S| \leq k} f(S)$$

Can be approximated to $1 - \frac{1}{e}$.

Problem: assuming that f is submodular, then

$$\min_{S \subseteq [n]} f(S)$$

Can be solved exactly in polytime.

Problem: assuming that f is submodular, then

$$\max_{S \subseteq [n]} f(S)$$

Is NP-Hard.

7 Oracle models

See 10.5 for an explanation of oracles.

Let us see an example. We have a database that assigns some value v_i to each key $i \in \{0, \dots, 2^n - 1\}$. You can call an oracle, sending it some $i \in \{0, \dots, 2^n - 1\}$, and getting back v_i.

How much time do you need to find the maximum v_i? Or equivalently, how many "calls/queries" do you need to make to the oracle to find the maximum key?

Suppose you have the following oracle. Given $\{i, j\} \in \binom{[n]}{2}$, you get "$v_i < v_j$" or "$v_j < v_i$". Use the oracle to sort the v_i's.

We know that this problem can be solved with $O(n \log n)$ queries (heapsort, mergesort, ...). Is it also $\Omega(n \log n)$?

If $f(n) \in \Omega(g(n))$, then $\lim \frac{f(n)}{g(n)} \geq c$. There are $n!$ possible orderings/permutations. Therefore, one needs $\log_2 n!$ bits to represent an ordering.

$$n \log_2 n = \log_2 n^n \geq \log_2 n! \log_2 (\frac{n}{2})^{\frac{n}{2}} = \frac{n}{2} \log_2 \frac{n}{2}$$

And $\log_2 n! = \Theta(n \log n)$ (if $f(n) \in \Theta(g(n))$, then $c \leq \lim \frac{f(n)}{g(n)} \leq c'$).

Given that I need $\Theta(n \log n)$ bits, and that each oracle call gives me $\Theta(1)$ bits, I need to call the oracle at least $\Omega(n \log n)$ times.

7.1 Unconstrained optimization of SM functions

Problem: for f submodular,

$$\min_{S \subseteq [n]} f(S)$$

It can be used on covering functions, or on the following problem. Let $G(V, E)$ be a graph, and let $c : 2^V \to \mathbb{R}$ be defines as $c(S) = |Cut(S, V \setminus S)|$. Then, $\min_{S \subseteq [n]} c(S)$ is Min-Cut (solvable in polytime).

Lemma 7.1. $c(S)$ is submodular.

Proof. Let $S, T \subseteq V$ be given. We want to prove $c(S)+c(T) \geq c(S \cup T)+c(S \cap T)$.

Let us define the following sets. $A_{00} = V \setminus (S \cup T)$, $A_{01} = T \setminus S$, $A_{10} = S \setminus T$, $A_{11} = S \cap T$.

Then the sets

$$\begin{gathered} Cut(A_{00}, A_{01}), Cut(A_{00}, A_{10}), Cut(A_{00}, A_{11}) \\ Cut(A_{01}, A_{10}), Cut(A_{01}, A_{11}), Cut(A_{10}, A_{11}) \end{gathered}$$

are pairwise disjoint.

Then,

$$\begin{aligned} Cut(S, V \setminus S) &= Cut(A_{11}, A_{01}) \cup Cut(A_{11}, A_{00}) \cup Cut(A_{10}, A_{01}) \cup Cut(A_{10}, A_{00}) = \\ &= \bigcup_{i \in B} \bigcup_{j \in \{0,1\}} Cut(A_{1i}, A_{0j}) \end{aligned}$$

$$\begin{aligned} c(S) = |Cut(S, V \setminus S)| &= \sum_{i \in \{0,1\}} \sum_{j \in \{0,1\}} |Cut(A_{1i}, A_{0j})| = \\ &= |Cut(A_{11}, A_{01})| + |Cut(A_{11}, A_{00})| + |Cut(A_{10}, A_{01})| + |Cut(A_{10}, A_{00})| \end{aligned}$$

$$c(T) = |Cut(T, V \setminus T)| = \sum_{i\in\{0,1\}} \sum_{j\in\{0,1\}} |Cut(A_{i1}, A_{j0})| =$$
$$= |Cut(A_{11}, A_{10})| + |Cut(A_{11}, A_{00})| + |Cut(A_{10}, A_{01})| + |Cut(A_{01}, A_{00})|$$

$$c(S \cap T) = |Cut(A_{11}, A_{00})| + |Cut(A_{11}, A_{01})| + |Cut(A_{11}, A_{10})|$$

$$c(S \cup T) = |Cut(A_{00}, A_{01})| + |Cut(A_{00}, A_{10})| + |Cut(A_{00}, A_{11})|$$

$$\begin{aligned} c(S \cap T) + c(S \cup T) = 2|Cut(A_{11}, A_{00})| + |Cut(A_{11}, A_{01})| + |Cut(A_{11}, A_{10})| + \\ + |Cut(A_{00}, A_{01})| + |Cut(A_{00}, A_{10})| \end{aligned}$$

$$\begin{aligned} c(S) + c(T) = 2|Cut(A_{11}, A_{00})| &+ 2|Cut(A_{01}, A_{10})| + |Cut(A_{11}, A_{01})| + \\ &+ |Cut(A_{11}, A_{10})| + |Cut(A_{00}, A_{01})| + \\ &+ |Cut(A_{00}, A_{10})| \end{aligned}$$

Thus, $(c(S) + c(T)) - (c(S \cup T) + c(S \cap T)) = 2|Cut(A_{01}, A_{10})| \geq 0$. ☐

Observation 7.1.1. c in not in general monotone.

Observation 7.1.2. c is "symmetric", $c(S) = c(V \setminus S)$, $\forall S \subseteq V$.

Proof. $c(S) = |Cut(S, V \setminus S)| = c(V \setminus S)$. ☐

Then, Max-Cut is just $\max_{S\subseteq V} c(S)$ for a function c that is submodular, symmetric and non-negative.

Theorem 7.2 (Feige, Mirrokni, Vondrák, '07)**.** *If $f : 2^{[n]} \to \mathbb{R}$ is submodular, non-negative and symmetric, then $\max_S f(S)$ can be 2-approximated in polytime, in the oracle model.*

Lemma 7.3. *Let $g : 2^{[n]} \to \mathbb{R}$ be a submodular function. Then, $\forall S \subseteq [n]$,*

$$\operatorname*{avg}_{T\subseteq S} g(T) = 2^{-|S|} \cdot \sum_{T\subseteq S} g(T) \geq \frac{g(\emptyset) + g(S)}{2}$$

Proof. We prove the lemma by induction on $|S|$.

Base case ($|S| = 0$). Trivial, since the LHS reduces to $2^{-0} \cdot g(\emptyset) = g(\emptyset)$, the the RHS $\frac{g(\emptyset)+g(\emptyset)}{2} = g(\emptyset)$.

Suppose the claim holds for each set of cardinality i. Let S be a set s.t. $|S| = i + 1$. Let $x \in S$, and let $S' = S \setminus \{x\}$. Then $|S'| = i$.

$$\begin{aligned}\sum_{T\subseteq S} g(T) &= \sum_{T'\subseteq S'} g(T') + \sum_{T'\subseteq S'} g(T'\cup\{x\}) = \\ &= 2\sum_{T'\subseteq S'} g(T') + \sum_{T'\subseteq S'} (g(T'\cup\{x\}) - g(T')) = \\ &= 2\sum_{T'\subseteq S'} g(T') + \sum_{T\subseteq S : x\in T} (g(T) - g(T'\cap S')) \geq \\ &\geq 2\sum_{T'\subseteq S'} g(T') + \sum_{T\subseteq S : x\in T} (g(T\cup S') - g(S')) = \\ &= 2\sum_{T'\subseteq S'} g(T') + \sum_{T\subseteq S : x\in T} (g(S) - g(S')) = \\ &= 2\sum_{T'\subseteq S'} g(T') + 2^i(g(S) - g(S'))\end{aligned}$$

So,

$$2^{-|S|}\sum_{T\subseteq S} g(T) \geq 2^{1-|S|}\sum_{T'\subseteq S'} g(T') + 2^{i-|S|}(g(S) - g(S'))$$

And

$$\begin{aligned}\operatorname*{avg}_{T\subseteq S} g(T) &\geq 2^{-i}\sum_{T'\subseteq S'} g(T') + 2^{-1}(g(S) - g(S')) = \\ &= \operatorname*{avg}_{T'\subseteq S'} g(T') + \frac{g(S) - g(S')}{2} \overset{\text{IH}}{\geq} \\ &\geq \frac{g(\emptyset) + g(S')}{2}' \frac{g(S) - g(S')}{2} = \frac{g(S) + g(\emptyset)}{2}\end{aligned}$$

□

Lemma 7.4. Let $g : 2^{[n]} \to \mathbb{R}$ be submodular, then $\forall S_1, S_2 \subseteq [n]$

$$\operatorname*{avg}_{T_1\subseteq S_1}\ \operatorname*{avg}_{T_2\subseteq S_2} g(T_1\cup T_2) \geq \frac{g(\emptyset) + g(S_1) + g(S_2) + g(S_1\cup S_2)}{4}$$

Proof. Given $X \subseteq [n]$, we define $h_X(T) = g(X \cup T)$. Then, h_X is submodular, because g is submodular (h_X is a sort of projection of g). In fact,

$$\begin{aligned}h_X(A) + h_X(B) = g(X\cup A) + g(X\cup B) &\overset{\text{submodularity}}{\geq} \\ &\geq g(X\cup A\cup B) + g((X\cup A)\cap(X\cup B)) = \\ &= g(X\cup A\cup B) + g(X\cup(A\cap B)) = h_X(A\cup B) + h_X(A\cap B)\end{aligned}$$

By Lemma 7.3,

$$\operatorname*{avg}_{T\subseteq S} h_X(T) \geq \frac{h_X(\emptyset) + h_X(S)}{2} = \frac{g(X) + g(X\cup S)}{2}$$

Then,

$$\begin{aligned}
\operatorname*{avg}_{T_1 \subseteq S_1} \operatorname*{avg}_{T_2 \subseteq S_2} g(T_1 \cup T_2) &= \operatorname*{avg}_{T_1 \subseteq S_1} \left(\operatorname*{avg}_{T_2 \subseteq S_2} g(T_1 \cup T_2) \right) = \\
&= \operatorname*{avg}_{T_1 \subseteq S_1} \left(\operatorname*{avg}_{T_2 \subseteq S_2} h_{T_1}(T_2) \right) \geq \\
&\geq \operatorname*{avg}_{T_1 \subseteq S_1} \frac{g(T_1) + g(T_1 \cup S_2)}{2} = \\
&= \frac{1}{2} \left(\operatorname*{avg}_{T_1 \subseteq S_1} g(T_1) + \operatorname*{avg}_{T_1 \subseteq S_1} g(T_1 \cup S_2) \right) = \\
&= \frac{1}{2} \left(\operatorname*{avg}_{T_1 \subseteq S_1} g(T_1) + \operatorname*{avg}_{T_1 \subseteq S_1} h_{S_2}(T_1) \right) \geq \\
&\geq \frac{1}{2} \left(\frac{g(\emptyset) + g(S_1)}{2} + \frac{g(S_2) + g(S_1 \cup S_2)}{2} \right) = \\
&= \frac{g(\emptyset) + g(S_1) + g(S_2) + g(S_1 \cup S_2)}{4}
\end{aligned}$$

□

Theorem 7.5. Let $f : 2^{[n]} \to \mathbb{R}$ be submodular, symmetric, non-negative. Then, if R is a UAR subset of the groundset $[n]$, then

$$\mathbb{E}[f(R)] \geq \frac{1}{2} \max_{S \subseteq [n]} f(S)$$

Where R is sampled as in algorithm 14

Algorithm 14 Sample UAR $R \subseteq [n]$

```
 $R \leftarrow \emptyset$
for $i = 1, \dots, n$ do
    flip an iid coin $c_i$
    if $c_i$ is heads then
        $R \leftarrow R \cup \{i\}$
    end if
end for
 return $R$
```

Note that this algorithm is basically the same algorithm for 2-approximating Max-Cut (see algorithm 1).

It is impossible for a polytime algorithm to return a set R that, in general, is a $(0.5 + \varepsilon)$-approximation.

Proof. Let S^* be an optimal solution, then

$$f(S^*) = \max_{S \subseteq [n]} f(S)$$

We apply Lemma 7.4, with $S_1 = S^*$ and $S_2 = [n] \setminus S^*$.

$$\frac{f(\emptyset)+f(S^*)+f([n]\setminus S^*)+f([n])}{4} \overset{\text{Lemma 7.4}}{\leq}$$
$$\leq 2^{-|S^*|}\cdot 2^{-n+|S^*|}\cdot \sum_{T_1\subseteq S_1}\sum_{T_2\subseteq S_2} f(T_1\cup T_2) =$$
$$= 2^{-n}\sum_{T_1\subseteq S_1}\sum_{T_2\subseteq S_2} f(T_1\cup T_2) \overset{S_1\cap S_2=\emptyset\wedge S_1\cup S_2=[n]}{=}$$
$$= 2^{-n}\sum_{T\subseteq [n]} f(T) = \mathbb{E}[f(R)]$$

Then,

$$\mathbb{E}[f(R)] \geq \frac{f(\emptyset)+f(S^*)+f([n]\setminus S^*)+f([n])}{4} \overset{\text{non-negative}}{\geq}$$
$$\geq \frac{f(S^*)+f([n]\setminus S^*)}{4} \overset{\text{symmetric}}{=} \frac{2\cdot f(S^*)}{4} = \frac{1}{2}f(S^*)$$

□

Theorem 7.6. Let $f : 2^{[n]} \to \mathbb{R}$ be submodular, non-negative and not symmetric. Then, if R is a UAR subset of the groundset $[n]$, then

$$\mathbb{E}[f(R)] \geq \frac{1}{4}\max_{S\subseteq[n]} f(S)$$

Proof. The proof is exactly like the proof of Theorem 7.5, except for the last passage:

$$\mathbb{E}[f(R)] \geq \frac{f(\emptyset)+f(S^*)+f([n]\setminus S^*)+f([n])}{4} \geq$$
$$\geq \frac{f(S^*)+f([n]\setminus S^*)}{4} \geq \frac{f(S^*)}{4}$$

□

8 Clustering problems

Given a set X of points, a metric d over X and an integer $k \geq 1$. We want to find a set $C \in \binom{X}{k}$, of "centers", s.t. some function is minimized.

Depending on the kind of function that we want to minimize, we have a different clustering problem. We are now going to see some of the most important ones.

K-Means is defined over an ℓ_2-loss. The function to minimize is:

$$\sum_{x\in X}\min_{c\in C} d^2(x,c)$$

K-Median is defined over an ℓ_1-loss. The function to minimize is:

$$\sum_{x\in X}\min_{c\in C} d(x,c)$$

K-Center is defined over an ℓ_∞-loss. The function to minimize is:

$$\max_{x \in X} \min_{c \in C} d(x, c)$$

In general, one can define the clustering with any ℓ_p-loss.

K-Means is often defined as Lloyd's algorithm, which is an error. Lloyd's algorithm has been proposed as an heuristic for solving K-Means. It is quite simple to understand, but in general it can have a bad approximation, and a bad running time.

8.1 K-Center

We are going to study K-Center, which, among the three clustering problems defined above, is the one with the simplest approximation. In Algorithm 15 a greedy approximation for K-Center.

Recall that, for a point x, and a set of points S, we have defined

$$d(x, S) = \min_{s \in S} d(x, s)$$

Algorithm 15 Greedy for K-Center

```
procedure GREEDY($X, d, k$)
    if $|X| \leq k$ then
        return $X$
    end if
    pick arbitrarily $x_1 \in X$
    $c_1 \leftarrow \{x_1\}$
    for $i = 2, \ldots, k$ do
        let $\Delta_{i-1} = \max_{x \in X} d(x, C_{i-1})$
        let $x_i \in \text{argmax}_{x \in X} d(x, C_{i-1})$
        $C_i \leftarrow C_{i-1} \cup \{x_i\}$
    end for
    return $C_k$
end procedure
```

Theorem 8.1. Greedy returns a 2-approximation for K-Center.

Proof. The algorithm defines sets $C_1, \ldots, C_k$ and $x_1, \ldots, x_k$, s.t. for each i, $C_i = \{x_1, \ldots, x_i\}$. It also defines the distances $\Delta_1, \ldots, \Delta_{k-1}$.

Let us define:

$$\Delta_k = \max_{x \in X} d(x, C_k)$$
$$x_{k+1} \in \underset{x \in X}{\text{argmax}}\, d(x, C_k)$$
$$C_{k+1} = C_k \cup \{x_{k+1}\}$$

Note that we are simulating an extra iteration of the for loop.

Claim 8.1.1. C_k, the set returned by GREEDY, acts as a K-Center solution of value Δ_k

Proof. The value of C_k is

$$C_k = \max_{x\in X} \min_{y\in C_k} d(x,y) = \max_{x\in X} d(x, C_k) = \Delta_k$$

□

Claim 8.1.2. $\Delta_1 \geq \Delta_2 \geq \cdots \geq \Delta_k$ (monotone decreasing)

Proof. Suppose $S \subseteq T \subseteq X$, and $x \in X$. Then

$$d(x,S) = \min_{y\in S} d(x,y) \overset{S\subseteq T}{\geq} \min_{y\in T} d(x,y) = d(x,T)$$

For each $i = 2, \dots, k$, it holds that

$$\Delta_{i-1} = d(x_i, C_{i-1}) \overset{\text{greedy}}{\geq} d(x_{i+1}, C_{i-1}) \overset{C_{i-1}\subseteq C_i}{\geq} d(x_{i+1}, C_i) = \Delta_i$$

□

Claim 8.1.3. $\forall i = 2, \dots, k+1$, $\forall \{x_j, x_{j'}\} \in \binom{C_i}{2}$, $d(x_j, x_{j'}) \geq \Delta_{i-1}$.

Proof. By induction on i.

Base case ($i = 2$). x_2 is at distance Δ_1 from x_1, and thus from $C_1 = \{x_1\}$.

Inductive case. Suppose the claim holds for $i \leq k$; we prove it for $i+1$. Since the claim holds at i, we have $\forall \{x_j, x_{j'}\} \in \binom{C_i}{2}$, $d(x_j, x_{j'}) \geq \Delta_{i-1}$.

The generic pair $\{x_j, x_{j'}\} \in \binom{C_{i+1}}{2}$ either contains two points from C_i, or it contains x_{i+1} and one point from C_i. Since $\Delta_{i-1} \geq \Delta_i$, w.m.a. that one point is x_{i+1}. Wlog, $x_{j'} = x_{i+1}$. We have $\Delta_i = \max_{x\in X} d(x, C_i) = d(x_{i+1}, C_i) = \min_{y\in C_i} d(x_{i+1}, y)$.

Thus, $\forall x_j \in C_i$, $d(x_{i+1}, x_j) \geq \Delta_i$. Then, $\forall \{x_j, x_{j'}\} \subseteq C_{i+1}$ s.t. $x_{i+1} \in \{x_j, x_{j'}\}$, $d(x_j, x_{j'}) \geq \Delta_i$. □

Claim 8.1.4. If C is any solution, that is, $C \in \binom{X}{k}$, then

$$\max_{x\in X} \min_{y\in C} d(x,y) \geq \frac{\Delta_k}{2}$$

Proof. Let $V = \max_{x\in X} \min_{y\in C} d(x,y)$ be the value of the solution C. Then, $\forall x \in X$, $\exists y = y_x \in C$ s.t. $d(x, y_x) \leq V$.

Consider the set $C_{k+1} = C_k \cup \{x_{k+1}\}$, then $|C_{k+1}| = k+1$. By the PigeonHole Principle (PHP), $\exists \{x_j, x_{j'}\} \in \binom{C_{k+1}}{2}$ s.t. $y_{x_j} = y_{x_{j'}}$.

Let $x_j, x_{j'}$ be two points of C_{k+1}, $x_j \neq x_{j'}$, that have the same $y \triangleq y_{x_j} = y_{x_{j'}} \in C$.

Then, $d(x_j, y) \leq V$, and $d(x_{j'}, y) \leq V$. By the triangle inequality $d(x_j, x_{j'}) \leq d(x_j, y) + d(y, x_{j'}) \leq 2V$.

By Claim 8.1.3, any two distinct points from C_{k+1} are at distance at least Δ_k from one another. Thurs,

$$\Delta_k \leq d(x_j, x_{j'}) \leq 2V \to V \geq \frac{\Delta_k}{2}$$

□

Claims 8.1.1 and 8.1.4 entail that Greedy returns a 2-approximation. □

Theorem 8.2. $\forall \varepsilon > 0$, K-Center is NP-Hard to approximate to $2 - \varepsilon$.

8.2 Dominating set

Input: undirected graph $G(V, E)$, $k \geq 1$

Question: does there exist $S \in \binom{V}{k}$ s.t. $\forall v \in V$, $v \in S$ or $\exists w \in S$ s.t. $\{v, w\} \in E$

That is, a dominating set is one that either contains all the vertices, or it touches all the edges.

Theorem 8.3. The Dominating set is NP-Hard

Let us define the cost of a K-Center solution S as

$$Cen(S) = \max_{v \in V} \min_{c \in S} d(v, c)$$

Theorem 8.4. It is NP-Hard to approximate K-Center to α, $\forall \alpha < 2$.

Proof. Let $G(V, E), k$ be an instance of Dominating Set.

We construct a metric $d : V \times V \to \mathbb{R}$ as follows. For the set of points V:

- $d(v, w) = 1$ if $\{v, w\} \in E$
- $d(v, w) = 2$ if $\{v, w\} \in \binom{V}{2} \setminus E$
- $d(v, v) = 0$ if $v \in V$

Claim 8.4.1. Suppose that S is a set of points s.t. $Cen(S) = 1$. Then S is a dominating set of $G(V, E)$.

Proof. If S is s.t. $Cen(S) = 1$, then $\forall v \in V$, $\exists w \in S$ s.t. either $v = w$ or $\{v, w\} \in E$. Therefore, S is a dominating set of $G(V, E)$. □

Thus, if $\exists$ dominating set S of k nodes, then $Cen(S) = 1$.

Suppose that, instead, $\nexists S \in \binom{V}{k}$ s.t. S is a dominating set of $G(V, E)$. Then, no matter which k nodes I pick, there will be one node v of the graph that is not covered by the k nodes: $\forall S \in \binom{V}{k}$, $\exists v \in V \setminus S$ s.t. $\forall w \in S : \{v, w\} \notin E$. Then, $d(v, S) = 2$.

Thus, the inapproximability has been proved. We only need to prove that d is a metric.

Let $\{v, w, x\} \in \binom{V}{3}$. The triangle inequality requires that $d(v, w) + d(w, x) \geq d(v, x)$. We have that

$$d(v, w) + d(w, x) \geq 2 \geq d(v, x)$$

$\square$

Without the triangle inequality, we could have set

$$d(v, w) = T \gg 2$$

for $\{v, w\} \in \binom{V}{2} \setminus E$. The inapproximability ratio would have been $\geq T$.

8.3 K-Median

Given a solution S for K-Median, we define the cost of S as

$$Med(S) = \sum_{v \in V} \min_{c \in S} d(v, c)$$

We define the problem as follows. Given as input a metric d, V and k. We want to find $S \in \binom{V}{k}$ that (approximately) minimizes $Med(S)$.

Observation 8.4.1. K-Median admits a $(2n)$-approximation, where $n = |V|$.

Proof. $\forall S \in \binom{V}{k}$,

$$Med(S) = \sum_{v \in V} \min_{c \in S} d(v, c)$$

$$Cen(S) = \max_{v \in V} \min_{c \in S} d(v, c)$$

Also,

$$\begin{aligned} Cen(S) \leq Med(S) = \sum_{v \in V} \min_{c \in S} d(v, c) \leq \\ \leq |V| \max_{v \in V} \min_{c \in S} d(v, c) = |V| Cen(S) \end{aligned}$$

Suppose that S' is a 2-approximation for K-Center. Then,

$$Cen(S') \leq 2 \min_{S \in \binom{V}{k}} Cen(S)$$

So,

$$Med(S') \leq n \cdot Cen(S') \leq 2n \min_{S \in \binom{V}{k}} Cen(S) \leq 2n \min_{S \in \binom{V}{k}} Med(S)$$

$\square$

Algorithm 16 Arya et al., '03

```
procedure LocalSearch($V, d, k$)
    pick any $S \in \binom{V}{k}$
    while $\exists o \in S, i \in V \setminus S$ s.t. $Med(S \setminus \{o\} \cup \{i\}) < Med(S)$ do
        $S \leftarrow S \setminus \{o\} \cup \{i\}$
    end while
    return $S$
end procedure
```

Let us now see an algorithm for approximating K-Median, presented in Algorithm 16.

Questions:

1. For how many iterations does the algorithm run?
2. Once I get to a local minimum, how close will it be to a global minimum?

Let $L = \{l_1, \dots, l_k\}$ be the solution returned by LocalSearch.

Let $Q = \{q_1, \dots, q_k\}$ be an optimal solution.

Let $\pi : Q \to L$ be a function that maps each optimal center to one of its closest LocalSearch center.

Let $D_j^* = \min_{q \in Q} d(j, q)$, $\forall j \in V$. D_j^* is the cost paid by the optimal solution for point j.

Let $D_j = \min_{l \in L} d(j, l)$, $\forall j \in V$. D_j is the cost paid by the LS solution, for point j.

Let $L_1, L_2, \dots, L_k$ be the partition of V induced by the centers of L. That is, $\forall l \in L$, $j \in L_i \to d(j, l_i) \leq d(j, l)$.

Let $Q_1, Q_2, \dots, Q_k$ be the partition of V induced be the centers of Q. That is, $\forall q \in Q$, $j \in Q_i \to d(j, q_i) \leq d(j, q)$.

$$\text{OPT} = \sum_{j \in V} D_j^* = \sum_{j \in V} \min_{q \in Q} d(j, q) = \sum_{i=1}^{k} \sum_{j \in Q_i} d(j, q_i)$$

$$\text{ALG} = \sum_{j \in V} D_j = \sum_{j \in V} \min_{l \in L} d(j, l) = \sum_{i=1}^{k} \sum_{j \in L_i} d(j, l_i)$$

Our goal is to bound $\frac{\text{ALG}}{\text{OPT}}$.

We first consider the special case where π is a bijection. If π is a bijection, we can assume, wlog, that $\pi(q_i) = l_i$, $\forall i \in \{1, \dots, k\}$.

Lemma 8.5. Let L be the set of points returned by LS. Then, $\forall i \in [k]$,

$$0 \leq Med(L \setminus \{l_i\} \cup \{q_i\}) - Med(L) \leq \sum_{j \in Q_i} (D_j^* - D_j) + 2 \sum_{j \in L_i} D_j^*$$

Proof. Let us define $L' = L \setminus \{l_i\} \cup \{q_i\}$.

By the LS stopping condition, $Med(L') \geq Med(L)$, thus $Med(L') - Med(L) \geq 0$.

So,

$$Med(L') = \sum_{j \in V} d(j, L') = \sum_{j \in V} \min_{l \in L'} d(j, l)$$

We now bound $\sum_{j \in V} d(j, L')$, term-by-term. Consider three cases:

1. $j \in Q_i$
2. $j \in L_i \setminus Q_i$
3. $j \in V \setminus (L_i \cup Q_i)$

Case 1. If $j \in Q_i$, we could serve j with $q_i \in L'$. Thus,

$$d(j, L') = \min_{q \in L'} d(j, q) \overset{q_i \in L'}{\leq} d(j, q_i) = D_j^*$$

Case 2. If $j \in L_i \setminus Q_i$, then j is not served by q_i in the optimal solution we chose. Say that the optimal solution serves j with $q_r \in Q$. Then, $q_r \neq q_i$. Also, recall that $\pi(q_r) = l_r \neq l_i$, since we have assumed that π is a bijection. Thus $\pi(q_r) \in L'$.

$$\begin{aligned} d(j, L') &\overset{\pi(q_r) \in L'}{\leq} d(j, \pi(q_r)) = \\ &= d(j, l_r) \overset{\text{tr. ineq.}}{\leq} \\ &\leq d(j, q_r) + d(q_r, l_r) = \\ &= D_j^* + d(q_r, l_r) \overset{\substack{\pi(q_r) = lr \\ \text{the closest center}}}{\leq} \\ &\leq D_j^* + d(q_r, l_i) \leq \\ &\leq D_j^* + d(q_r, j) + d(j, l_i) = \\ &= D_j^* + D_j^* + D_j = \\ &= 2D_j^* + D_j \end{aligned}$$

Case 3. If $j \in V \setminus (L_i \cup Q_i)$, then j can be served by its closest center in the LS solution, thus

$$d(j, L') \leq D_j$$

Thus,

$$\begin{aligned} Med(L') &= \sum_{j \in V} d(j, L') = \\ &= \sum_{j \in Q_i} d(j, L') + \sum_{j \in L_i \setminus Q_i} d(j, L') + \sum_{j \in V \setminus (L_i \cup Q_i)} d(j, L') \leq \\ &\leq \sum_{j \in Q_i} D_j^* + \sum_{j \in L_i \setminus Q_i} (2D_j^* + D_j) + \sum_{j \in V \setminus (L_i \cup Q_i)} D_j \end{aligned}$$

And

$$Med(L) = \sum_{j \in V} D_j$$

$$\begin{aligned}
Med(L') - Med(L) &\leq \sum_{j \in Q_i} (D_j^* - D_j) + \sum_{j \in L_i \setminus Q_i} (2D_j^* + D_j - D_j) + \sum_{j \in V \setminus (L_i \cup Q_i)} (D_j - D_j) = \\
&= \sum_{j \in Q_i} (D_j^* - D_j) + 2 \sum_{j \in L_i \setminus Q_i} D_j^* \leq \\
&\leq \sum_{j \in Q_i} (D_j^* - D_j) + 2 \sum_{j \in L_i} D_j^*
\end{aligned}$$

□

Theorem 8.6. If π is a bijection, the LS solutions is a 3-approximation to K-Median.

Proof. We use Lemma 8.5 to sum up the k bounds ($i = 1, 2, \ldots, k$):

$$\begin{aligned}
0 &\leq \sum_{i=1}^{k} Med(L \setminus \{l_i\} \cup \{q_i\}) - k \cdot Med(L) \leq \\
&\leq \sum_{i=1}^{k} \sum_{j \in Q_i} (D_j^* - D_j) + 2 \sum_{i=1}^{k} \sum_{j \in L_i} D_j^* = \\
&= \sum_{j \in V} (D_j^* - D_j) + 2 \sum_{j \in V} D_j^* = \\
&= 3 \sum_{j \in V} D_j^* - \sum_{j \in V} D_j = \\
&= 3\mathrm{OPT} - \mathrm{ALG}
\end{aligned}$$

Then, ALG $\leq$ 3OPT. □

The case just studied, where π is a bijection, although quite common in real life, is not true in general. Let us now consider the case in which π is not a bijection; that is, where multiple q_i's get mapped to the same l_j.

Note that now:

- some q_i's end up being mapped to the same l_j
- some q_i are still mapped one-to-one to some l_j (as in the bijective case)
- some l_i's end up not being mapped at all

Let $T \subseteq L$ be defined as

$$T = \{l \mid l \in L \wedge \exists \{q, q'\} \in \binom{Q}{2} : \pi(q) = \pi(q') = l\}$$

Let $Z \subseteq L$ be defined as

$$Z = \{l \mid l \in L \wedge \nexists q \in Q : \pi(q) = l\}$$

Let $M \subseteq Q$ be defined as

$$M = \{q \mid q \in Q \wedge \exists q' \in Q \setminus \{q\} : \pi(q) = \pi(q')\}$$

Lemma 8.7. $2|Z| \geq |M|$

Proof. We have that $|Z| + |T| = |M|$. Because $|L| = |Q| = k$, and there are as many 1-matched q's as there are 1-matched l's, so the only points that remain are the ones in Z, T and M.

Moreover, the number of "arcs" between points fo $M \cup Z \cup T$ is equal to M.

Also, the number of "arcs" is at least $2|T|$, since each node in T is hit by at least 2 arcs.

$$\begin{aligned}
&|T| = |M| - |Z| \\
&2(|M| - |Z|) = 2|T| \leq (\# \text{ of arcs in } M \cup Z \cup T) = |M| \\
&2|M| - 2|Z| \leq |M| \\
&2|Z| \geq |M|
\end{aligned}$$

□

Now, let us define an L, Q-match as a pair (l, q) s.t. $l \in L$ and $q \in Q$. We will consider the following set of L, Q matches.

If $\pi(q)$ is unique to q (if $q \in Q \setminus M$), then we add the match $(\pi(q), q)$.

For each of the remaining q's, we act as follows. We match $q \in M$ to elements of Z in such a way that:

- each $q \in M$ is part of exactly 1 match
- each $l \in Z$ is part of exactly 0, 1 or 2 matches

These matches can be created, since $|M| \leq 2|Z|$. Thus, if you have some extra $q \in M$ to match, just match it to some $l \in Z$ that has been matches the fewest times.

Let P be the set of matches created by this algorithm. Note that we don't really have to come up with P when we run the LS, we just consider P for analysis' sake.

Lemma 8.8. $\forall (l_y, q_x) \in P$,

$$0 \leq Med(L \setminus \{l_y\} \cup \{q_x\}) - Med(L) \leq \sum_{j \in Q_x} D_j^* - \sum_{j \in Q_x} D_j + 2 \sum_{j \in L_y} D_j^*$$

Exercise 8.8.1. Prove Lemma 8.8.

Theorem 8.9. In the general where (where π is not a bijection), LS provides a 5-approximation.

Proof. We sum up the k inequalities, one for each $(l_y, q_x) \in P$.

$$
\begin{aligned}
0 &\le \sum_{(l_y,q_x)\in P} (Med(L \setminus \{l_y\} \cup \{q_x\}) - Med(L)) \le \\
&\le \sum_{(l_y,q_x)\in P} \sum_{j\in Q_x} D_j^* - \sum_{(l_y,q_x)\in P} \sum_{j\in Q_x} D_j + 2 \sum_{(l_y,q_x)\in P} \sum_{j\in L_y} D_j^* \overset{\substack{\text{each } q_x\in Q \\ \text{appears in exactly 1 match}}}{=} \\
&= \sum_{j\in V} D_j^* - \sum_{j\in V} D_j + 2 \sum_{(l_y,q_x)\in P} \sum_{j\in L_y} D_j^* \overset{\substack{\text{each } l_y\in L \\ \text{appears in at most 2 matches}}}{\le} \\
&\le 5\text{OPT} = 5\sum_{j\in V} D_j^* \ge \sum_{j\in V} D_j = \text{ALG}
\end{aligned}
$$

□

A couple of observations on the running time of Algorithm 16:

- If, at the outset (i.e. beginning), S is a local optimum, then the algorithm and the analysis provides a short (i.e. polynomial-size) proof that S is a 5-approximation
- In practive, this LS converges pretty quickly, after 10s of iterations

Algorithm 17

```
procedure LOCALSEARCH$_\varepsilon(V, d, k)$
    let $S_0 \in \binom{V}{k}$ be a 2-approximation to K-Center
    $S \leftarrow S_0$
    while $\exists o \in S, i \in V \setminus S$ s.t. $Med(S \setminus \{o\} \cup \{i\}) < Med(S) - \frac{\varepsilon Med(S_0)}{2nk}$ do
        $S \leftarrow S \setminus \{o\} \cup \{i\}$
    end while
    return $S$
end procedure
```

Theorem 8.10. *LS_ε provides a $(5+\varepsilon)$-approximation to K-Median.*

Proof.

$$
\begin{aligned}
\sum_{(l_y,q_x)\in P} (-\varepsilon \frac{Med(S_0)}{2nk}) &\le \sum_{(l_y,q_x)\in P} (Med(L \setminus \{l_y\} \cup \{q_x\}) - Med(L)) \le \\
&\le 5\text{OPT} - \text{ALG}
\end{aligned}
$$

$$
\begin{aligned}
5\text{OPT} - \text{ALG} &\ge |P|(-\varepsilon \frac{Med(S_0)}{2nk}) \\
5\text{OPT} - \text{ALG} &\ge -\varepsilon k \frac{Med(S_0)}{2nk}
\end{aligned}
$$

$$\text{ALG} \leq 5\text{OPT} + \varepsilon\frac{Med(S_0)}{2n} \leq 5\text{OPT} + \varepsilon\frac{2n\text{OPT}}{2n} = (5+\varepsilon)\text{OPT}$$

□

Theorem 8.11. *LS_ε runs for at most $O(\frac{nk}{\varepsilon})$ iterations (thus, in polytime).*

Proof. Let S_i be the value of S when the i-th iteration ends.

Suppose that the loop runs for ℓ iterations. $\forall i = 0, \ldots, \ell - 1$, the following holds

$$Med(S_{i+1}) \leq Med(S_i) - \varepsilon\frac{Med(S_0)}{2nk}$$

Then,

$$\begin{aligned}
Med(S_1) &\leq Med(S_0) - \varepsilon\frac{Med(S_0)}{2nk} = (1 - \frac{\varepsilon}{2nk})Med(S_0) \\
Med(S_2) &\leq Med(S_1) - \varepsilon\frac{Med(S_0)}{2nk} \leq (1 - \frac{2\varepsilon}{2nk})Med(S_0) \\
&\vdots \\
Med(S_i) &\leq (1 - \frac{i\varepsilon}{2nk})Med(S_0) \\
&\vdots
\end{aligned}$$

Then,

$$0 \leq Med(S_\ell) \leq (1 - \frac{\ell\varepsilon}{2nk})Med(S_0)$$

Thus, given that $Med(S_0) > 0$, it must hold that $1 - \frac{\ell\varepsilon}{2nk} \geq 0$.

Then $\ell \leq \frac{2nk}{\varepsilon}$. □

9 Predictions over time (ML)

We want to predict future events.

- Will the stocks of company A increase in value today?
- Will it rain today?

For instance, you have some training data, that you split in, say, n chunks. You train, independently, a model on each chunk. At testing time, you get the prediction of each of the n models, and you have to select one of them.

9.1 Expert model

We consider the expert model. We have n "experts" that make (binary) predictions for future events. We take into consideration these predictions, and we then compare them with what actually happens.

That is, for $i = 1, 2, \dots, T$, we are given a vector $(Y_{1t}, Y_{2t}, \dots, Y_{nt})$, where $Y_{it} \in \{0, 1\}$ is the prediction of expert i for time t.

We then have to "bet" on whether nature will, at time t, show us 0 or 1. Let $z_t \in \{0, 1\}$ be our prediction for time t.

Nature shows us her bit $x_t \in \{0, 1\}$ for time t.

We pay 1€ is $z_t \neq x_t$.

In Algorithm 18 a simple procedure for choosing experts' predictions.

Algorithm 18 Halving algorithm

```
$S \leftarrow [n]$
for $t = 1, \dots, T$ do
    $A_t \leftarrow |\{i \mid i \in S \wedge Y_{it} = 1\}|$
    $B_t \leftarrow |\{i \mid i \in S \wedge Y_{it} = 0\}|$
    if $A_t \geq B_t$ then
        $z_t \leftarrow 1$
    else
        $z_t \leftarrow 0$
    end if
    we "see" nature's $x_t$ (and pay 1€ is $x_t \neq z_t$)
    $S \leftarrow S \setminus \{i \mid Y_{it} \neq x_t\}$
end for
```

What Algorithm 18 does is it considers what the majority of the experts are predicting at time t, and all that are wrong will not be considered anymore.

Theorem 9.1. If there exists a perfect expert (that is, if $\exists i^* \in [n]$ s.t. $\forall t$, $Y_{i^*t} = x_t$), then the halving algorithm makes m mistakes, with $m \leq \lfloor \log_2 n \rfloor$.

Proof (sketch). If we make a mistake at time t, then at least $\frac{|S|}{2}$ (currently trustworthy) experts make a mistake at time t. At the outset, $|S| = n$. □

In Algorithm 19 we consider a slightly modified version of Algorithm 18, where, in case we've removed all the experts, we put them all back.

Let $m^* = m^*_T$ be the number of mistakes of the best expert in the T rounds.

Theorem 9.2. The number of mistakes of Algorithm 19 is

$$m \leq (m^* + 1)(\log_2 n + 1)$$

Proof (sketch). We could cut the timespan in $m^* + 1$ pieces (cut at the mistakes of a best expert). □

Algorithm 19 Iterative halving algorithm

```
  $S \leftarrow [n]$
  for $t = 1, \dots, T$ do
      $A_t \leftarrow |\{i \mid i \in S \wedge Y_{it} = 1\}|$
      $B_t \leftarrow |\{i \mid i \in S \wedge Y_{it} = 0\}|$
      if $A_t \geq B_t$ then
          $z_t \leftarrow 1$
      else
          $z_t \leftarrow 0$
      end if
      we "see" nature's $x_t$ (and pay 1€ is $x_t \neq z_t$)
      $S \leftarrow S \setminus \{i \mid Y_{it} \neq x_t\}$
      if $S = \emptyset$ then
          $S \leftarrow [n]$
      end if
  end for
```

In Algorithm 20 we consider a different approach. Instead of simply counting how many 0's and 1's we get for the experts (removing them when they are wrong), we always consider all the experts, but weighting their answers based on how reliable they've been so far. At the outset, we assume every expert is perfectly reliable, and dimish their "reliability" each time they are wrong (that is, we give less weight to their answers).

Algorithm 20 Weighted majority

```
  assign a weight $w_i = 1$ to expert $i$, $\forall i \in [n]$
  for $t = 1, \dots, T$ do
      $A_t \leftarrow \sum_{i:Y_{it}=1} w_i$
      $B_t \leftarrow \sum_{i:Y_{it}=0} w_i$
      if $A_t \geq B_t$ then
          $z_t \leftarrow 1$
      else
          $z_t \leftarrow 0$
      end if
      we "see" nature's $x_t$ (and pay 1€ is $x_t \neq z_t$)
      for $i = 1, \dots, n$ do
          if $Y_{it} \neq x_t$ then
              $w_i \leftarrow \frac{w_i}{2}$
          end if
      end for
  end for
```

Theorem 9.3. Algorithm 20 makes number of mistakes

$$m \leq 2.41(m^* + \log_2 n)$$

Proof. Let w_i^t be the weight of expert i at the end of round t (after the weight-update step). Also, let $w_i^0 = 1$, $\forall i \in [n]$.

We define a potential

$$W^t = \sum_{i=1}^{n} w_i^t$$

A.

$$W^0 = \sum_{i=1}^{n} w_i^0 = \sum_{i=1}^{n} 1 = n$$

B. $W^t = \sum_{i=1}^n w_i^t \geq \sum_{i=1}^n w_i^{t+1} = W^{t+1}$. Thus,

$$W^t \geq W^{t+1}$$

C. If $z_t \neq x_t$ (if we make a mistake at time t), then at the beginning of round t the total weight I^{t-1} of the experts that make a mistake in round t satisfies $I^{t-1} \geq \frac{W^{t-1}}{2}$.

Then,

$$\begin{aligned} W^t &= \frac{I^{t-1}}{2} + (W^{t-1} - I^{t-1}) = W^{t-1} - \frac{I^{t-1}}{2} \leq \\ &\leq W^{t-1} - \frac{W^{t-1}}{4} = \frac{3}{4} W^{t-1} \end{aligned}$$

D. After round T, if our algorithm has made m mistakes, then

$$W^t \leq \left(\frac{3}{4}\right)^m \cdot W^0 = \left(\frac{3}{4}\right)^m \cdot n$$

E. If i^* is an expert that has made the fewest number of mistakes m^*, then

$$W^T = \sum_{i=1}^{m} w_i^T \geq w_{i^*}^T$$

$$\begin{aligned} &2^{-m^*} = w_{i^*}^T \leq W^T \leq \left(\frac{3}{4}\right)^m \cdot n \\ &m \log_2 \frac{3}{4} + \log_2 n \geq -m^* \\ &-m \log_2 \frac{4}{3} + \log_2 n \geq -m^* \\ &m^* + \log_2 n \geq m \log_2 \frac{4}{3} \\ &m \leq \frac{1}{\log_2 \frac{4}{3}} (m^* + \log_2 n) \leq 2.41 (m^* + \log_2 n) \end{aligned}$$

□

By changing the $\frac{1}{2}$ factor, we can get to a $(2+\varepsilon)m^* + \frac{O(1)}{\varepsilon} \log n$ upper bound.

9.2 Deterministic algorithms for the Expert problem

Let us consider the worst possibile deterministic algorithm, that says always the opposite of nature.

Consider for instance Table 1.

	$t = 1$	$t = 2$	$t = 2$	$\cdots$	$t = T$
Expert$_0$	0	0	0	$\cdots 0$	0
Expert$_1$	1	1	1	$\cdots 1$	1
Deterministic algorithm	a_1	a_2	a_3	$\dots a_{t-1}$	a_T
Nature	$1 - a_1$	$1 - a_2$	$1 - a_3$	$\dots 1 - a_{t-1}$	$1 - a_T$

Table 1: Example of a bad deterministic algorithm

Let the deterministic algorithm be any (fixed) deterministic algorithm.

Claim 9.3.1. At time T, the deterministic algorithm will have made T mistakes

Proof. $z_t = a_t$, and $x_t = 1 - a_t \neq z_t$. □

Claim 9.3.2. At time T, Expert$_0$ or Expert$_1$ will have made at most $\frac{T}{2}$ mistakes.

Proof. If nature chooses a majority of 0's (if the deterministic algorithm chooses a majority of 1's) then Expert$_0$ will make no more than $\frac{T}{2}$ mistakes.

Analogously, if nature chooses a majority of 1's (if the deterministic algorithm chooses a majority of 0's) then Expert$_1$ will make no more than $\frac{T}{2}$ mistakes. □

Corollary 9.3.1. No deterministic algorithm can guarantee a number of mistakes $m \leq 2m^*$.

9.3 Randomized algorithms

We will now consider randomized algorithms. The expert problem is an instance of a problem where randomization helps us a lot. In fact, it will let us "fool" nature; that is, nature won't be able to construct instances s.t. we are wrong w.p. $\frac{1}{2}$ (as it happend with deterministic algorithms).

Consider Algorithm 21.

Theorem 9.4. Fix $0 < \varepsilon < \frac{1}{2}$. The expected number of mistakes of Algorithm 21 is

$$\mathbb{E}[m] \leq (1 + \varepsilon)m^* + \frac{1}{\varepsilon} \ln n$$

Where m^* is the number of mistakes of a best expert.

Above inequality is often written as

$$\mathbb{E}[m] \leq m^* + \varepsilon \cdot m^* + \frac{1}{\varepsilon} \ln n$$

Where $\varepsilon \cdot m^* + \frac{1}{\varepsilon} \ln n$ is called the "regret" of the algorithm.

Algorithm 21 Randomized Weighted Majority$_\varepsilon$

```
assign weight $w_i = 1$, $\forall i \in [n]$
for $t = 1, \dots, T$ do
    choose randomly expert $i$ w.p. $\frac{w_i}{\sum_{j \in [n]} w_j}$
    suppose $i$ is sampled
    predict $z_t \triangleq Y_{it}$
    see $x_t$ (and pay 1€ is $x_t \neq z_t$)
    for $i = 1, \dots, n$ do
        if $Y_{it} \neq x_t$ then
            $w_i \leftarrow (1 - \varepsilon) w_i$
        end if
    end for
end for
```

Proof. Let w_i^t be the weight of expert i at the end of round t.

Let $w_i^0 = 1$, $\forall i \in [n]$.

Let $W^t = \sum_{i=1}^m w_i^t$ be our "potential".

Let I^t be the weighted fraction of experts that are wrong at time t,

$$I^t = \frac{\sum_{i:Y_{it} \neq x_t} w_i^{t-1}}{\sum_{i=1}^n w_i^{t-1}} = \frac{\sum_{i:Y_{it} \neq x_t} w_i^{t-1}}{W^{t-1}}$$

Suppose i^* is a best expert, and it makes m^* mistakes. Then,

$$w_{i^*}^T = w_i^0 \cdot (1-\varepsilon)^{m^*} = 1 \cdot (1-\varepsilon)^{m^*} = (1-\varepsilon)^{m^*}$$

Thus,

$$W^T \geq w_{i^*}^T = (1-\varepsilon)^{m^*}$$

Moreover,

$$\begin{aligned}
\mathbb{E}[m] &= \sum_{t=1}^T Pr[\text{RWM makes a mistake in round } t] = \\
&= \sum_{t=1}^T Pr[\text{RWM selects an incorrect expert in round } t] = \\
&= \sum_{t=1}^T \sum_{\substack{i \in [n] \\ Y_{it} \neq x_t}} Pr[\text{RWM selects expert } i \text{ in round } t] = \\
&= \sum_{t=1}^T \sum_{\substack{i \in [n] \\ Y_{it} \neq x_t}} \frac{w_i^{t-1}}{\sum_{j \in [n]} w_j^{t-1}} = \sum_{t=1}^T \sum_{\substack{i \in [n] \\ Y_{it} \neq x_t}} \frac{w_i^{t-1}}{\sum_{j \in [n]} W^{t-1}} = \\
&= \sum_{t=1}^T I^t
\end{aligned}$$

Let $t \geq 0$, then

$$\begin{aligned}
W^{t+1} &= \sum_{i=1}^{n} w_i^t = \sum_{i=1}^{n} (w_i^t(1 - \varepsilon[Y_{i,t+1} \neq x_{t+1}])) = \\
&= \sum_{i=1}^{n} w_i^t - \varepsilon \sum_{i=1}^{n} (w_i^t[Y_{i,t+1} \neq x_{t+1}]) = \sum_{i=1}^{n} w_i^t - \varepsilon \sum_{\substack{i \in [n] \\ Y_{it} \neq x_t}} w_i^t = \\
&= \sum_{i=1}^{n} w_i^t - \varepsilon W^t \sum_{\substack{i \in [n] \\ Y_{it} \neq x_t}} \frac{w_i^t}{W^t} = W^t - \varepsilon W^t I^{t+1} = W^t(1 - \varepsilon I^{t+1})
\end{aligned}$$

We then have

$$\begin{aligned}
& W^0 = \sum_{i=1}^{n} w_i^0 = n \\
& W^1 = W^0(1 - \varepsilon I^1) = n(1 - \varepsilon I^1) \\
& W^2 = W^1(1 - \varepsilon I^2) = n(1 - \varepsilon I^1)(1 - \varepsilon I^2) \\
& \vdots \\
& W^t = n \prod_{s=1}^{t} (1 - \varepsilon I^s)
\end{aligned}$$

Then,

$$\begin{aligned}
& n \prod_{s=1}^{T} (1 - \varepsilon I^s) = W^T \geq (1 - \varepsilon)^{m^*} \\
& \ln n - \varepsilon \sum_{s=1}^{T} I^S \geq^{1} \ln n + \sum_{s=1}^{T} \ln(1 - \varepsilon \cdot I^S) \geq m^* \ln(1 - \varepsilon) \\
& \ln n \geq m^* \ln(1 - \varepsilon) + \varepsilon \sum_{s=1}^{T} I^s \\
& \ln n \geq m^* \ln(1 - \varepsilon) + \varepsilon \cdot \mathbb{E}[m] \\
& \varepsilon \cdot \mathbb{E}[m] \leq \ln n - m^* \ln(1 - \varepsilon) \\
& \varepsilon \cdot \mathbb{E}[m] \leq \ln n + m^* \ln \frac{1}{1 - \varepsilon}
\end{aligned}$$

Thus,

$$\begin{aligned}
\mathbb{E}[m] &\leq \frac{1}{\varepsilon} \ln n + \frac{m^*}{\varepsilon} \ln \frac{1}{1 - \varepsilon} \leq^{2} \\
&\leq \frac{1}{\varepsilon} \ln n + \frac{m^*}{\varepsilon}(\varepsilon + \varepsilon^2) = \\
&= (1 + \varepsilon) m^* + \frac{1}{\varepsilon} \ln n
\end{aligned}$$

[1] $\forall x \in (0, 1)$, $\ln(1 - x) \leq -x$

□

10 Math and CS notions

10.1 Tail inequalities

- Markov inequality
- Chebyshev inequality
- Chernoff bound

10.1.1 Markov inequality

If Y is a non negative random variable, then $\forall c \geq 1$

$$Pr[Y \geq c \cdot \mathbb{E}[Y]] \leq \frac{1}{c}$$

Proof. Let ξ be an event, and let

$$X_\xi = \begin{cases} 1 & \text{if } \xi \text{ happens} \\ 0 & \text{o/w} \end{cases} \tag{41}$$

In particular, for $a > 0$, consider

$$X_{Y \geq a} = \begin{cases} 1 & \text{if } Y \geq a \text{ happens} \\ 0 & \text{o/w} \end{cases} \tag{42}$$

Then $Pr[a \cdot X_{Y \geq a} \leq Y] = 1$.

So, $\mathbb{E}[a \cdot X_{Y \geq a}] \leq \mathbb{E}[Y]$. Also, $\mathbb{E}[a \cdot X_{Y \geq a}] = a \cdot \mathbb{E}[X_{Y \geq a}] = a \cdot Pr[Y \geq a]$.

$a \cdot Pr[Y \geq a] \leq \mathbb{E}[Y]$, and $Pr[Y \geq a] \leq \frac{\mathbb{E}[Y]}{a}$.

Let us choose $a = c \cdot \mathbb{E}[Y]$, then $Pr[Y \geq c \cdot \mathbb{E}[Y]] \leq \frac{\mathbb{E}[Y]}{c \cdot \mathbb{E}[Y]} = \frac{1}{c}$. □

10.2 NP

Disclaimer: as of now I'm **not** going to be too formal in the definition of NP, since for this course it is important to understand at least that NP problems are hard to solve.

NP stands for *Non-deterministic Polynomial.* It is the set of problems that can be solved in polynomial time by a Non-Deterministic Turing Machine (NDTM). Suppose input size n, we say that an algorithm runs in *polynomial time* if it runs in time $O(n^c)$, for some constant $c > 0$.

These problems capture the idea of puzzles. Given a puzzle, it is hard to solve it, but, given a solution to the puzzle, it is easy to check if the solution is correct.

[2] $\forall \varepsilon \in (0, \frac{1}{2})$, $\ln \frac{1}{1-\varepsilon} \leq \varepsilon + \varepsilon^2$

A bit more formally, given a set S, we want to know if the instance x belongs to S. Given a solution y to the problem, there exists an algorithm $V(\cdot, \cdot)$ that, runs in polynomial time, and given as inputs x and y (i.e. $V(x, y)$) returns true iff $x \in S$; if this is the case, then y is said to be a valid witness.

Let us see an example on *clique*. We want to know if a graph G has a k-clique; so G would be our instance x and S would be the set of all graphs with a k-clique. Finding a k-clique is hard in general, but, given a set of vertices, our witness y, it is quite easy to check if y is a k-clique. Suppose the graph has n node, at most (i.e. for an n-clique) we would have to check if every node is connected to every other node; this procedure would take time $O(n^2)$.

What happens with many (if not all?) of these NP problems is that the solution size is polynomial w.r.t. the input size (possibly even linear), but the number of possible solutions is exponential.

The ways in which we can deal with these problems are the following:

- brute force the optimal solution (not recommended)
- using smart constructs and smart heuristics, some algorithms, although still having exponential time in the worst case, can solve many real life instances really really fast (possibly even in linear or almost linear time)
- find an approximation in polynomial time (which is what we do in this course)

10.2.1 NP-Hardness

A problem is said to be NP-Hard if it is harder to solve than any other problem in NP.

To formally define NP-Hardness we would should also define what a Karp-Reduction is. Very informally, if a problem A is harder to solve than B means that, if we have an algorithm to solve A, then we could also solve B.

10.2.2 NP-Completeness

A problem is said to be NP-Complete if it is both NP-Hard and in NP. So the NP-Complete problems are the hardest to solve in NP.

10.3 Graphs

10.3.1 Induced subgraph

Let $G(V, E)$ be a graph. We define $G[S]$, the subgraph of G induced by S, for $S \subseteq V$, as the graph having S as a set of vertices and edges $\{e \mid e \in E \wedge e \subseteq S\}$.

Informally, we take we original graph, we cut away any node not in S and all the edges connected to these removed nodes.

10.3.2 Bipartite graph

A graph is said to be *bipartite* if its set of vertices can be partitioned into two sets, V, W, s.t. all the edges have one endpoint in one partition set, and the other endpoint in the other partition set. We denote such a graph $G((V, W), E)$.

Formally, $\forall e = \{v, w\} \in E$, e is s.t. $v \in V$ and $w \in W$.

10.4 Hypersphere

Let $\underline{\mathrm{v}}$ be a vector. Recall that the 2-norm of a vector is defined as

$$\|\underline{\mathrm{v}}\|_2 = \sqrt{\sum_t \underline{\mathrm{v}}(t)^2} = \sqrt{\underline{\mathrm{v}} \cdot \underline{\mathrm{v}}}$$

An hypersphere, on n dimensions, is the generalization of a sphere. It is defined as follows:

$$H_n = \{x \in \mathbb{R}^n \mid \|x\|_2 = 1\}$$

H_2 is a circle. H_3 is a sphere.

10.5 Oracles

An oracle can be seen as an abstract algorithm that can solve a problem in time $O(1)$. It is often thought of having a giant table/matrix with all the answers to a problem; we query the oracle, asking for the solution of an instance, and it picks the answer from the table and gives it to us in one step of computation. A possibly more concrete way of seeing oracle, is they are just sub-routines, that only require time $O(1)$.

Oracles are used in many theorical fields of computer science; e.g. computational complexity and cryptography. Depending on the field of application, they can be seen as algorithms, or Turing machines.

The way in which the interaction with an oracle is defined is usually the following. We want to solve problem A, and we have an oracle that can solve problem B. While solving problem A, we "query" the oracle, that is, we give it an instance of problem B, and it gives us the answer in time $O(1)$. How much time do we need to solve problem A, given an oracle for problem B?

A trivial example of an oracle is the following. Consider an oracle O that solves Max-Cut. We can solve Max-Cut in time $O(1)$, simply by asking O the solution, and returning it.

Oracles can be assumed to solve NP problems, and even undecidable problems. For instance, what problems can we solve, and in how much time, given an oracle for the Halting problem?