SAPIENZA
UNIVERSITÀ DI ROMA

"Sapienza" University of Rome
Faculty of Information Engineering,
Informatics and Statistics
Department of Computer Science

Advanced Algorithms

Author
Alessio Bandiera

April 10, 2025

Contents

Information and Contacts

Personal notes and summaries collected as part of the *Advanced Algorithms* course offered by the degree in Computer Science of the University of Rome "La Sapienza".

Further information and notes can be found at the following link:
https://github.com/aflaag-notes. Anyone can feel free to report inaccuracies, improvements or requests through the Issue system provided by GitHub itself or by contacting the author privately:

- Email: **alessio.bandiera02@gmail.com**
- LinkedIn: **Alessio Bandiera**

The notes are constantly being updated, so please check if the changes have already been made in the most recent version.

Suggested prerequisites:

- Progettazione degli Algoritmi

Licence:

1 Approximation algorithms

TODO

missing introduction

Lastly, we will adopt the following convention in these notes, when talking about **approximation algorithm**. Given a problem P for which there is an optimal solution OPT, and an algorithm A that solves P which outputs a solution ALG to P

- if P is a **minimization** problem, we say that A yields an α-approximation of P if

$$\text{ALG} \leq \alpha \cdot \text{OPT}$$

- if P is a **maximization** problem, we say that A yields an α-approximation of P if

$$\text{ALG} \geq \alpha \cdot \text{OPT}$$

1.1 Approximation through randomness

1.1.1 The Maximum Cut problem

The first problem that will be discussed is the Maximum Cut problem. The **Maximum Cut problem** — in the unweighted case — is a classic combinatorial optimization problem in the branch of graph theory, in which we seek to partition the vertices of an undirected graph into two disjoint subsets while maximizing the number of edges that have endpoints in both subsets. More formally, we will define a **cut** of a graph as follows.

Definition 1.1: Cut

Given an undirected graph $G = (V, E)$, and a subset of its vertices $S \subseteq V$, the **cut** induced by S on G is defined as follows

$$\text{cut}(S) := \{e \in E : |S \cap e| = 1\}$$

Note that in the definition above we are defining the cut of a graph through the intersection between a set of vertices S and edges in E; this is because, in the undirected case, we will consider the edges of a graph $G = (V, E)$ as sets of 2 elements

$$E = \{\{u, v\} \mid u, v \in V\}$$

Therefore, given a set of vertices S, the cut induced by S is simply the set of edges that have only one endpoint in S (implying that the other one will be in $V - S$).

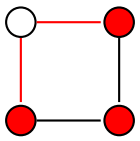


Figure 1.1: Given the set of red vertices S, the green edges represent $\text{cut}(S)$.

With this definition, we can introduce the **Maximum Cut** problem, which is defined as follows.

Definition 1.2: Maximum Cut problem

The **Maximum Cut** (MC) problem is defined as follows: given an undirected graph $G = (V, E)$, determine the set $S \subseteq V$ that maximizes $|\text{cut}(S)|$.

Although this problem is known to be APX-Hard [ALM+98], approximation algorithms and heuristic methods like greedy algorithms and local search are commonly used to find near-optimal solutions.

For now, we present the following **randomized algorithm**, which provides a straightforward $\frac{1}{2}$-approximation for MC. This algorithm runs in polynomial time and achieves the approximation guarantee with high probability.

Algorithm 1.1: Random Cut

Given an undirected graph $G = (V, E)$, the algorithm returns a cut of G.

```
1:  function RandomCut(G)
2:      S := ∅
3:      for v ∈ V do
4:          Let c_v be the outcome of the flip of an independent fair coin
5:          if c_v == H then
6:              S = S ∪ {v}
7:          end if
8:      end for
9:      return S
10: end function
```

Note that this algorithm is powerful, because *it does not care about the structure of the graph in input*, since the output is completely determined by the coin flips performed in the `for` loop. Now we will prove that this algorithm provides a correct **expected $\frac{1}{2}$-approximation** of MC.

Theorem 1.1: Expected approximation ratio of RandomCut

Let $G = (V, E)$ be a graph, and let S^* be an optimal solution to MC on G. Then, given $S = \text{RandomCut}(G)$, it holds that

$$\mathbb{E}[|\text{cut}(S)|] \geq \frac{|\text{cut}(S^*)|}{2}$$

Proof. By definition, note that

$$\forall e \in E \quad e \in \text{cut}(S) \iff |S \cap e| = 1$$

Consider an edge $e = \{v, w\} \in E$; then, by definition

$$\{v, w\} \in \text{cut}(S) \iff (v \in S \wedge w \notin S) \vee (v \notin S \wedge w \in S)$$

and let ξ_1 and ξ_2 be these last two events respectively. Then

$$\Pr[\xi_1] = \Pr[c_v = \text{heads} \wedge c_w = \text{tails}]$$

by definition of the algorithm, and by independence of the flips of the fair coins we have that

$$\Pr[\xi_1] = \Pr[c_v = \text{heads}] \cdot \Pr[c_w = \text{tails}] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

Analogously, we can show that

$$\Pr[\xi_2] = \frac{1}{4}$$

This implies that

$$\Pr[e \in \text{cut}(S)] = \Pr[\xi_1 \vee \xi_2] = \Pr[\xi_1] + \Pr[\xi_2] - \Pr[\xi_1 \wedge \xi_2] = \frac{1}{4} + \frac{1}{4} - 0 = \frac{1}{2}$$

Hence, we have that

$$\mathbb{E}[|\text{cut}(S)|] = \sum_{e \in E} 1 \cdot \Pr[e \in \text{cut}(S)] = \frac{|E|}{2} \geq \frac{|\text{cut}(S^*)|}{2}$$

where the last inequality directly follows from the definition of cut of a graph. □

As previously mentioned, this algorithm has an **expected approximation ratio** of $\frac{1}{2}$, which implies that it may return very bad solutions in some cases, depending on the outcomes of the coin flips. However, thanks to the following algorithm, we can actually transform the **guarantee of expectations** into a **guarantee of high probability** — note that it is possible to show that the previous algorithm provides guarantee of high probability as well, but the proof is much more complex.

Algorithm 1.2: t-times Random Cut

Given an undirected graph $G = (V, E)$ and an integer $t > 0$, the algorithm returns a cut of G.

1: **function** t-TIMESRANDOMCUT(G, t)
2: **for** $i \in [t]$ **do**
3: $S_i :=$ RANDOMCUT(G)
4: **end for**
5: **return** $S \in \arg\max_{i \in [t]} |\text{cut}(S_i)|$
6: **end function**

The algorithm above simply runs the RANDOMCUT algorithm t times, and returns the set S_i that maximizes the cut, among all the various $S_1, \ldots, S_t$. The following theorem will show that a *reasonable number* of runs of the RANDOMCUT algorithm suffices in order to almost certainly obtain a $\approx \frac{1}{2}$-approximation of any optimal solution.

Theorem 1.2

Let $G = (V, E)$ be a graph, and let S^* be an optimal solution to MC on G. Then, given $S = t$-TIMESRANDOMCUT(G, t), it holds that

$$\Pr\left[|\text{cut}(S)| > \frac{1-\varepsilon}{2} |\text{cut}(S^*)|\right] > 1 - \delta$$

where $t = \frac{2}{\varepsilon} \ln \frac{1}{\delta}$ and $0 < \varepsilon, \delta < 1$.

Proof. For each $i \in [t]$, let $C_i := |\text{cut}(S_i)|$ for each S_i defined by the algorithm, and let $N_i := |E| - C_i$. Let $0 < \varepsilon < 1$; since N_i is a non-negative random variable, by Markov's inequality we have that

$$\Pr[N_i \geq (1+\varepsilon)\mathbb{E}[N_i]] \leq \frac{1}{1+\varepsilon} = 1 - \frac{\varepsilon}{1+\varepsilon} \leq 1 - \frac{\varepsilon}{2}$$

In particular, this inequality can be rewritten as follows:

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

As shown in the proof of Theorem 1.1, we know that $\mathbb{E}[C_i] = \frac{|E|}{2}$, therefore

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

Note that the event in the last probability, namely

$$|\mathrm{cut}(S_i)| \leq (1-\varepsilon)\mathbb{E}[|\mathrm{cut}(S_i)|]$$

corresponds to a "bad" solution, i.e. one whose cardinality is at most $(1-\varepsilon)$-th of the expected value.

By definition of the algorithm, each of the t runs of the RandomCut algorithm is independent from the others, therefore the probability of *all* the solutions $S_1, \ldots, S_t$ being "bad" is bounded by

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

Using the fact that

$$\forall x \in \mathbb{R} \quad 1 - x \leq e^{-x} \implies 1 - \frac{\varepsilon}{2} \leq e^{-\frac{\varepsilon}{2}}$$

we have that

$$\Pr[\forall i \in [t] \quad C_i \leq (1-\varepsilon)\mathbb{E}[C_i]] \leq \left(1 - \frac{\varepsilon}{2}\right)^t \leq e^{-\frac{\varepsilon}{2}\cdot t} = e^{-\ln\frac{1}{\delta}} = \delta$$

Therefore, the probability that at least one among $S_1, \ldots, S_t$ is a "good" solution is bounded by

$$\Pr[\exists i \in [t] \quad C_i > (1-\varepsilon)\mathbb{E}[C_i]] = 1 - \Pr[\forall i \in [t] \quad C_i \leq (1-\varepsilon)\mathbb{E}[C_i]] \geq 1 - \delta$$

placeholder □ last part

Note that this result is *very powerful*: for instance, if $\varepsilon = \delta = 0.1$, we get that

$$\Pr[|\mathrm{cut}(S)| > 0.45 \cdot |\mathrm{cut}(S^*)|] \geq 0.9$$

and $t \approx 46$, meaning that we just need to run the RandomCut algorithm approximately 46 times in order to get a solution that is better than a 0.45-approximation with 90% probability.

1.2 Approximation through reduction

1.2.1 The Vertex Cover problem

Another very important problem in graph theory is the Vertex Cover problem, which concerns the combinatorial structure of the **vertex cover**, defined as follows.

Definition 1.3: Vertex cover

Given an undirected graph $G = (V, E)$, a **vertex cover** of G is a set of vertices $S \subseteq V$ such that

$$\forall e \in E \quad \exists v \in S \quad v \in e$$

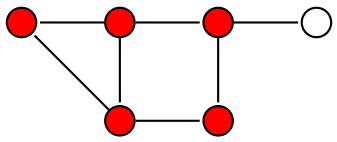


Figure 1.2: An example of a vertex cover.

As shown in figure, a vertex cover is simply a set of vertices that must *cover* all the edges of the graph. Clearly, the trivial vertex cover is representd by $S = V$, but a more interesting solution to the problem is represented by the **minimum vertex cover**.

Definition 1.4: Vertex Cover problem

The **Vertex Cover** (VC) problem is defined as follows: given an undirected graph $G = (V, E)$, determine the vertex cover $S \subseteq V$ of smallest cardinality.

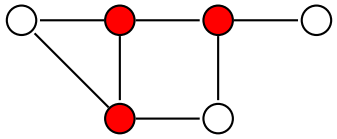


Figure 1.3: This is the *minimum vertex cover* of the previous graph.

As famously proved by Karp [Kar72] in 1972, this problem is NP-Complete, hence we are interested in finding algorithms that allow to find approximations of optimal solutions. For instance, an algorithm that is able to approximate VC concerns the matching problem.

Definition 1.5: Matching

Given an undirected graph $G = (V, E)$, a **matching** of G is a set of edges $A \subseteq E$ such that

$$\forall e, e' \in A \quad e \cap e' = \varnothing$$

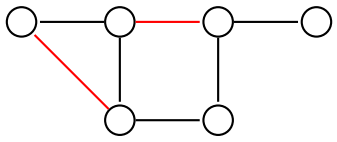


Figure 1.4: A *matching* of the previous graph.

As shown in figure, a matching is nothing more than a set of edges that must not share endpoints with each other — for this reason, in literature it is often referred to as **independent edge set**. Differently from the vertex cover structure, in this context the trivial matching is clearly the set $A = \varnothing$, which vacuously satisfies the matching condition. However, a more interesting solution is represented by the **maximum matching**, but this time we have to distinguish two slightly different definitions, namely the concept of *maximal* and *maximum*.

Definition 1.6: Maximal matching

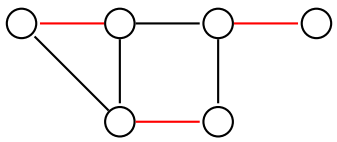
A **maximal matching** is a matching that cannot be extended any further.

For instance, the matching shown in Figure 1.4 is actually a **maximal matching**, because no other edge in E can be added to the current set of edges A of the matching without breaking the matching condition.

Definition 1.7: Maximum matching

A **maximum matching** is a matching that has the largest cardinality.

Clearly, the previous example does not represent a **maximum matching**, because the following set of edges



is still a valid matching for the graph, but has a larger cardinality than the previous set.

Differently from VC, a *maximum matching* can be found in polynomial time. Moreover, the following algorithm can be used to determine a *maximal matching* of a given

graph.

Algorithm 1.3: Maximal matching

Given an undirected graph $G = (V, E)$, the algorithm returns a maximal matching of G.

```
1: function MaximalMatching(G)
2:     S := ∅
3:     while E ≠ ∅ do
4:         Choose e ∈ E
5:         S = S ∪ {e}
6:         Remove from E all the edges incident on u or on v
7:         E = E − {e}
8:     end while
9:     return S
10: end function
```

Idea. The algorithm is very straightforward: at each iteration, a random edge $e = \{u, v\}$ is chosen from E, and then any edge $e' \in E$ such that $e \cap e' \neq \varnothing$ is removed from E.

Clearly, line 6 ensures that the output is a *matching*, and the terminating condition of the `while` loop ensures that it is *maximal*, but since the output depends on the chosen edges, S is not guaranteed to be *maximum*.

Another major reason on why we focus on matchings is the following theorem.

Theorem 1.3: Matchings bound vertex covers

Given an undirected graph $G = (V, E)$, a matching $A \subseteq E$ of G, and a vertex cover $S \subseteq V$ of G, we have that $|S| \geq |A|$.

Proof. By definition, any vertex cover S of $G = (V, E)$ is also a vertex cover for $G^B = (V, B)$, for any set of edges $B \subseteq E$, and in particular this is true for $G^A = (V, A)$.

Now consider G^A, and a vertex cover C on it: by construction we have that $\Delta \leq 1$, therefore any vertex in C will cover at most 1 edge of A. This implies that if $|C| = k$, then C will cover at most k edges of G^A.

Lastly, since G^A has $|A|$ edges by definition, any vertex cover defined on G^A has to contain at least $|A|$ vertices. This implies that no vertex cover S of G smaller than $|A|$ can exist, because S will have to cover at least the edges in A. □

Thanks to this theorem, we can easily show that the algorithm that we just presented in order to find maximal matchings is a 2-approximation of VC.

Theorem 1.4: 2-approximation of VC problem

Given a graph G, and $M = \text{MAXIMALMATCHING}(G)$, let $S := \bigcup_{e \in M} e$. Then S is a 2-approximation of VC on G.

Proof. Consider an optimal solution S^* to VC on G, and let $M = \{e_1, \dots, e_t\}$. Note that at each iteration of the algorithm exactly 1 edge is added to $M \subseteq V$, hence it always holds that

$$S_i \cap S_{i+1} = e_i = \{u, v\}$$

for any iteration $i \in [t-1]$. Moreover, since M is a matching, it holds that $\forall e, e' \in M \quad e \cap e' = \varnothing$, therefore $|S| = 2\,|M|$. Finally, by the previous theorem we have that $|M| \leq |S^*|$, concluding that

$$|S| = 2\,|M| \leq 2\,|S^*|$$

□

This 2-approximation algorithm is conjectured to be optimal, but it has not been proven yet. In fact, VC is conjectured to be NP-Hard to $(2-\varepsilon)$-approximate, for any $\varepsilon > 0$.

Interestingly, the decisional version of VC is Fixed Parameter Tractable. This characterization comes from the nature of the problem: for each edge $e = \{u, v\}$ of a given undirected graph $G = (V, E)$, either u or v has to be in the vertex cover, therefore it possible to approach VC by trying all possible choices of set of vertices $S \subseteq V$, and backtrack if necessary. The following algorithm employs this idea.

Algorithm 1.4: Decisional VC

Given an undirected graph $G = (V, E)$, and an integer k, the algorithm returns `True` if and only if G admits a vertex cover of size k.

```
1:  function VC(G, k)
2:      if E == ∅ then
3:          return True
4:      else if k == 0 then
5:          return False
6:      else
7:          Choose e = {u, v} ∈ E
8:          if VC(G[V − {u}]), k − 1) then
9:              return True
10:         end if
11:         if VC(G[V − {v}]), k − 1) then
12:             return True
13:         end if
14:         return False
15:     end if
16: end function
```

Note that this algorithm actually solves the *decisional* version of VC. Moreover, the algorithm uses the definition of **induced subgraph**, which is the following.

Definition 1.8: Induced subgraph

Given an undirected graph $G = (V, E)$, and a set of vertices $S \subseteq V$, then $G[S]$ represents the **subgraph induced by** S **on** G, and it is obtained by removing from G all the nodes of $V - S$ — and their corresponding edges.

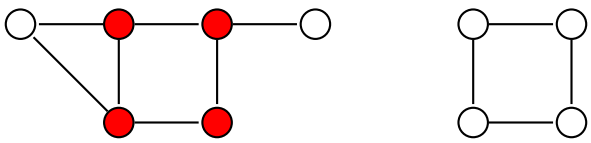


Figure 1.5: On the left: a graph G and a set of vertices S. On the right: the graph $G[S]$.

Idea. The structure of the algorithm consists of a simple backtracking algorithm:

- if the current graph has no edges, we covered every edge of the graph, therefore we return `True`
- if the current graph has some edges, but $k = 0$, then G does not admit a vertex cover of size k, thus we return `False`
- if the current graph has some edges, and $k \neq 0$, then we choose an edge $e = \{u, v\} \in E$ arbitrarily, and we try to consider first u then v in a possible vertex cover — note that $G[V - \{x\}]$ is a graph that does not contain x, neither any edge adjacent to it; if both attempts fail, we return `False`

Cost analysis. It is easy to see that the cost directly depends on the number of recursive calls that the algorithm performs, which is 2^k in the worst case, and the cost of constructing $G[V - \{x\}]$, which we can assume to be $O(n^2)$. Hence, the algorithm has a total cost of $O\left(2^k \cdot n^2\right)$.

2

Mathematical Programming

Up to this point, we have introduced several NP-Complete problems and discussed approximation algorithms that provide near-optimal solutions within a certain approximation ratio. We will now explore a new technique that can be leveraged to design approximation algorithms for a broader class of problems, namely **mathematical programming**.

Mathematical programming is a powerful framework used to model and solve **optimization problems** across various fields, including operations research, computer science and engineering. It provides a structured way to *maximize* or *minimize* an **objective function** while satisfying a set of **constraints**. In particular, optimization problems are usually expressed as follows

$$\begin{aligned} &\max \ f(x) \\ &g_i(x) \leq b_i \quad \forall i \in [n] \\ &x \in V \end{aligned}$$

In this example, we see that

- x is a *vector* that lies inside the vector space V
- the objective function is $f(x)$, which can be either maximized or minimized
- $g_i(x) \leq b_i$ is a constraint — i.e. an inequality — that x must satisfy

Among the most widely studied forms of mathematical programming are **Linear Programming** (LP), **Integer Programming** (IP), and **Semidefinite Programming** (SDP), each with distinct properties and applications.

Starting from linear programming, in LPs both the objective function and the constraints are *linear* w.r.t. V, and there are no *equality* constraints.

$$\max\ x_1 + x_2$$

$$\begin{aligned} 2x_1 + x_2 &\leq 10 \\ x_i &\geq 0 \qquad \forall i \in [n] \\ x &\in \mathbb{R}^n \end{aligned}$$

Figure 2.1: Example of an LP.

LPs can be solved in **polynomial time**: specifically, if an LP has n variables, m constraints, and each coefficient can be expressed as the ratio of two t-bit integers (where real numbers are approximated as rationals), then the Ellipsoid method can solve it in time $O((nmt)^c)$ for some constant $c > 0$. This result extends, to some extent, to SDPs as well.

Despite its theoretical guarantee of polynomial runtime, in practice the Simplex method is often preferred, as it operates based on *pivot rules*. In fact, although all known pivot rules for the Simplex method exhibit a theoretical *exponential* lower bound due to specially constructed worst-case instances, its average performance is significantly better than that of the Ellipsoid method in real-world scenarios.

2.1 IPs and LP relaxation

Differently from LPs, in IPs the vector space of interest is $\{0,1\}^n$. IPs can be used to solve a wide range of problems. For instance, given a graph $G = (V, E)$, the **vertex cover** problem that we discussed in the previous section can be formulated through the following IP:

$$\min \sum_{u \in V}^{n} x_v$$

$$\begin{aligned} x_u + x_v &\geq 1 \quad \forall \{u, v\} \in E \\ x_v &\in \{0,1\} \quad \forall u \in V \end{aligned}$$

Figure 2.2: IP for VC.

It is fairly straightforward to prove that an optimal solution to this IP yields an optimal solution to VC.

Lemma 2.1

Given a graph G, if $\{x_u^*\}_{u \in V}$ is an optimal solution to the previous IP, then

$$S^* := \{v \in V \mid x_v^* = 1\}$$

is a minimum vertex cover for G.

Proof. Consider an optimal solution $\{x_u^*\}_{u \in V}$ to VC, and define S^* as the set of vertices $v \in V$ such that $x_v^* = 1$. Note that any optimal solution is also a feasible solution, i.e. it satisfies the constraints of the IP.

Note that the first constraint of the IP forces that for each $\{u, v\} \in E$ the sum between $x_u^* + x_v^*$ is at least one, and the second constraint forces each variable x_v^* to be either 0 or 1. Therefore, together these two constraints imply that for any edge $\{u, v\} \in E$ at least one between x_u^* and x_v^* is 1, and by definition of S^* this means at least one of the endpoints of $\{u, v\}$ is inside S^*. We conclude that S^* is indeed a vertex cover for G, and by its definition note that $|S^*| = \sum_{u \in V} x_u^*$.

Claim: Given a vertex cover S of a graph $G = (V, E)$, there exists a feasible solution $\{x_u\}_{u \in V}$ to the IP having value $|S|$.

Proof of the Claim. Define the solution $\{x_u\}_{u \in V}$ by setting $x_u = 1$ if and only if $u \in S$. Clearly, the value of this solution is indeed $\sum_{u \in V} x_u = |S|$; moreover, by definition of vertex cover, for any edge $\{u, v\} \in E$ at least one between u and v must be in S, therefore at least one between x_u and x_v is set to 1, implying that $x_u + x_v \geq 1$ is always satisfied. □

By way of contradiction, suppose that S^* is not a minimum vertex cover. Hence, there must be another vertex cover S' such that $|S'| < |S^*|$. By the previous claim, this implies that there exists a feasible solution $\{x'_u\}_{u \in V}$ for the IP that has value $|S'|$, but then

$$\sum_{u \in V} x'_u = |S'| < |S^*| = \sum_{u \in V} x_u^*$$

which contradicts the optimality of the solution of $\{x_u^*\}_{u \in V}$ for the IP. □

In particular, this lemma implies that VC can be reduced to Integer programming, indeed solving IPs is actually NP-Hard [Kar72], differently from LPs. This result shows that IPs cannot be used *directly* to obtain perfect solutions, but they are still very useful thanks to **relaxation**.

To *relax* an IP, we simply replace the constraint $x \in \{0, 1\}^n$ with $0 \leq x \leq 1$, transforming the IP into an LP.

$$\min \sum_{u \in V}^{n} x_v$$

$$\begin{aligned} x_u + x_v &\geq 1 \quad \forall \{u, v\} \in E \\ 0 \leq x_v &\leq 1 \quad \forall u \in V \end{aligned}$$

Figure 2.3: LP relaxation for the IP of VC.

But solving this LP is not enough to obtain a meaningful solution: in fact, a real-valued solution for this problem does not directly yield a vertex cover for a given graph. To fix this issue, the optimal solution of the LP relaxation is usually transformed through techniques such as **rounding**. Intuitively, the simplest possible type of *rounding rule* is the following: given a solution $\{\overline{x}_u\}_{u \in V}$ to the LP relaxation, to obtain a VC consider the following set

$$S := \left\{ v \in V \mid \overline{x}_v \geq \tfrac{1}{2} \right\}$$

and for VC in particular, we can prove that this rounding rule actually yields a 2-approximation of any optimal solution.

Theorem 2.1

Given a graph $G = (V, E)$, if $\{\overline{x}_u\}_{u \in V}$ is an optimal solution to the LP relaxation of the IP for VC, then

$$\overline{S} := \left\{ v \in V \mid \overline{x}_v \geq \tfrac{1}{2} \right\}$$

is a 2-approximation for VC.

Proof. Since $\{\overline{x}_u\}_{u \in V}$ is an optimal solution to the LP relaxation, it must satisfy the first constraint for which $\overline{x}_u + \overline{x}_v \geq 1$ for any $\{u, v\} \in E$. Moreover, for the second constraint we have that $\overline{x}_u \geq 0$ for all $u \in V$, therefore

$$\forall \{u, v\} \in E \quad \max(\overline{x}_u, \overline{x}_v) \geq \frac{\overline{x}_u + \overline{x}_v}{2} \geq \frac{1}{2}$$

which means that at least one between $\overline{x}_u$ and $\overline{x}_v$ is at least $\frac{1}{2}$, implying that the edge $\{u, v\}$ will be covered by at least one of the two endpoints u and v, by definition of $\overline{S}$. This proves that $\overline{S}$ is a vertex cover of G.

To prove that $\overline{S}$ is indeed a 2-approximation, we just need to show the following: given a minimum vertex cover S^* of G, it holds that $\left|\overline{S}\right| \leq 2\,|S^*|$. By the claim in Lemma 2.1, we know that S^* there exists a feasible solution $\{x^*_u\}_{u \in V}$ for the IP, which must be optimal

for the IP since S^* is a minimum vertex cover for G. Therefore, we have that

$$\begin{aligned}
\left|\overline{S}\right| &= \sum_{v \in \overline{S}} 1 \\
&\leq \sum_{v \in \overline{S}} 2\overline{x}_v \qquad \left(v \in \overline{S} \implies \overline{x}_v \geq \tfrac{1}{2}\right) \\
&= 2 \sum_{v \in \overline{S}} \overline{x}_v \\
&\leq 2 \sum_{v \in V} \overline{x}_v \qquad \left(\overline{S} \subseteq V \wedge \overline{x}_v \geq 0\right) \\
&\leq 2 \sum_{v \in V} x^*_v \\
&= 2 \left|S^*\right|
\end{aligned}$$

where the last inequality comes from the fact that the constraints of the LP are *weaker* than the ones of the IP. □

However, note that this result should not come a surprise. In fact, consider a graph G, an optimal solution to the LP relaxation $\{\overline{x}_u\}_{u \in V}$ and $\overline{S}$ defined as previously shown; given an edge $\{u, v\} \in E(G)$, in the worst case we have that

$$\overline{x}_u = \overline{x}_v = \frac{1}{2}$$

which still satisfies both constraints of the LP relaxation, since $\frac{1}{2} + \frac{1}{2} = 1 \geq 1$. This means that, in the worst case, both u and v end up inside $\overline{S}$, which gives an intuitive reason to why this LP relaxation indeed yields a 2-approximation solution.

2.2 Integrality gap

Consider a problem P, its equivalent IP, and the relative LP relaxation. Given an instance $I \in \mathrm{P}$ of the problem, we will denote with $\mathrm{IP}^*_{\mathrm{P}}(I)$ and $\mathrm{LP}^*_{\mathrm{P}}(I)$ the optimal values for the IP and the LP of the problem P on the instance I — we will omit P and I the context is clear enough. Note that, in general, it holds that $\mathrm{LP}^* \leq \mathrm{IP}^*$ since the constraints of the LP relaxation are *weaker* than the ones of the IP.

For example, the inequalities discussed in the proof of Lemma 2.1 could be rewritten as follows

$$\begin{aligned}
\left|\overline{S}\right| &= \dots \\
&\leq 2 \sum_{v \in V} \overline{x}_v = 2\mathrm{LP}^* \\
&\leq 2 \sum_{v \in V} x^*_v = 2\mathrm{IP}^* = 2 \left|S^*\right|
\end{aligned}$$

and in particular $\left|\overline{S}\right| \leq 2\mathrm{LP}^* \leq 2\mathrm{IP}^*$. Can we improve this approximation ratio of 2 through LP relaxation? In general, for any α possible approximation ratio, it must hold

that

$$\alpha \geq \frac{\mathrm{IP}^*}{\mathrm{LP}^*}$$

because otherwise

$$\alpha < \frac{\mathrm{IP}^*}{\mathrm{LP}^*} \implies \left|\overline{S}\right| \leq \alpha \mathrm{LP}^* < \frac{\mathrm{IP}^*}{\mathrm{LP}^*} \cdot \mathrm{LP}^* = \mathrm{IP}^*$$

meaning that $\overline{S}$ would be a solution better than the optimal solution of the IP, which is impossible. We can generalize this concept as follows.

Definition 2.1: Integrality gap

Given a problem P and an instance $I \in \mathrm{P}$, the **integrality gap** between $\mathrm{IP}^*_{\mathrm{P}}(I)$ and $\mathrm{LP}^*_{\mathrm{P}}(I)$ is defined as follows

$$\mathrm{IG}_{\mathrm{P}}(I) = \frac{\mathrm{IP}^*_{\mathrm{P}}(I)}{\mathrm{LP}^*_{\mathrm{P}}(I)}$$

The integrality gap for the problem P is defined as follows

$$\mathrm{IG}_{\mathrm{P}} = \sup_{I \in \mathrm{P}} \mathrm{IG}_{\mathrm{P}}(I) = \sup_{I \in \mathrm{P}} \frac{\mathrm{IP}^*_{\mathrm{P}}(I)}{\mathrm{LP}^*_{\mathrm{P}}(I)}$$

In fact, through the previous argument, we can derive the following property that *must* hold for any approximation ratio.

Proposition 2.1: Limits of LP relaxation

Given a problem P for which there is an α-approximation algorithm which uses LP relaxation, it holds that

- if P is a **minimization problem**, then $\alpha \geq \mathrm{IG}_{\mathrm{P}}$
- if P is a **maximixation problem**, then $\alpha \leq \mathrm{IG}_{\mathrm{P}}$

Now, let us analyze again VC and try to bound $\mathrm{IG}_{\mathrm{VC}}$. Consider the following *clique* graph:

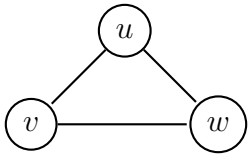


Figure 2.4: The graph K_3.

it is easy to see that

$$\mathrm{IP}^*(K_3) = 1 + 1 = 2$$

because 1 single node is not sufficient to cover all 3 edges in $E(K_3)$. However, since the values in the solution of the LP can be *real-valued*, the value of an optimal solution for the LP relaxation is actually achieved by setting

$$\overline{x}_u = \overline{x}_v = \overline{x}_w = \frac{1}{2} \implies \mathrm{LP}^*(K_3) = 3 \cdot \frac{1}{2} = \frac{3}{2}$$

therefore, by definition of IG we have that

$$\exists I \in \mathrm{P} \quad \mathrm{IG}_{\mathrm{VC}}(I) := \frac{\mathrm{IP}^*_{\mathrm{VC}}(K_3)}{\mathrm{LP}^*_{\mathrm{VC}}(K_3)} = \frac{2}{\frac{3}{2}} = \frac{4}{3} \implies \mathrm{IG}_{\mathrm{VC}} := \sup_{I \in \mathrm{P}} \frac{\mathrm{IP}^*_{\mathrm{VC}}(I)}{\mathrm{LP}^*_{\mathrm{VC}}(I)} \geq \frac{4}{3}$$

Moreover, Theorem 2.1 shows that we already know an algorithm that employs LP relaxation which yields a 2-approximation of VC; therefore, this lower bound on $\mathrm{IG}_{\mathrm{VC}}$ — together with the previous proposition — implies that any possible approximation ratio α on VC must satisfy

$$2 \geq \alpha \geq \mathrm{IG}_{\mathrm{VC}} \geq \frac{4}{3}$$

The following theorem proves that we can actually bound $\mathrm{IG}_{\mathrm{VC}}$ tightly.

Theorem 2.2: Integrality gap for VC

$\mathrm{IG}_{\mathrm{VC}} = 2$

Proof. We already proved that the upper bound is 2, so we just need to prove that the lower bound is 2 as well.

Consider a clique K_n; by the same reasoning presented for the case of K_3, a feasible solution for the LP relaxation over this graph would be

$$x_1 = \ldots = x_n = \frac{1}{2} \implies \mathrm{LP}^*(K_n) \leq n \cdot \frac{1}{2} = \frac{n}{2}$$

Claim: Any minimum vertex cover of K_n has exactly $n-1$ vertices.

Proof of the Claim. Consider a vertex cover $S = \{v_1\}$ for any vertex $v_1 \in V(K_n)$; since K_n is a clique, by definition $\deg(v_1) = n-1$, therefore S is able to cover only $n-1$ *uncovered* edges of K_n. Now, consider another vertex $v_2 \in V(K_n)$, and add it to $S = \{v_1, v_2\}$; we observe that $\deg(v_2) = n-1$, but $v_1 \sim v_2$ because K_n is a clique, therefore v_2 will be able to cover only $n-2$ *uncovered* edges of K_n. By the same reasoning, each new vertex $v_i \in V(K_n)$ added to S will be able to cover only $n-i$ *uncovered* edges of K_n. However, note that

$$|E(K_n)| = \binom{n}{2} = \frac{n(n-1)}{2}$$

therefore, to cover all the edges of K_n we need $|S|$ to satisfy the following inequality

$$\sum_{i=1}^{|S|} (n-i) \geq \frac{n(n-1)}{2} \iff n\,|S| - \frac{|S|\,(|S|+1)}{2} \geq \frac{n(n-1)}{2}$$

After some calculations, we derive the following inequality

$$|S|^2 + |S|(1-2n) + n(n-1) \leq 0$$

which is satisfied for any value $n-1 \leq |S| \leq n$, therefore a vertex cover of cardinality $|S| = n-1$ suffices to cover all the edges of K_n. □

This claim shows that for any n it holds that $\mathrm{IP}^*(K_n) = n-1$, therefore we have that

$$\mathrm{IG}(K_n) = \frac{\mathrm{IP}^*(K_n)}{\mathrm{LP}^*(K_n)} \geq \frac{n-1}{\frac{n}{2}}$$

which means that

$$\mathrm{IG}_{\mathrm{VC}} = \sup_{I \in \mathrm{VC}} \mathrm{IG}_{\mathrm{VC}}(I) \geq \lim_{n \to +\infty} \frac{n-1}{\frac{n}{2}} = 2$$

□

2.3 The Set Cover problem

The next problem that we will study can be seen as a *generalization* of the VC problem, which is the Set Cover problem, defined as follows.

Definition 2.2: Set Cover problem

The **Set Cover** (SC) problem is defined as follows: given a *universe* (or *ground*) set $\mathcal{U} = [n]$, and a collection of sets $C = \{S_1, \ldots, S_m\}$ such that $S_i \subseteq \mathcal{U}$, determine the smallest sub-collection $S \subseteq C$ such that $\bigcup_{S_j \in S} S_j = \mathcal{U}$.

In other words, we are asked to determine the smallest sub-collection of the given C such that we can still cover the whole universe set $\mathcal{U}$. For instance, given $\mathcal{U} = [3]$ and $S_1 = \{1,2\}$, $S_2 = \{2,3\}$, $S_3 = \{1,3\}$, we can cover $\mathcal{U}$ with just $S = \{S_1, S_2\}$.

As for VC, in 1972 [Kar72] proved that SC is NP-Complete as well. Moreover, similarly to what we did for VC, we can convert SC into an IP, as follows.

$$\min \sum_{j=1}^{m} x_j$$

$$\sum_{\substack{j \in [m]: \\ i \in S_j}} x_j \geq 1 \quad \forall i \in [n]$$

$$x_j \in \{0,1\} \quad \forall j \in [m]$$

Figure 2.5: IP for SC.

The first constraint of the IP states that, given an element i in the universe set, at least one of the variables x_j, representing the sets S_j which contain i, must be set to 1. In other words, we are guaranteeing that all the elements $i \in \mathcal{U}$ are covered by at least one set of C. Lastly, we want to minimize over the size of the sub-collection of C, hence the objective function.

The LP relaxation that we will consider is the same that we defined for VC. However, differently from VC, the *rounding rule* that we applied to obtain an integral solution — namely by defining

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

cannot be applied for this problem. For instance, say that some element $i \in \mathcal{U}$ is contained in 3 sets S_1, S_2 and S_3; hence, the second constraint forces the solution of the LP to satisfy

$$x_1 + x_2 + x_3 \geq 1$$

Nevertheless, by setting

$$x_1 = x_2 = x_3 = \frac{1}{3}$$

we would get a feasible solution for the LP relaxation, but then our rounding rule would return an empty set $S = \varnothing$, which is not a feasible solution for our instance of SC since i would not be covered.

To fix this issue, we are going to present a **randomized rounding algorithm**, which surprisingly seem to be the *best* approach to perform rounding on LP relaxation solutions that we have at our disposal.

Algorithm 2.1: Randomized rounding for SC

Given an instance $(\mathcal{U}, C)$ of SC, the algorithm returns a set cover A for $\mathcal{U}$.

---

function RandomizedRoundingSC($\mathcal{U}$, C)
    $A := \varnothing$
    $\{\overline{x}_j\}_{j \in [m]} := \mathrm{LP}_{\mathrm{SC}}(\mathcal{U}, C)$ ▷ an optimal soluion on the LP relaxation
    for $k \in [\lceil 2 \ln n \rceil]$ **do**
        for $j \in [m]$ **do**
            Let $c_{k,j}$ be the outcome of the flip of an ind. coin with H prob. set to $\overline{x}_j$
            if $c_{k,j} ==$ H **then**
                $A = A \cup \{S_j\}$
            end if
        end for
    end for
end function

First, we are going to prove that the output A of this algorithm is indeed a set cover, with enough probability.

Lemma 2.2

Let $(\mathcal{U}, C)$ be a SC instance, and let $A = \textsc{RandomizedRoundingSC}(\mathcal{U}, C)$. Then, it holds that

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

Proof. Each iteration of the outermost `for` loop will be referred to as *phase*.

Claim: The element i is covered by A in *phase* k with probability at least $1 - \frac{1}{e}$.

Proof of the Claim.

$$\begin{aligned}
\Pr[i \text{ is not covered in } \textit{phase } k] &= \prod_{\substack{j \in [m]: \\ i \in S_j}} (1 - \overline{x}_j) && \text{(the prob. of T)} \\
&\leq \prod_{\substack{j \in [m]: \\ i \in S_j}} e^{-\overline{x}_j} && (1 - x \leq e^{-x}) \\
&= e^{-\sum_{\substack{j \in [m]: \\ i \in S_j}} \overline{x}_j} \\
&\leq e^{-1} && \text{(second constraint of the LP)} \\
&= \frac{1}{e}
\end{aligned}$$

□

Claim: The element i is not covered by any set of A with probability at most $\frac{1}{n^2}$.

Proof of the Claim.

$$\begin{aligned}
\Pr[i \text{ is not covered by any set of } A] &= \prod_{k=1}^{\lceil 2 \ln n \rceil} \Pr[i \text{ is not covered in } \textit{phase } k] \\
&\leq \prod_{k=1}^{\lceil 2 \ln n \rceil} \frac{1}{e} && \text{(previous claim)} \\
&= e^{-\lceil 2 \ln n \rceil} \\
&\leq e^{-2 \ln n} \\
&= \frac{1}{n^2}
\end{aligned}$$

□

Claim: A is a set cover with probability at least $1 - \frac{1}{n}$.

Proof of the Claim.

$$\begin{aligned}\Pr[A \text{ is not a set cover}] &= \Pr[\exists i \quad i \text{ is not covered by any set of } A] \\ &\leq \sum_{i=1}^{n} \Pr[i \text{ is not covered by any set of } A] \\ &\leq \sum_{i=1}^{n} \frac{1}{n^2} \qquad \text{(previous claim)} \\ &= \frac{n}{n^2} \\ &= \frac{1}{n}\end{aligned}$$

□

□

Next, we will show that this algorithm yields *on average* a $\lceil 2 \ln n \rceil$-approximation of SC.

Lemma 2.3

Let $(\mathcal{U}, C)$ be a SC instance, and let $A = \text{RandomizedRoundingSC}(\mathcal{U}, C)$. Then, it holds that

$$\mathbb{E}\left[|A|\right] \leq \lceil 2 \ln n \rceil \, \text{IP}^*$$

Proof. Fix a *phase* k, and let A_k be the collection of sets added to A at *phase* k; then it holds that

$$\mathbb{E}\left[|A_k|\right] = \sum_{j=1}^{m} 1 \cdot \Pr[S_j \in A_k] = \sum_{j=1}^{m} \overline{x}_j = \text{LP}^*$$

Moreover, since $A = \bigcup_{k \in \lceil 2 \ln n \rceil} A_k$, we have that

$$\mathbb{E}\left[|A|\right] \leq \mathbb{E}\left[\sum_{k \in \lceil 2 \ln n \rceil} |A_k|\right] = \sum_{k \in \lceil 2 \ln n \rceil} \mathbb{E}\left[|A_k|\right] = \sum_{k \in \lceil 2 \ln n \rceil} \text{LP}^* = \lceil 2 \ln n \rceil \, \text{LP}^* \leq \lceil 2 \ln n \rceil \, \text{IP}^*$$

□

TODO missing something

Additionally, note that the algorithm can be modified to get a better bound, by simply replacing $\lceil 2 \ln n \rceil$ with $\lceil (1 + \varepsilon) \ln n \rceil$, for any $\varepsilon > 0$. In fact, with this modification we would get that $\mathbb{E}\left[|A|\right] \leq \lceil (1 + \varepsilon) \ln n \rceil \, \text{LP}^*$, therefore missing something

With VC we were able to provide a tight bound on IG_{VC}; for SC instead, even though the value of IG_{SC} is known [Lov75], the proof is very complex and we will only show a lower bound.

Theorem 2.3: Integrality gap of SC

For any $n \in \mathbb{N}$, it holds that

$$\frac{1}{4 \ln 2} \leq \mathrm{IG}_{\mathrm{SC}} \leq \lceil \ln n \rceil$$

Proof. We already proved the upper bound, so we just need to prove the lower bound. Furthermore, as shown with VC in the previous section, to provide a lower bound on $\mathrm{IG}_{\mathrm{SC}}$ it suffices to show an instance of SC for which the bound holds.

Let $m \geq 2$ be an even integer, and define the following instance of SC: set

$$\mathcal{U}_m := \{e_A \mid A \subseteq [m] \wedge |A| = \tfrac{m}{2}\} \implies n = |\mathcal{U}_m| = \binom{m}{\frac{m}{2}}$$

and define the collection of sets as follows

$$\forall j \in [m] \quad S_j := \{e_A \mid e_A \in \mathcal{U}_m \wedge j \in A\}$$

and set $C_m := \{S_1, \ldots, S_m\}$ For example, if $m = 4$ we have that

$$\mathcal{U}_4 := \{e_A \mid A \subseteq \{1,2,3,4\} \wedge |A| = \tfrac{4}{2} = 2\} = \{e_{\{1,2\}}, e_{\{1,3\}}, e_{\{1,4\}}, e_{\{2,3\}}, e_{\{2,4\}}, e_{\{3,4\}}\}$$

$$S_1 := \{e_{\{1,2\}}, e_{\{1,3\}}, e_{\{1,4\}}\}$$
$$S_2 := \{e_{\{1,2\}}, e_{\{2,3\}}, e_{\{2,4\}}\}$$
$$S_3 := \{e_{\{1,3\}}, e_{\{2,3\}}, e_{\{3,4\}}\}$$
$$S_4 := \{e_{\{1,4\}}, e_{\{2,4\}}, e_{\{3,4\}}\}$$

Note that, thanks to Stirling's approximation we know that

$$n = \binom{m}{\frac{m}{2}} = \Theta\left(\frac{2^m}{\sqrt{m}}\right) \implies m = \log n - \Theta(\log \log n)$$

Claim: $\forall m \geq 2$ even $\quad \mathrm{LP}^*_{\mathrm{SC}}(\mathcal{U}_m, C_m) \leq 2$.

Proof of the Claim. Consider the solution

$$x_1 = \ldots = x_m = \frac{2}{m}$$

for the LP relaxation; clearly, $m \geq 2$ therefore $\forall j \in [m] \quad 0 \leq x_j \leq 1$, and

$$\forall e_A \in \mathcal{U} \quad \sum_{\substack{j \in [m]: \\ e_A \in S_j}} x_j = \sum_{\substack{j \in [m]: \\ e_A \in S_j}} \frac{2}{m} = |A| \cdot \frac{2}{m} = \frac{m}{2} \cdot \frac{2}{m} = 1 \geq 1$$

hence this is a feasible solution to the LP, and its value is simply given by $m \cdot \frac{2}{m} = 2$. □

Claim: $\forall m \geq 2$ even $\quad \mathrm{IP}^*(\mathcal{U}_m, C_m) \geq \frac{1}{2}\log n - O(\log\log n)$.

Proof of the Claim. By way of contradiction, assume that there exists a sub-collection $S = \{S_{i_1}, \ldots, S_{i_k}\} \subseteq C_m$ with $k \leq \frac{m}{2}$ that covers $\mathcal{U}_m$. Consider the following set

$$T := [m] - \{i_1, \ldots, i_k\}$$

thus, we have that

$$|T| \geq m - \frac{m}{2} = \frac{m}{2}$$

hence, we can always define a subset $A \subseteq T$ such that $|A| = \frac{m}{2}$. However, we observe that

$$A \subseteq T \implies i_1, \ldots, i_k \notin A \implies e_A \notin S_{i_1} \cup \ldots \cup S_{i_k}$$

hence e_A is not covered by S ↯.

This means that for any set cover S of $(\mathcal{U}_m, C_m)$ it must hold that

$$|S| > \frac{m}{2} = \frac{1}{2}\log n - \Theta(\log\log n) \implies \mathrm{IP}^*(\mathcal{U}_m, C_m) \geq \frac{m}{2} = \frac{1}{2}\log n - O(\log\log n)$$

□

Finally, from these two claims, it follows that

$$\mathrm{IG}_{\mathrm{SC}}(\mathcal{U}_m, C_m) = \frac{\mathrm{IP}^*(\mathcal{U}_m, C_m)}{\mathrm{LP}^*(\mathcal{U}_m, C_m)} \geq \frac{\frac{1}{2}\log n - O(\log\log n)}{2} = \frac{1}{4}\log n - O(\log\log n)$$

which means that

$$\mathrm{IG}_{\mathrm{SC}} = \sup_{I \in \mathrm{SC}} \mathrm{IG}_{\mathrm{SC}}(I) \geq \mathrm{IG}_{\mathrm{SC}}(\mathcal{U}_m, C_m) \geq \frac{1}{4}\log n = \frac{1}{4\ln 2}\ln n$$

□

2.4 The Densest Subgraph problem

Up to this point, we only discussed NP-Complete problems, and various techniquesused to obtain approximate solutions. The next problem that we are going to introduce, instead, can be solved in **polynomial** time, thanks to linear programming. First, consider the following definition.

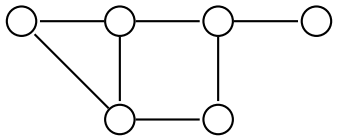
Definition 2.3: Graph density

Given a graph $G = (V, E)$ such that $V \neq \varnothing$, its **density** is defined as follows

$$\rho(G) := \frac{|E|}{|V|}$$

Note that we require $V \neq \varnothing$, but we will omit this detail in the following sections.

For instance, the density of the graph $G = (V, E)$ below



is evaluated through the following ratio

$$\rho(G) := \frac{|E|}{|V|} = \frac{7}{6} = 1.1\overline{6}$$

The problem that we are going to introduce is the following.

Definition 2.4: Densest Subgraph problem

The **Densest Subgraph** (DS) problem is defined as follows: given an undirected graph $G = (V, E)$, determine the induced subgraph of G that maximizes its density. In other words, the problem asks to find a set of vertices S^* in

$$S^* \in \arg\max_{S \subseteq V} \rho(G[S])$$

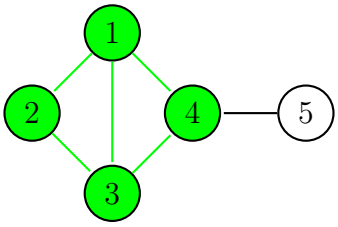


Figure 2.6: For instance, in this graph the set S that maximizes $\rho(G[S])$ is given by $S = \{1, 2, 3, 4\}$, which yields a density of $\frac{5}{4} = 1.25$.

The densest subgraph is relevant because it actually *maximizes* the average degree of its vertices. In fact, thanks to the handshaking lemma we have that

$$\rho(G[S]) := \frac{|E(G[S])|}{|S|} = \frac{1}{2} \cdot \frac{2 \cdot |E(G[S])|}{|s|} = \frac{1}{2} \cdot \frac{\sum_{v \in S} \deg(v)}{|S|} = \frac{1}{2} \operatorname*{avg}_{v \in S} \deg(v)$$

The LP that we will present which solves this problem is the following. However, to show that this LP actually yields a solution for DS is not trivial, and it will be proven through the following theorems.

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

$$\begin{aligned} x_{ij} &\leq y_i && \forall ij \in E \\ x_{ij} &\leq y_j && \forall ij \in E \\ \sum_{i \in V} y_i &\leq 1 \\ x_{ij} &\geq 0 && \forall ij \in E \\ y_i &\geq 0 && \forall i \in V \end{aligned}$$

Figure 2.7: LP for DS.

Lemma 2.4

Given a graph $G = (V, E)$, for any $S \subseteq V$ there exists a feasible solution $\{x_e, y_i\}_{e \in E, i \in V}$ to the LP having value at least $\rho(G[S])$.

Proof. Given a set $S \subseteq V$, construct the following solution:

- $\forall ij \in E \quad x_{ij} := \begin{cases} \frac{1}{|S|} & i, j \in S \\ 0 & i \notin S \vee j \notin S \end{cases}$
- $\forall i \in V \quad y_i := \begin{cases} \frac{1}{|S|} & i \in S \\ 0 & i \in V - S \end{cases}$

We observe that

- $ij \in E(G[S]) \implies i, j \in S \implies \begin{cases} x_{ij} = \frac{1}{|S|} \\ y_i = y_j = \frac{1}{|S|} \end{cases} \implies x_i j \leq y_i, y_j$
- $ij \notin E(G[S]) \implies i \notin S \vee j \notin S \implies x_{ij} = 0$, but since $y_i, y_j \geq 0$ for any $i, j \in V$ by definition, it holds that $x_{ij} \leq y_i, y_j$

Additionally, notice that

$$\sum_{i \in V} y_i = \sum_{i \in S} y_i + \sum_{i \in V - S} y_i = |S| \cdot \frac{1}{|S|} + |V - S| \cdot 0 = 1 \leq 1$$

This proves that this is a feasible solution to the LP. Lastly, the value of this solution is simply given by

$$\sum_{e \in E} x_e = \sum_{e \in E(G[S])} e + \sum_{e \notin E(G[S])} x_e = |E(G[S])| \cdot \frac{1}{|S|} + |E - E(G[S])| \cdot 0 = \frac{|E(G[S])|}{|S|} =: \rho(G[S])$$

□

In particular, this lemma proves that if S^* is an optimal solution for DS on a graph G, then there is a solution to this LP such that $\text{LP}^* \geq \rho(G[S^*])$. The next theorem will prove that $\text{LP}^* \leq \rho(G[S^*])$ holds as well, which implies that our LP does in fact produce an optimal solution to DS, and that $\text{IG}_{\text{DS}} = 1$.

Theorem 2.4

Given a graph $G = (V, E)$, for any $\{x_e, y_i\}_{e \in E, i \in V}$ feasible solution to the LP that has value v, there exists a set $S \subseteq V$ such that $\rho(G[S]) \geq v$.

Proof. Consider a feasible solution $\{x'_e, y'_i\}_{e \in E, i \in V}$ of the LP, having value v. We will construct another solution to the LP $\{x_e, y_i\}_{e \in E, i \in V}$ starting from the previous solution, as follows:

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

It is easy to see that this is a feasible solution to the LP, in fact

$$\forall ij \in E \quad x_{ij} := \min(y_i, y_j) \leq y_i, y_j$$

and additionally, by feasibility of $\{x'_e, y'_i\}_{e \in E, i \in V}$, we have that

$$\sum_{i \in V} y_i = \sum_{i \in V} y'_i \leq 1$$

Furthermore, note that

$$v = \sum_{e \in E} x'_e \leq \sum_{e \in E} \min(y'_i, y'_j) = \sum_{e \in E} \min(y_i, y_j) = \sum_{e \in E} x_e$$

meaning that the value of this solution is at least v, but it may be better — this happens if $x_{ij} < y_i$ or $x_{ij} < y_j$ for some $ij \in E$, and if this is the case we say that there is some *slack*.

Next, we will construct a series of sets of vertices starting from this new LP solution we just defined. In particular, given a value $r \geq 0$, let

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

$$E(r) := \{e \mid x_e \geq r\}$$

Notice that $ij \in E(r) \iff i, j \in S(r)$, because

$$ij \in E(r) \iff x_{ij} \geq r \iff \min(y_i, y_j) \geq r \iff y_i, y_j \geq r \iff i, j \in S(r)$$

which means that $S(r)$ and $E(r)$ are well-defined — note that $E(G[S(r)]) = E(r)$.

Claim: There exists a value $r^* \geq 0$ such that $\rho(G[S(r^*)]) \geq v$.

Proof of the Claim. Consider the solution $\{x_e, y_i\}_{e \in E, i \in V}$ to the LP that we constructed previously, and let π be the permutation of the y_i's such that

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

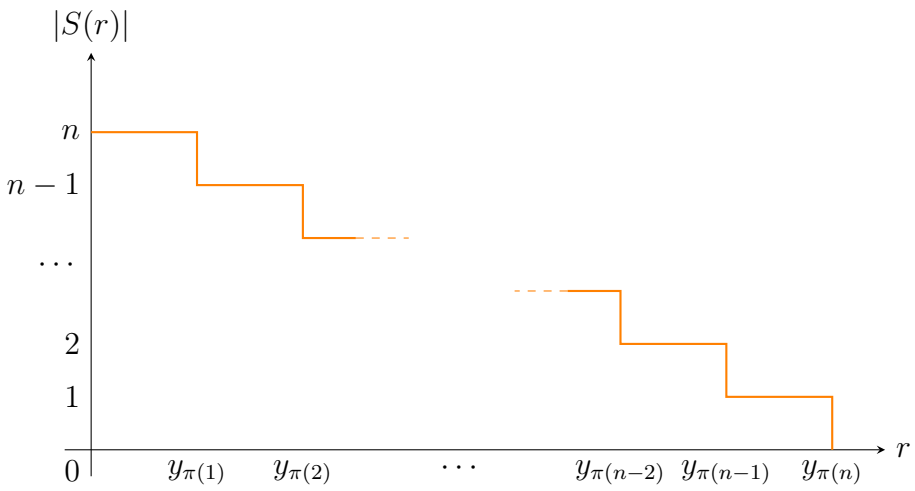
which means that π defines a sorting of the y_i's. Now, consider the possible values of $r \geq 0$, and what happens inside $S(r)$:

- for $r = 0$, by feasibility of our solution we have that

$$S(r) = S(0) := \{i \mid y_i \geq 0\} \implies |S(r)| = n$$

- notice that for any value $0 \leq r \leq y_{\pi(1)} \leq \ldots \leq y_{\pi(n)}$, the value of $|S(r)|$ will still be n, by its definition
- however, if $0 \leq y_{\pi(1)} < r \leq \ldots \leq y_{\pi(n)}$, then the vertex $\pi(1)$ will be the only verterx not contained in $S(r)$, therefore $|S(r)| = n - 1$

Repeating the same argument for all the possible values of r, we obtain the following graph for $|S(r)|$.



From this graph, it is easy to see that computing the integral of $|S(r)|$ is trivial, because

$$\begin{aligned}
\int_0^{y_{\pi(n)}} |S(r)|\, dr &= \int_0^{y_{\pi(1)}} |S(r)|\, dr + \int_{y_{\pi(1)}}^{y_{\pi(2)}} |S(r)|\, dr + \ldots + \int_{y_{\pi(n-1)}}^{y_{\pi(n)}} |S(r)|\, dr \\
&= \int_0^{y_{\pi(1)}} n\, dr + \int_{y_{\pi(1)}}^{y_{\pi(2)}} (n-1)\, dr + \ldots + \int_{y_{\pi(n-1)}}^{y_{\pi(n)}} 1\, dr \\
&= n \cdot (y_{\pi(1)} - 0) + (n-1) \cdot (y_{\pi(2)} - y_{\pi(1)}) + \ldots + 1 \cdot (y_{\pi(n)} - y_{\pi(n-1)}) \\
&= y_{\pi(1)} \cdot [n - (n-1)] + y_{\pi(2)} \cdot [(n-1) - (n-2)] + \ldots + y_{\pi(n)} \cdot (1 - 0) \\
&= y_{\pi(1)} \cdot 1 + y_{\pi(2)} \cdot 1 + \ldots + y_{\pi(n)} \cdot 1 \\
&= \sum_{i=1}^{n} y_{\pi(i)} \\
&= \sum_{i \in V} y_i \\
&\leq 1 \qquad \text{(by feasibility of the solution)}
\end{aligned}$$

By the same argument, we can derive that

$$\int_0^{y_{\pi(n)}} |E(r)|\, dr = \sum_{e \in E} x_e \geq v$$

Note that the last inequality was proved before the claim.

Lastly, we will prove the statement of the claim by way of contradiction. In particular, suppose that for all $r \geq 0$ it holds that $\rho(G[S(r)]) < v$. We can rewrite this inequality as follows

$$\rho(G[S(r)]) := \frac{|E(r)|}{|S(r)|} < v \iff |E(r)| < v \cdot |S(r)|$$

This inequality can be used to upper bound the integral of $|E(r)|$ *point-by-point*, i.e.

$$v \leq \int_0^{y_{\pi(n-1)}} |E(r)|\, dr < \int_0^{y_{\pi(n-1)}} v \cdot |S(r)|\, dr = v \cdot \int_0^{y_{\pi(n-1)}} |S(r)|\, dr \leq v \cdot 1 = v$$

meaning that $v < v$ ↯. □

Finally, since we proved that $G[S(r)]$ is well-defined, and this claim proves that there exists an r^* such that $\rho(G[S(r^*)]) \geq v$, the statement holds. □

2.4.1 Approximation through duality

As we discussed at the beginning of this section, the cost of solving an LP generally depends on the number of *variables* and *constraints* it has. In particular, the LP we presented for solving DS has a *large* number of both, which means that while the cost remains *polynomial*, the degree of the polynomial is too high, making this approach **impractical** for large graphs.

Now, we are going to present a **greedy algorithm**, developed by Charikar [Cha00] in 2000, and we will prove that this algorithm yields a $\frac{1}{2}$-approximation of DS.

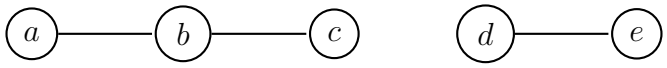
Algorithm 2.2: Charikar's algorithm

Given an undirected graph $G = (V, E)$, the algorithm returns a $\frac{1}{2}$-approximation solution of DS on G.

1: **function** CHARIKAR(G)
2: $S_0 := V(G)$
3: **for** $i \in [n-1]$ **do**
4: $v_i \in \arg\min_{v \in S_{i-1}} \deg_{G[S_{i-1}]}(v)$
5: $S_i := S_{i-1} - \{v_i\}$
6: **end for**
7: **return** $S^* \in \arg\max_{i \in [0,n-1]} \rho(G[S_i])$
8: **end function**

Idea. The algorithm construct a series $S_0, \ldots, S_n$ of sets of vertices, such that $|S_i| = n - i$, and at each iteration the i-th vertex that gets removed from $G[S_{i-1}]$ is the one having minimum degree. At the end, the algorithm returns the set S^* that maximizes the density of $G[S^*]$.

We observe that this algorithm does not return an optimal solution in general; in particular, problems can arise if there are vertices of equal degree. For instance, consider the following disconnected graph



In this graph, the densest subgraph is induced by the set $S^* = \{a, b, c\}$, because

$$\rho(G[S^*]) = \frac{2}{3}$$

However, since $\deg(a) = \deg(c) = \deg(d) = \deg(e) = 1$, the algorithm may choose $v_1 := a$, meaning that $G[S_1]$ would be

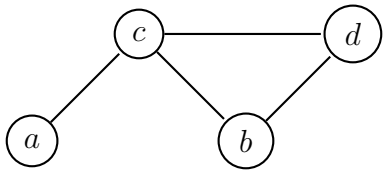
and it is easy to see that $G[S_0] = G$, and any other subgraph of this graph, have lower density than $G[S^*]$.

To prove that this algorithm yields a $\frac{1}{2}$-approximation, we are going to introduce some definitions first.

Definition 2.5: Orientation

Given an undirected graph $G = (V, E)$, an **orientation** of G is a function $\phi : E \to V$ that assigns to each edge $\{u, v\} \in E$ either u or v.

Note that, by definition for any $e \in E$ it holds that $\phi(e) \in e$. For instance, given the following graph



and the following function ϕ

e	$\phi(e)$
ac	c
bc	c
bd	b
cd	d

we would get the following orientation on G

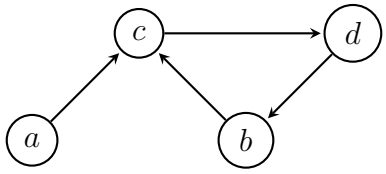


Figure 2.8: The previous graph, oriented through ϕ.

Given an orientation ϕ of G, and a vertex $v \in V(G)$ we will indicate with $\deg_\phi(v)$ the in-degree of v w.r.t. ϕ, i.e.

$$\deg_\phi(v) := |\{e \in E \mid \phi(e) = v\}|$$

For example, in the previous graph we have that $\deg_\phi(c) = 2$, $\deg_\phi(a) = 0$ and $\deg_\phi(b) = \deg_\phi(d) = 1$. By the same argument that proves the handshaking lemma, we observe that

$$\sum_{v \in V(G)} \deg_\phi(v) = |E|$$

Lastly, the maximum in-degree w.r.t. ϕ will be denoted as

$$\Delta_\phi := \max_{v \in V} \deg_\phi(v)$$

We are now ready to prove the approximation ratio of Charikar's algorithm.

Theorem 2.5: Charikar's approximation ratio

Given a graph G, and an optimal solution $\rho(G[S^*])$ for DS on G, let $M :=$ Charikar(G); then, it holds that $\frac{1}{2}\rho(G[S^*]) \leq M$.

Proof. We are going to prove this approximation ratio through two claims.

Claim: Given an undirected graph G, and an optimal solution $\rho(G[S^*])$ for DS on G, for any orientation ϕ of G it holds that $\rho(G[S^*]) \leq \Delta_\phi$.

Proof of the Claim. By definition of ϕ, each $uv \in E$ is going to be oriented towards either u or v, therefore if $uv \in E(G[S^*])$ then uv is going to be oriented towards a node of S. Hence, we have that

$$|E(G[S^*])| \leq \sum_{v \in V(G[S^*])} \deg_\phi(v) \leq \sum_{v \in V(G[S^*])} \Delta_\phi = |V[G(S^*)]| \cdot \Delta_\phi \iff \rho(G[S^*]) := \frac{|E(G[S^*]|)}{|V(G[S^*])|} \leq \Delta_\phi$$

□

Consider Charikar's algorithm; before proving the other claim, we need to introduce an orientation ϕ_{GR} — where GR stands for *greedy* — of G defined as follows: if v_i is the i-th vertex — removed from $G[S_{i-1}]$ — picked by the algorithm, orient any edge $\{u, v_i\} \in E(G)$ towards v_i, for any $u \in V(G)$. For instance, Charikar's algorithm applied on the graph shown in Figure 2.8 would set $v_1 := a$, therefore $\phi(ca) = a$, and so on. This implies that

$$\forall i \in [n] \quad \deg_{G[S_{i-1}]}(v_i) = \deg_{\phi_{\text{GR}}}(v_i)$$

Claim: Given an undirected graph G, and $M := \textsc{Charikar}(G)$, it holds that $\Delta_{\phi_{\text{GR}}} \leq 2M$.

Proof of the Claim. Observe that for any $i \in [n]$ it holds that

$$v_i \in \arg\min_{v \in S_{i-1}} \deg_{G[S_{i-1}]}(v) \iff \deg_{G[S_{i-1}]}(v_i) = \min_{v \in S_{i-1}} \deg_{G[S_{i-1}]}(v)$$

therefore, we have that

$$\begin{aligned}
\deg_{\phi_{\text{GR}}}(v_i) &= \deg_{G[S_{i-1}]}(v_i) \\
&= \min_{v \in S_{i-1}} \deg_{G[S_{i-1}]}(v) \\
&\leq \operatorname{avg}_{v \in S_{i-1}} \deg_{G[S_{i-1}]}(v) \\
&= \frac{\displaystyle\sum_{v \in S_{i-1}} \deg_{G[S_{i-1}]}(v)}{|S_{i-1}|} \\
&= \frac{2\,|E(G[S_{i-1}])|}{|V(G[S_{i-1}])|} \\
&=: 2\rho(G[S_{i-1}])
\end{aligned}$$

Finally, we get that

$$\begin{aligned}
\Delta_{\phi_{\text{GR}}} &:= \max_{v \in V(G)} \deg_{\phi_{\text{GR}}}(v) \\
&= \max_{i \in [n]} \deg_{\phi_{\text{GR}}}(v_i) \\
&\leq \max_{i \in [n]} 2\rho(G[S_{i-1}]) \qquad \text{(for the previous inequality)} \\
&= 2 \max_{i \in [n]} \rho(G[S_{i-1}]) \\
&\leq 2 \max_{i \in [0, n-1]} \rho(G[S_i]) \\
&=: 2M
\end{aligned}$$

□

Lastly, putting the two claims together, we get that for any optimal solution $\rho(G[S^*])$ of a given undirected graph G, if $M := \textsc{Charikar}(G)$ then

$$\rho(G[S^*]) \leq \Delta_{\phi_{\text{GR}}} \leq 2M \iff \tfrac{1}{2}\rho(G[S^*]) \leq M$$

□

In this proof, we defined ϕ_{GR}, which helped us determine the approximation ratio of the algorithm. But how did Charikar come up with such an orientation of the graph? To answer this question, we need to introduce the concept of the **duality** of linear programs.

Consider the following *maximization* generic linear program:

$$\begin{gathered}
\max \; c_1 x_1 + \ldots + c_n x_n \\
\\
a_{11} x_1 + \ldots + a_{1n} x_n \leq b_1 \\
\vdots \\
a_{m1} x_1 + \ldots + a_{mn} x_n \leq b_m \\
x_1, \ldots, x_n \geq 0
\end{gathered}$$

Figure 2.9: A *primal* LP.

As a convention, we will call the **primal** LP the one that *maximizes* its objective function. Hence, given a primal LP, its **dual** linear program is defined as follows:

$$\min \; b_1 y_1 + \ldots + b_m y_m$$

$$\begin{gathered} a_{11} y_1 + \ldots + a_{m1} y_m \geq c_1 \\ \vdots \\ a_{1n} y_1 + \ldots + a_{mn} y_m \geq c_n \\ y_1, \ldots, y_m \geq 0 \end{gathered}$$

Figure 2.10: The *dual* of a primal LP.

In other words, given a matrix $A \in \mathbb{R}^{m \times n}$, two vectors $c, x \in \mathbb{R}^n$, and two vectors $b, y \in \mathbb{R}^m$, these two programs can be rewritten as follows:

Primal LP		Dual LP
$\max \; c^T x$	$\Longleftrightarrow$	$\min \; b^T y$
$Ax \leq b$		$A^T y \geq c$
$x \geq 0$		$y \geq 0$

Observe that, by definition, the dual of the dual of a primal LP is the primal LP itself. Moreover, we can relate the solutions of primal LPs and their duals through the following two theorems.

Theorem 2.6: Weak duality theorem

Given a feasible solution x for a primal LP, and a feasible solution y for its dual LP, such that both x and y exist and are finite, it holds that

$$c^T x \leq b^T y$$

Proof. Given any two matrices $N \in \mathbb{R}^{m \times n}$, $M \in \mathbb{R}^{n \times l}$ it holds that

$$(NM)^T = M^T N^T$$

therefore, we have that

$$\begin{aligned} c^T x &= x^T c \\ &\leq x^T A^T y && \text{(by feasibility of } y\text{)} \\ &= (Ax)^T y && \text{(by the previous observation)} \\ &\leq b^T y && \text{(by feasibility of } x\text{)} \end{aligned}$$

□

Theorem 2.7: Strong duality theorem

Given an optimal solution x for a primal LP, and an optimal solution y for its dual LP, such that both x and y exist and are finite, it holds that

$$c^T x = b^T y$$

Going back to our problem, consider again the LP for DS presented in Figure 2.7. We observe that its dual LP is the following linear program:

$$\begin{aligned} \min\ & \delta \\ & \\ \gamma_{ij} + \gamma_{ji} &\geq 1 && \forall ij \in E \\ \delta - \sum_{\substack{j \in V: \\ ij \in E}} \gamma_{ij} &\geq 0 && \forall i \in V \\ \delta &\geq 0 \\ \gamma_{ij} &\geq 0 && \forall ij \in E \\ \gamma_{ji} &\geq 0 && \forall ij \in E \end{aligned}$$

Figure 2.11: The dual of the LP for DS.

Notice that this is the LP that corresponds to the **Minimum Max-Degree Orientation** problem, which asks to find the edge orientation with the lowest possible maximum degree. In fact, observe that

$$\forall i \in V \quad \delta - \sum_{\substack{j \in V: \\ ij \in E}} \gamma_{ij} \geq 0 \iff \delta \geq \deg(i)$$

TODO

missing last part

Even though this approach yields a $\frac{1}{2}$-approximation algorithm that runs in only n iterations, it may still be too slow for very large graphs. Can we improve the algorithm further? Consider how CHARIKAR algorithm works: at each iteration, it removes the vertex with the minimum degree from the current graph. What if, instead of removing just one such vertex per iteration, we remove all vertices of minimum degree *at once*?

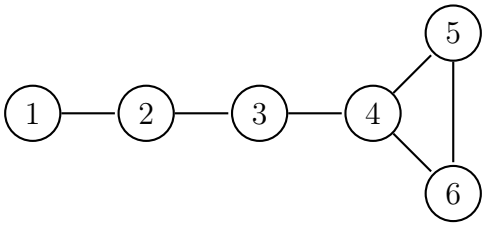
Algorithm 2.3: Charikar's algorithm (improved version)

Given an undirected graph $G = (V, E)$, the algorithm returns a $\frac{1}{2}$-approximation solution of DS on G.

---

function CharikarImproved(G)
  $S_0 := V(G)$
  $i := 0$
  while $S_i \neq \varnothing$ **do**
    $A_i := \arg\min_{u \in S_i} \deg_{G[S_i]}(u)$
    $S_{i+1} := S_i - A_i$
  end while
  return $S^* \in \arg\max_{i \in \mathbb{N}} \rho(G[S_i])$
end function

This algorithm is able to reduce significantly the number of iterations, by removing the vertices of minimum degree from S_i all at once. However, in the worst case the number of iterations is still n. For instance, consider the following graph



In this graph, our improved version of Charikar would still need n iterations to remove all the vertices — in the figure, the node labeled with i is the only node present in A_i at the i-th iteration of the algorithm. Can we do better?

Consider the proof of Theorem 2.5; the second claim of the theorem uses the fact that the vertex v_i has minimum degree in $G[S_{i-1}]$. However, note that the only reason why we need this fact is to bound

$$\deg_{G[S_{i-1}]}(v_i) \leq \operatorname*{avg}_{v \in S_{i-1}} \deg_{G[S_{i-1}]}(v_i)$$

Hence, we may try to use the *average degree* instead of the *minimum degree*, in the definition of A_i.

Consider a path P_n; observe that CharikarImproved(P_n) would require $\left\lceil \frac{n}{2} \right\rceil$ iteration to remove all the vertices from the initial graph, because it would remove at most (for the case when n is odd) 2 vertices per iteration — i.e. the endpoints. What happens if we replace the definition of A_i as follows?

$$A_i := \left\{ v \in S_i \mid \deg_{G[S_i]}(v) \leq \operatorname*{avg}_{u \in S_i} \deg_{G[S_i]}(u) \right\}$$

In the case of P_n, CharikarImproved would still fail at processing the graph all at once, because the average degree is *less than* 2. Nonetheless, this problem suggest the following approach.

Algorithm 2.4: Charikar's algorithm (ε version)

Given an undirected graph $G = (V, E)$, the algorithm returns a $\frac{1}{2(1+\varepsilon)}$-approximation solution of DS on G.

1: **function** $\text{CHARIKAR}_\varepsilon(G)$
2: $\quad S_0 := V(G)$
3: $\quad i := 0$
4: $\quad$ **while** $S_i \neq \varnothing$ **do**
5: $\quad\quad A_i := \left\{ v \in S_i \mid \deg_{G[S_i]}(v) \leq (1+\varepsilon) \operatorname{avg}_{u \in S_i} \deg_{G[S_i]}(u) \right\}$
6: $\quad\quad S_{i+1} := S_i - A_i$
7: $\quad$ **end while**
8: $\quad$ **return** $S^* \in \arg\max_{i \in \mathbb{N}} \rho(G[S_i])$
9: **end function**

Theorem 2.8: $\text{CHARIKAR}_\varepsilon$'s approximation ratio

Given a graph G, and $S := \text{CHARIKAR}_\varepsilon(\text{G})$, let S^* be an optimal solution to DS on G. Then, for all $\varepsilon > 0$ it holds that $|S| \geq \frac{1}{2(1+\varepsilon)} |S^*|$. Moreover, the algorithm runs in at most $O(\log_{1+\varepsilon}(n))$ iterations.

Proof. Consider the optimal solution S^*; note that $S^* \neq \varnothing$ because $\rho(G[\varnothing])$ is not defined, and if $|S^*| = 1$, then $|E(G[S^*])| = 0 \implies \rho(G[S^*]) = 0$ meaning that . Hence, we may assume that $|S^*| \geq 2$.

non ho capito

Claim: $\forall v \in S^* \quad \deg_{G[S^*]}(v) \geq \rho(G[S^*])$

Proof of the Claim. Fix a vertex $v \in S^*$. By optimality of S, we know that $\rho(G[S^*]) \geq \rho(G[S^* - \{v\}])$. Moreover, note that

$$\rho(G[S^* - \{v\}]) = \frac{|E(G[S^* - \{v\}])|}{|S^* - \{v\}|} = \frac{|E(G[S^*])| - \deg_{G[S^*]}(v)}{|S^*| - 1}$$

Therefore, we have that

$$\rho(G[S^*]) \geq \rho(G[S^* - \{v\}]) \iff \frac{|E(G[S^*])|}{|S^*|} \geq \frac{|E(G[S^*])| - \deg_{G[S^*]}(v)}{|S^*| - 1}$$

and the claim follows by solving the inequality.

TODO □

da finire

□

TODO buco

2.5 The Unique Games Conjecture

Before discussing the conjecture itself, consider the following definition.

Definition 2.6: Unique label cover

Let G be a bipartite graph, bipartitioned through (A, B); given a value $k \in \mathbb{N}$, for each edge $e \in E(G)$ let $\pi_e : [k] \to [k]$ be a permutation. A **unique label cover** (ULC) of G is an assignment $\phi : A \cup B \to [k]$ defining the set S_ϕ of satisfied edges:

$$S_\phi := \{ab \in E(G) \mid a \in A, b \in B, \pi_{ab}(\phi(a)) = \phi(b)\}$$

In other words, an edge $ab \in E(G)$ of G is sait to be *satisfied* by ϕ if the color $\phi(a)$ gets permuted into the color $\phi(b)$ by the permutation π_{ab}.

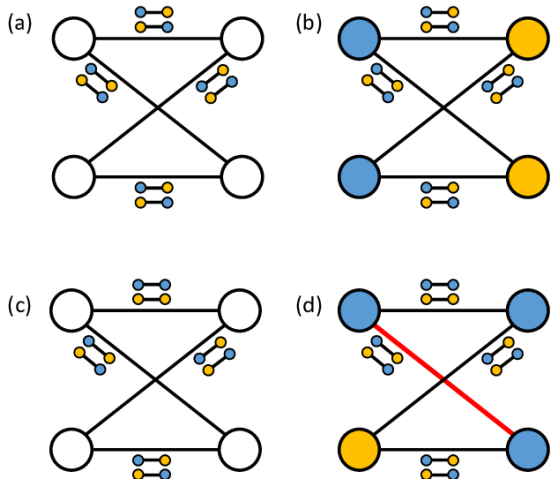


Figure 2.12: In this figure, (a) and (b) are two instances of the UCL problem with 2 colors; (b) is a solution to (a) that satisfies all the edges, while (d) is a solution to (c) with an unsatisfied edge.

Note that UCL instances are *strongly constrainted*, i.e. the color of a vertex uniquely defines the colors of its neighbors, and therefore its entire connected component. Thus, if the input instance admits a valid assignment, such assignment can be found *efficiently* by iterating over all the possible colors of a single node. In particular, this implies that the problem of deciding if a given instance admits a satisfying assignment can be solved in polynomial time.

The **value** of a UCL instance is the ratio of the edges that are satisfiable by any assignment. Hence, for satisfiable instances the ratio is 1, and we can find a satisfiying assignment as described. In contrast, determining the value of an unsatisfiable game — even approximately — appears to be very hard. This difficulty was formalized by Khot [Kho02] in terms of NP-hardness, as follows.

Conjecture 2.1: Unique Games Conjecture (UGC)

It is conjectured that for each $\varepsilon > 0$ there is a value k_ε for which it is $\mathsf{NP} - Hard$ to determine if for a UCL instance with k_ε labels one of the following holds:

- at most an ε-fraction of the edges are satisfied
- at least a $(1-\varepsilon)$-fraction of the edges are satisfied

Furthermore, Khot's conjecture has been shown to be linked with Constraint Satisfaction Problems, a particular type of optimization problems defined as follows.

Definition 2.7: Constraint Satisfaction Problem

Let $\mathcal{P}$ be a set of k-ary predicates defined on $[q]$, for $q, k \in \mathbb{N}$. An instance of a **Constraint Satisfaction Problem** (CSP) is a set of variables $X = \{x_1, \ldots, x_n\}$ and a set of constraints $C_1, \ldots, C_m$ such that $C_j = \langle I_j, P_j \rangle$ where $I_j \subseteq [n]$ and $P_j \in \mathcal{P}$.

A constraint $C_j = \langle I_j, P_j \rangle$ is said to be *satisfied* by an assignment $\alpha : I_j \to \{0,1\}$ if $P_j \begin{bmatrix} x_{i_1} & \ldots & x_{i_k} \\ \alpha(x_{i_1}) & \ldots & \alpha(x_{i_k}) \end{bmatrix}$ evaluates to true.

TODO buco

Given a problem that can be described through a CSP $\mathcal{C}$, let $\alpha(\mathcal{C})$ be the best possible approximation ratio for $\mathcal{C}$, i.e. the minimum ratio such that the problem does not become NP-Hard to approximate. The following theorem, proved by Raghavendra [Rag08], shows that the relationship between approximation algorithms, SDPs, and the UGC is as strong as it can be.

Theorem 2.9: Raghavendra's theorem

For every CSP $\mathcal{C}$ with values in $[q]$ and arity k, the following hold.

- There is an SDP with IG of $\alpha(\mathcal{C})$ and a rounding algorithm that is an $\alpha(\mathcal{C})$-approximation of $\mathcal{C}$.
- If the UGC is true, it is NP-Hard to approximate $\mathcal{C}$ with a ratio $\alpha(\mathcal{C}) - \varepsilon$ for any $\varepsilon > 0$.

In particular, if the UGC holds true, Raghavendra's theorem implies that the SDP integrality gap for any problem that can be described through a CSP is the best possible approximation ratio that can be achieved. Although many researchers believe in the validity of the conjecture, it remains unproven.

As a direct corollary of Raghavendra's theorem, we get that the approximation ratios that we previously found for MC and VC are the best possible ones, if the UGC is true.

Corollary 2.1

If the UGC is true, then $\mathrm{IG}_{\mathrm{MC}}^{\mathrm{SDP}} = \alpha(\mathrm{MC})$ and $\mathrm{IG}_{\mathrm{VC}} = \alpha(\mathrm{VC})$.

3

Metric geometry

TODO

missing introduction

First, consider the following definition

Definition 3.1: Set spartity

Given a graph $G = (V, E)$, and a set $\varnothing \subset S \subseteq V$, the **sparsity** of S is defined as follows:

$$\psi(G) := \frac{|\text{cut}(S)|}{\left|S \times \overline{S}\right|} = \frac{|\text{cut}(S)|}{|S| \cdot \left|\overline{S}\right|}$$

As for the case of *graph density*, we require $S \neq \varnothing$, but we will omit this detail in the following sections. Moreover, note that by definition of *cut*, we have that for any $S \subseteq V(G)$ it holds that

$$0 \leq |\text{cut}(S)| \leq \left|S \times \overline{S}\right| = |S| \cdot \left|\overline{S}\right|$$

since an edge e is in $\text{cut}(S)$ if and only if $|S \cap e| = 1$, hence the cartesian product $S \times \overline{S}$ represents the edges that have exactly one endpoint in S. Therefore, we have that $0 \leq \psi(S) \leq 1$, and in particular

- $\psi(S) = 0 \implies |\text{cut}(S)| = 0$ which happens if S is *disconnected* from $\overline{S}$
- $\psi(S) = 1 \implies |\text{cut}(S)| = \left|S \times \overline{S}\right|$ which happens if S is "*fully connected*" to $\overline{S}$, i.e. every edge of S is connected to every edge of $\overline{S}$.

Given this metric, we are interested in finding the subset of vertices of a given graph that *minimizes* its sparsity.

Definition 3.2: Sparsest Cut problem

The **Sparsest Cut** (SC) problem is defined as follows: given a graph $G = (V, E)$, determine the subset of vertices that minimizes its sparsity. In other words, the problems asks to finde a set S^* in

$$S^* \in \arg\min_{S \subseteq V} \psi(S)$$

In order to reason about SC, we are going to introduce the following type of graph, invented by Erdős and Rényi [ER22] in 1959.

Definition 3.3: Erdős-Rényi random graph

Given two values $n \in \mathbb{N}$ and $\mu \in [0, 1]$, the Erdős-Rényi random graph $G(n, \mu)$ is a graph that has a fixed vertex set $V(G) = [n]$, and a *probabilistic* edge set $E(G)$ such that

$$\forall e \in [V(G)]^2 \quad \Pr[e \in E(G)] = \mu$$

By definition, this model represents an *evenly sparse graph*, because for each $S \subseteq V(G)$ it holds that

$$\mathbb{E}[\psi(S)] = \frac{\mathbb{E}\left[|\mathrm{cut}(S)|\right]}{|S| \cdot \left|\overline{S}\right|} = \frac{\sum_{ij \in S \times \overline{S}} \Pr\left[ij \in \mathrm{cut}(S)\right]}{|S| \cdot \left|\overline{S}\right|} = \frac{\mu \cdot |S| \cdot \left|\overline{S}\right|}{|S| \cdot \left|\overline{S}\right|} = \mu$$

The Maximum Cut problem — which we discussed in Section 1.1.1 — can be reduced to SC, implying that the latter is NP-Hard as well. Nevertheless, the problem can be approximated through the **Leighton-Rao algorithm** [LR88], which relies on LP relaxation and a series of reduction steps. In particular, its integrality gap is bounded using *geometric arguments*, therefore in the following sections we will focus on **metric geometry** in order to establish this result.

3.1 Metrics

3.1.1 Cut metrics

A **metric** (or *distance function*) provides a way to quantify how far apart objects are within a given space, formalizing the concept of *distance*. In particular, in geometry we have the following definition.

Definition 3.4: Metric

Given a set S, a **metric** on S is a function $d : S \times S \to \mathbb{R}$ that satisfies

1. *non-negativity*: $\forall x, y \in S \quad d(x, y) \geq 0$
2. *simmetry*: $\forall x, y \in S \quad d(x, y) = d(y, x)$
3. *self-distance*: $\forall x \in S \quad d(x, x) = 0$
4. *triangle inequality*: $\forall x, y, z \in S \quad d(x, y) \leq d(x, z) + d(z, y)$

The typical example of metric is the Euclidean distance, which defines d as follows

$$d(x, y) := ||x - y||$$

In particular, for our discussion we are interested in *cut metrics*, defined as follows.

Definition 3.5: Elementary cut metric

Given a graph $G = (V, E)$, and a set $T \subseteq V$, the **elementary cut metric of** T is a function $d_T : V \times V \to \mathbb{R}$ such that

$$d_T(x, y) := \begin{cases} 1 & |T \cap \{x, y\}| = 1 \\ 0 & \text{otherwise} \end{cases}$$

In other words, for a given set of vertices T, the elementary cut metric $d_T(x, y)$ is equal to 1 if and only if the edge xy is in $\text{cut}(T)$.

Proposition 3.1

Any elementary cut metric is a metric.

Proof. Given a graph G, and a susbset $T \subseteq V(G)$, consider the elementary cut metric d_T; we observe that, by definition

- *non-negativity* is satisfied, because

$$\forall x, y \in V(G) \quad d_T(x, y) = 0 \vee d_T(x, y) = 1 \implies d_T(x, y) \geq 0$$

- *simmetry* is satisfied, because

$$\forall x, y \in V(G) \quad |T \cap \{x, y\}| = |T \cap \{y, x\}| \implies d_T(x, y) = d_T(y, x)$$

- *self-distance* is satisfied, because

$$\forall x \in V(G) \quad \nexists \{x, x\} \in E(G) \implies d_T(x, x) = 0$$

hence, we just need to prove that the *triangle inequality* is also satisfied by d_T. Therefore, fix three vertices $x, y, z \in V(G)$; we have three cases:

- if $x, y, z \in T$ then

$$0 = d_T(x, y) \leq d_T(x, z) + d_T(z, y) = 0 + 0 = 0$$

- if $x, y, z \notin T$ then

$$0 = d_T(x, y) \leq d_T(x, z) + d_T(z, y) = 0 + 0 = 0$$

- if $\exists A \in \{T, \overline{T}\}$ such that *exactly one* of x, y, z lies in A, we have three sub-cases:
  - if $x \in A$ then

$$1 = d_T(x, y) \leq d_T(x, z) + d_T(z, y) = 1 + 0 = 1$$

  - if $y \in A$ then

$$1 = d_T(x, y) \leq d_T(x, z) + d_T(z, y) = 0 + 1 = 1$$

  - if $z \in A$ then

$$0 = d_T(x, y) \leq d_T(x, z) + d_T(z, y) = 1 + 1 = 2$$

□

We observe that, by definition, for any T it holds that $d_T(x, y) = d_{\overline{T}}(x, y)$. In fact, if $\{x, y\} \in \text{cut}(T)$ then $|T \cap \{x, y\}| = 1$, meaning that exactly one endpoint of $\{x, y\}$ lies in T, which must imply that the other endpont lies in $\overline{T}$. Vice versa, if $\{x, y\} \notin \text{cut}(x, y)$, then either $x, y \in T$ — and therefore $x, y \notin \overline{T} \implies \left|\overline{T} \cap \{x, y\}\right| \neq 1$ — or $x, y \notin T$ — and therefore $x, y \in \overline{T} \implies \left|\overline{T} \cap \{x, y\}\right| \neq 1$ as well.

But why are we discussing cut metrics in the first place? We are going to show that the concept of elementary cut metric is *deeply* related to the concept of *sparsity* previously introduced. First, consider the following definition.

Definition 3.6: Cut-ratio

Given a graph G, and a subset $T \subseteq V(G)$, the **cut-ratio** induced by the elementary cut metric d_T is defined as

$$\phi(d_T) := \frac{\sum_{xy \in E(G)} d_T(x, y)}{\sum_{xy \in [V(G)]^2} d_T(x, y)}$$

Through some algebraic manipulation, we see that the cut-ratio of d_T is in fact the sparsity of T, as shown in the following proposition.

Proposition 3.2

Given a graph G, and a subset $T \subseteq V(G)$, it holds that $\phi(d_T) = \psi(T)$.

Proof. We observe that

$$\phi(d_T) := \frac{\sum_{ij \in E(G)} d_T(x,y)}{\sum_{ij \in [V(G)]^2} d_T(x,y)} = \frac{\sum_{xy \in E(G)} \mathbb{1}[x \in T \oplus y \in T]}{\sum_{xy \in [V(G)]^2} \mathbb{1}[x \in T \oplus y \in T]} = \frac{\text{cut}(S)}{|S| \cdot |\overline{S}|} = \psi(T)$$

□

We can generalize the concept of elementary cut metric through **cut metrics**, i.e. linear combinations over a given set of elementary cut metrics, as defined below.

Definition 3.7: Cut metric

Given a graph G, a sequence $d_{T_1}, \dots, d_{T_k}$ of elementary cut metrics on G, and k positive real values $\lambda_1, \dots, \lambda_k > 0$, a **cut metric** is a function defined as follows

$$d : V(G) \times V(G) \to \mathbb{R} : (x,y) \mapsto \sum_{i \in [k]} \lambda_i d_{T_i}(x,y)$$

As for elementary cut metrics, it can be easily proven that cut metrics are indeed metrics — we will omit the proof. Moreover, the following proposition shows that the cut-ratio of a cut metric d is lower bounded by the smallest cut-ratio among the elementary cut metrics that define d itself.

Proposition 3.3

Given a graph G, and a cut metric d defined through $d_{T_1}, \dots, d_{T_k}$, it holds that

$$\phi(d) \geq \min_{j \in [k]} \phi(d_{T_j})$$

Proof. First, consider the following property of sums.

Claim: For any $a_1, \dots, a_k, b_1, \dots, b_k > 0$, it holds that

$$\frac{\sum_{i \in [k]} a_i}{\sum_{i \in [k]} b_i} \geq \min_{j \in [k]} \frac{a_j}{b_j}$$

Proof of the Claim. We observe that

$$\frac{\sum_{i \in [k]} a_i}{\sum_{i \in [k]} b_i} = \frac{\sum_{i \in [k]} \frac{b_i}{b_i} \cdot a_i}{\sum_{i \in [k]} b_i} \geq \frac{\sum_{i \in [k]} b_i \cdot \min_{j \in [k]} \frac{a_j}{b_j}}{\sum_{i \in [k]} b_i} = \min_{j \in [k]} \frac{a_j}{b_j}$$

□

Let $\lambda_1, \dots, \lambda_k > 0$ be the coefficients that define d; then, thanks to the claim, we get

$$\begin{aligned}
\phi(d) &= \frac{\sum_{xy \in E(G)} d(x,y)}{\sum_{xy \in [V(G)]^2} d(x,y)} \\
&= \frac{\sum_{xy \in E(G)} \sum_{i \in [k]} \lambda_i d_{T_i}(x,y)}{\sum_{xy \in [V(G)]^2} \sum_{i \in [k]} \lambda_i d_{T_i}(x,y)} \\
&= \frac{\sum_{i \in [k]} \sum_{xy \in E(G)} \lambda_i d_{T_i}(x,y)}{\sum_{i \in [k]} \sum_{xy \in [V(G)]^2} \lambda_i d_{T_i}(x,y)} \\
&\geq \min_{j \in [k]} \frac{\sum_{xy \in E(G)} \lambda_j d_{T_j}(x,y)}{\sum_{xy \in [V(G)]^2} \lambda_j d_{T_j}(x,y)} \\
&= \min_{j \in [k]} \frac{\sum_{xy \in E(G)} d_{T_j}(x,y)}{\sum_{xy \in [V(G)]^2} d_{T_j}(x,y)} \\
&= \min_{j \in [k]} \phi(d_{T_j})
\end{aligned}$$

□

An important consequence of Proposition 3.2 and Proposition 3.3 is the following corollary, which implies that optimizing over **the sparsest cut** is equivalent to both optimizing over all **elementary cut metrics** and **cut metrics** in general.

Corollary 3.1

Given a graph G, it holds that

$$\min_{T \subseteq V(G)} \psi(T) = \min_{T \subseteq V(G)} \phi(d_T) = \min_{d \text{ cut metric}} \phi(d)$$

We observe that the last equality derives from the fact that elementary cut metrics are *trivial* cut metrics.

3.1.2 The ℓ_1 metric

Up until this point, we have explored cut metrics in great detail. In the following section, we are going to shift our focus towards other types of metrics, in patricular the ℓ_μ type of metrics, which are defined as shown below.

Definition 3.8: ℓ_μ metrics

Given $\mu \in \mathbb{R}_{\geq 1}$, and a set $S \subseteq \mathbb{R}^d$ for some d, the ℓ_μ **metric** is a function defined as follows

$$\ell_\mu(x,y) = \sqrt[\mu]{\sum_{i=1}^{d} |x_i - y_i|^\mu}$$

We will denote $\ell_\mu(x,y)$ also as $|x-y|_{\ell_\mu}$.

In particular, we are interested in the ℓ_1 metric. In fact, consider the following definition.

Definition 3.9: Isometrical embedding

Consider two metrics $d_1 : A \times A \to B$ and $d_2 : X \times X \to Y$; we say that d_1 is **isometricaly embedded** into d_2 if there is a function $f : A \to X$ such that

$$d_1(x, y) = d_2(f(x), f(y))$$

Lemma 3.1

Any cut metric defined over k cuts is isometrically embedded into ℓ_1 over $\mathbb{R}^k$.

Proof. Let d be a cut metric defined over $T_1, \dots, T_k$, i.e. $d(x, y) = \sum_{i \in [k]} \lambda_i d_{T_i}(x, y)$ for some $\lambda_1, \dots, \lambda_k > 0$. For each $u \in V(G)$, let $\overline{x_u} \in \mathbb{R}^k$ be a vector defined as follows:

$$\overline{x_u}(i) := \begin{cases} \lambda_i & u \in T_i \\ 0 & u \notin T_i \end{cases}$$

where $\overline{x_u}(i)$ is the i-th component of $\overline{x_u}$.

Now, consider the function

$$f : V(G) \to \mathbb{R}^k : u \mapsto \overline{x_u}$$

Now, fix $i \in [k]$, and two vertices $u, v \in V(G)$; we observe that

- $|\{u, v\} \cap T_i| = 1 \implies (u \in T_i \wedge v \notin T_i) \vee (u \notin T_i \wedge v \in T_i)$; hence, we have that

  $$u \in T_i \wedge v \notin T_i \implies \overline{x_u}(i) = \lambda_i \wedge \overline{x_v}(i) = 0 \implies |\overline{x_u}(i) - \overline{x_v}(i)| = 1$$

  and also that

  $$u \notin T_i \wedge v \in T_i \implies \overline{x_u}(i) = 0 \wedge \overline{x_v}(i) = \lambda_i \implies |\overline{x_u}(i) - \overline{x_v}(i)| = 1$$

- $|\{u, v\} \cap T_i| = 0 \implies u, v \in T_i \vee u, v \notin T_i$; hence, we have that

  $$u, v \in T_i \implies \overline{x_u}(i) = \overline{x_v}(i) = \lambda_i \implies |\overline{x_u}(i) - \overline{x_v}(i)| = 0$$

  and also that

  $$u, v \notin T_i \implies \overline{x_u}(i) = \overline{x_v}(i) = 0 \implies |\overline{x_u}(i) - \overline{x_v}(i)| = 0$$

This means that for any $i \in [k]$ and $u, v \in V(G)$, it holds that

$$\mathbb{1}\left[|\{u, v\} \cap T_i| = 1\right] = |\overline{x_u}(i) - \overline{x_v}(i)|$$

Therefore, we obtain the following

$$\begin{aligned} d(u,v) &= \sum_{i\in[k]} \lambda_i d_{T_i}(u,v) \\ &= \sum_{i\in[k]} \lambda_i \cdot \mathbb{1}\left[|\{u,v\} \cap T_i| = 1\right] \\ &= \sum_{i\in[k]} |\overline{x_u}(i) - \overline{x_v}(i)| \\ &= |\overline{x_u} - \overline{x_v}|_{\ell_1} \end{aligned}$$

meaning that d is indeed isometrically embedded into ℓ_1 over $\mathbb{R}^k$. □

Lemma 3.2

For any d, and any set $X \underset{fin}{\subseteq} \mathbb{R}^d$, the ℓ_1 metric over X is isometrically embedded into a cut metric defined over $d \cdot (|x| - 1)$ cuts.

Proof. We prove the statement for $d = 1$, and then extend the proof to all other values of d.

Let $X \underset{fin}{\subseteq} \mathbb{R}^d$, i.e. $X = \{x_1, \ldots, x_n\}$ for some $n \in \mathbb{N}$; moreover, let $\pi : [n] \to [n]$ be a permutation of the indices ordering the n elements of X such that they are ordered in *ascending order*

$$x_{\pi(1)} \leq \ldots \leq x_{\pi(n)}$$

For each $j \in [n-1]$, let $S_j := \{\pi(1), \ldots, \pi(j)\}$; by construction, we have that

$$\forall j \in [n-1] \quad S_1 \subseteq \ldots \subseteq S_j$$

Moreover, for each $j \in [n-1]$ set $\lambda_j := x_{\pi(j+1)} - x_{\pi(j)}$; since the indices order the elements of X in ascending order, we know that $\lambda_j \geq 0$ for all $j \in [n-1]$.

Define d to be the cut metric over the cuts $S_1, \ldots, S_{n-1}$ with coefficients $\lambda_1, \ldots, \lambda_{n-1}$; hence, for each i and j such that $i < j$ we get that

$$\begin{aligned} d(x_{\pi(i)}, x_{\pi(j)}) &= \sum_{t\in[j]} \lambda_t d_{S_t}(x_{\pi(i)}, x_{\pi(j)}) \\ &= \sum_{t\in[j]} \lambda_t \cdot \mathbb{1}\left[\left|\{x_{\pi(i)}, x_{\pi(j)}\} \cap S_t\right| = 1\right] \\ &= \sum_{i\leq t<j} \lambda_t \\ &= (x_{\pi(j)} - x_{\pi(j-1)}) + (x_{\pi(j-1)} - x_{\pi(j-2)}) + \ldots + (x_{\pi(i+1)} - x_{\pi(i)}) \\ &= x_{\pi(j)} - x_{\pi(i)} \\ &= \left|x_{\pi(j)} - x_{\pi(i)}\right|_{\ell_1} \end{aligned}$$

Lastly, since metrics are symmetric, we conclude that

$$\forall i, j \quad d(x_{\pi(i)}, x_{\pi(j)}) = \ell_1(x_{\pi(1)}, x_{\pi(j)})$$

which means that for $d = 1$ finite ℓ_1 metrics can be embedded into a cut metric with $|X| - 1$ cuts. Now, for any dimension $d > 0$, finite ℓ_1 metrics can be embedded into a cut metric that is the sum of each 1-dimensional embedding, meaning that ℓ_1 can be embedded into a cut metric with $d \cdot (|X| - 1)$ cuts. □

These two lemmas further improve our optimization equalities: optimizing over a cut metric is equal to optimizing over an ℓ_1 **metric**.

Corollary 3.2

Given a graph G, it holds that

$$\min_{T \subseteq V(G)} \psi(T) = \min_{T \subseteq V(G)} \phi(d_T) = \min_{d \text{ cut metric}} \phi(d) = \min_{d \ \ell_1 \text{ metric}} \phi(d)$$

3.2 Metric relaxations and distortions

3.2.1 Metric relaxations

Consider the following program:

$$\max \frac{\sum_{ij \in E(G)} d_{ij}}{\sum_{ij \in [V(G)]^2} d_{ij}}$$

$$\sum_{S \subseteq V(G)} \lambda_S d_S(i,j) = d_i j \quad \forall ij \in [V(G)]^2$$
$$\lambda \in \mathbb{R}^{2^n}$$
$$d \in \mathbb{R}^{|\binom{n}{2}|}$$

Figure 3.1: Non-linear program for SC.

In this *non*-linear program, the variables d_{ij} and λ_S — as their names suggest — represent the cut values for the optimal cut metric d^* described by the optimal solution, which corresponds to $\phi(d^*)$. Note that the values $d_S(i,j)$ are elementary cut metrics, hence they act as nothing more than a constant coefficient, while we optimize over the values λ_S defining the *actual* coefficients of the cut.

Additionally, the main constraint of the program optimizes over cut metrics, which we proved to be "isometrically equivalent" to ℓ_1 metrics in the previous lemmas. Therefore, this program can be viewed as optimizing both over cut metrics and ℓ_1 metrics. However, the objective function is clearly *non*-linear, hence the program is *non*-linear as well. To

fix this, we can normalize the denominator of the objective function since it is shared among all feasible solutions, obtaining the following LP.

$$\max \sum_{ij \in E(G)} d_{ij}$$

$$\sum_{S \subseteq V(G)} \lambda_S d_S(i,j) = d_i j \quad \forall ij \in [V(G)]^2$$
$$\sum_{ij \in [V(G)]^2} d_i j = 1$$
$$\lambda \in \mathbb{R}^{2^n}$$
$$d \in \mathbb{R}^{|\binom{n}{2}|}$$

Figure 3.2: LP for SC.

By construction, the solution to this LP corresponds to the optimal cut metric d^*, and the optimal solution to SC is given by the minimal elementary cut describing d^*; therefore, this LP could be used to get an exact solution to SC. However, we cannot employ this LP because it requires an *exponential* number of variables. To solve this issue, consdier the following definition.

Definition 3.10: Metric distortion

Given two metrics d_1 and d_2 over the same vector space V, we say that d_2 has a **distortion** from d_1 of at most $\alpha\beta$ if

$$\forall x, y \in V \quad \frac{d_1(x,y)}{\alpha} \leq d_2(x,y) \leq \beta d_1(x,y)$$

Metric distortion describes the concept of *similarity* between measures. In 1985 Bourgain [Bou85] proved that any finite metric can be isometrically embedded into an ℓ_1 metric, up to some distortion factor.

Theorem 3.1: Bourgain's theorem (computational version)

Any metric d on n points can be isometrically embedded in time $n^O(1)$ into an ℓ_1 metric on $\mathbb{R}^d$, where $d = O(\log^3 n)$ and distortion factor $O(\log n)$.

Thanks to this result, Leighton and Rao [LR88] were able to provide an algorithm that yields an $O(\log n)$-approximation to SC by using the following **metric LP relaxation**: instead of optimizing over ℓ_1 metrics, they constructed a relaxation that optimizes over *all* metrics — introducing the approximation factor of $O(\log n)$, which is precisely the distortion factor.

$$\max \sum_{ij \in E(G)} d_{ij}$$

$$\begin{aligned} x_{ij} + xjk &\geq x_{ik} \quad \forall i, j, k \in V(G) \\ \sum_{ij \in [V(G)]^2} d_i j &= 1 \\ \lambda &\in \mathbb{R}^{2^n} \\ d &\in \mathbb{R}^{\left|\binom{n}{2}\right|} \end{aligned}$$

Figure 3.3: Metric LP relaxation for SC.

Algorithm 3.1: Leighton-Rao algorithm

Given a graph G, the algorithm returns a cut of G.

1: **function** LEIGHTONRAO(G)
2: $d := \mathrm{LP}_{\mathrm{metric}}(G)$
3: $d' :=$ TO-ℓ_1-METRIC(d) ▷ apply Theorem 3.1
4: $d'' :=$ TO-CUT-METRIC(d') ▷ apply Lemma 3.2
5: Let $T_1, \ldots, T_k$ be the k cuts of d''
6: **return** $S \in \arg\min_{i \in [k]} \phi(T_i)$
7: **end function**

Thanks to Corollary 3.2 and Theorem 3.1, we obtain the following result.

Theorem 3.2: LEIGHTONRAO's approximation ratio

Given a graph G, and an optimal solution S^* to SC on G, let $S :=$ LEIGHTONRAO(G). Then, it holds that

$$\psi(S) \leq O(\log n) \cdot \psi(S^*)$$

In recent years Arora, Rao, and Vazirani [ARV09] were able to improve the approximation ratio up to $O(\sqrt{\log n})$ through the following SDP.

$$\max \sum_{ij \in E(G)} \langle x_i, x_i \rangle - 2 \langle x_i, x_j \rangle + \langle x_j, x_j \rangle$$

$$\begin{aligned} \langle x_i, x_j \rangle - \langle x_j, x_j \rangle + \langle x_j, x_k \rangle \leq \langle x_i, x_k \rangle \qquad & \forall i, j, k \in V(G) \\ \sum_{ij \in [V(G)]^2} \langle x_i, x_i \rangle - 2 \langle x_i, x_j \rangle + \langle x_j, x_j \rangle = 1 & \\ \lambda \in \mathbb{R}^{2^n} & \\ d \in \mathbb{R}^{\left|\binom{n}{2}\right|} & \end{aligned}$$

Figure 3.4: ℓ_2^2 metric SDP relaxation for SC.

The idea is similar to that of the Leighton-Rao algorithm: instead of relaxing to every metric, they relax the problem to **squared ℓ_2 metrics**. In particular, they observed that

$$\begin{aligned} \ell_2^2(x, y) &= \sum_{i \in [n]} |x_i - y_i|^2 \\ &= \sum_{i \in [n]} \left(x_i^2 - 2 x_i y_i + y_i^2\right) \\ &= \sum_{i \in [n]} x_i^2 - 2 \sum_{i \in [n]} x_i y_i + \sum_{i \in [n]} y_i^2 \\ &= \langle x, x \rangle - 2 \langle x, y \rangle + \langle y, y \rangle \end{aligned}$$

Note that ℓ_2^2 does not always respect the *triangle inequality*. For instance, in $\mathbb{R}$ we have that $\ell_2^2(-1, 0) = \ell_2(0, 1) = 1$, but $\ell_2^2(-1, 1) = 4 > 1$. Hence, the triangle inequality is forced by second constraint of the SDP itself, in fact:

$$\ell_2^2(x, z) + \ell_2^2(z, y) \leq \ell_2^2(x, y) \implies$$

$$(\langle x_i, x_i \rangle - 2 \langle x_i, x_j \rangle + \langle x_j, x_j \rangle) + (\langle x_j, x_j \rangle - 2 \langle x_j, x_k \rangle + \langle x_k, x_k \rangle) \geq \langle x_i, x_i \rangle - 2 \langle x_i, x_k \rangle + \langle x_k, x_k \rangle$$

$$\implies -2 \langle x_i, x_j \rangle + 2 \langle x_j, x_j \rangle - 2 \langle x_j, x_k \rangle \geq -2 \langle x_i, x_k \rangle$$

$$\langle x_i, x_j \rangle - \langle x_j, x_j \rangle + \langle x_j, x_k \rangle \leq \langle x_i, x_k \rangle$$

Proposition 3.4

Any ℓ_2^2 metric on n points can be isometrically embedded in time $n^{O}(1)$ into an ℓ_1 metric on $\mathbb{R}^d$ with distortion factor $O(\sqrt{\log n})$.

Proof. Omitted. □

This $O(\sqrt{\log n})$ is the current best known approximation for SC. Moreover, SC cannot be expressed in temrs of CSP — recall that the above SDP is a *relaxation* for the minimal ℓ_2^2 problem. Furthermore, even if this approximation ratio is believed to be the best possible one, the UGC only implies that there are no constant factor approximations for SC, meaning that the non-constant ratio could be proved to be lower.

3.2.2 Shortest path metric

Metric distortion has showed to be a useful tool for various approximation algorithms. However, in some instances there is a *lower bound* on the distortion required in order to embed a metric into another. For instance, consider the following metric.

Definition 3.11: Shortest path metric

Given a graph G, the **shortest path metric** is a function $d : V(G) \times V(G) \to \mathbb{R}$ defined as follows

$$d_G(x, y) = |\{e \in E(P) \mid P \text{ shortest } u \to v \text{ path}\}|$$

It can be easily proven that this is indeed a metric — we will omit the proof.

We are going to show that the shortest path metric cannot be embedded into ℓ_2 over $\mathbb{R}^d$ withouth distortion. In particular, we will show the case for the cycle graph C_4. Consider the embedding $f : V(C_4) \to \mathbb{R}^2$ such that

$$f(1) = \left(-\frac{1}{2}; \frac{1}{2}\right) \quad f(2) = \left(\frac{1}{2}; \frac{1}{2}\right) \quad f(3) = \left(\frac{1}{2}; -\frac{1}{2}\right) \quad f(4) = \left(-\frac{1}{2}; -\frac{1}{2}\right)$$

It is easy to see that this embedding is isometrical from d_{C_4} to ℓ_1, since all distances are preserved. However, for the ℓ_2 metrc the diagonals are *not* isometric, in fact

$$2 = d_{C_4}(1, 3) \neq \ell_2(f(1), f(3)) = \ell_2\left(\left(-\frac{1}{2}; \frac{1}{2}\right), \left(\frac{1}{2}; -\frac{1}{2}\right)\right) = \sqrt{2}$$

In particular, this embedding induces a distortion factor of at most $\sqrt{2}$. Furthermore, thanks to the following lemma we prove that this distortion factor is actually the *best* that we can achieve.

Lemma 3.3: Shortest Diagonal lemma

For any $y_1, y_2, y_3, y_4 \in \mathbb{R}^4$ it holds that

$$\ell_2^2(y_1, y_3) + \ell_2^2(y_2, y_4) \leq \ell_2^2(y_1, y_2) + \ell_2^2(y_2, y_3) + \ell_2^2(y_3, y_4) + \ell_2^2(y_4, y_1)$$

Proof. We will employ a technique called *sum of squares proof*: the idea is to show that one equation made of sums of squares is equivalent to the square of an other equation, implying that the first one must be always non-negative. We observe that finding the second equation to use as "comparison" is *very* hard.

We will prove the lemma coordinate-by-coordinate: fix $t \in [d]$, and for each $i \in [4]$ let $z_i := y_i(t)$ be the t-th coordinate of y_i.

Claim: For all $t \in [d]$ it holds that

$$(z_1 - z_4)^2 + (z_2 - z_3)^2 + (z_3 - z_4)^2 + (z_4 - z_1)^2 - (z_1 - z_3)^2 - (z_2 - z_4)^2 = (z_1 - z_2 + z_3 - z_4)^2$$

Proof of the Claim. It can be proven by expanding both the left and right hand side of the quation. □

Because of the claim, we get that

$$(z_1-z_4)^2+(z_2-z_3)^2+(z_3-z_4)^2+(z_4-z_1)^2-(z_1-z_3)^2-(z_2-z_4)^2 = (z_1-z_2+z_3-z_4)^2 \geq 0$$

since the LHS is a square. Now, recalling that $\ell_2^2(y_i, y_j) := \sum_{t\in[d]} (y_t(i) - y_t(j))^2$, we get that

$$\ell_2^2(y_1, y_2) + \ell_2^2(y_2, y_3) + \ell_2^2(y_3, y_4) + \ell_2^2(y_4, y_1) - \ell_2^2(y_1, y_3) - \ell_2^2(y_2, y_4) \geq 0$$

□

Proposition 3.5

Any embedding of the shortest path metric d_{C_4} into ℓ_2 over $\mathbb{R}^d$, for any $d \in \mathbb{N}_{\geq 1}$, has a distortion factor of at least $\sqrt{2}$.

Proof. Fix $d \in \mathbb{N}_{\geq 1}$, and consider an embedding $f : V(C_4) \to \mathbb{R}^d$. Without loss of generality, we may assume that $f(i) = x_i$ for each $i \in [4]$. Let M be the maximum ℓ_2 edge length over all the points embedded into $\mathbb{R}^d$, in other words

$$M_{\ell_2} := \max(\ell_2(x_1, x_2), \ell_2(x_2, x_3), \ell_2(x_3, x_4), \ell_2(x_4, x_1))$$

and let m be the minimum ℓ_2 diagonal length over all the points embedded into $\mathbb{R}^d$, in other words

$$m_{\ell_2} := \min(\ell_2(x_1, x_3), \ell_2(x_2, x_4))$$

Thanks to the Shortest Diagonal lemma, we have that

$$\begin{aligned} 2m_{\ell_2}^2 &\leq \ell_2^2(y_1, y_3) + \ell_2^2(y_2, y_4) \\ &\leq \ell_2^2(y_1, y_2) + \ell_2^2(y_2, y_3) + \ell_2^2(y_3, y_4) + \ell_2^2(y_4, y_1) \\ &\leq 4M_{\ell_2}^2 \end{aligned}$$

implying that $2m_{\ell_2}^2 \leq 4M_{\ell_2}^2 \iff m_{\ell_2} \leq \sqrt{2}M_{\ell_2}$. Now, note that the maximum edge length M_{C_4} w.r.t. d_{C_4} is 1, and the minimum diagonal length m_{C_4} w.r.t. d_{C_4} is 2, which means that $m_{C_4} = 2M_{C_4}$. Therefore, the embedding must have applied a distortion factor of at least $\sqrt{2}$. □

4

Submodular optimization

Submodular optimization plays a central role in discrete optimization, particularly in algorithmic settings where efficiency and approximation guarantees are critical. Submodular functions model a *diminishing returns* property that arises naturally in many computational problems, such as influence maximization, data summarization, and sensor placement. But before introducing submodular optimization in detail, we will discuss the following problem.

Definition 4.1: Max Cover problem

The **Max Cover** (MC) problem is defined as follows: given a *universe* (or *ground*) set $\mathcal{U} = [n]$, a collection of sets $C = \{S_1, \ldots, S_m\}$ such that $S_i \subseteq \mathcal{U}$, and an integer $k \geq 1$, determine the sub-collection $S \subseteq C$ such that $|S| = k$ that maximizes $\left|\bigcup_{S_j \in S} S_j\right|$.

In other words, we are asked to determine the sub-collection of the given C of k sets that covers as many elements of $\mathcal{U}$ as possible. For instance, given $\mathcal{U} = [7]$, $S_1 = \{1, 2\}$, $S_2 = \{2, 3\}$, $S_3 = \{3, 4, 5\}$, $S_4 = \{5, 6, 7\}$, an optimal solution would be $S = \{S_1, S_3\}$ because $|S_1 \cup S_3| = |\{1, 2, 3, 4, 5\}| = 5$.

The approximation of MC is *solved*, meaning that the best known approximation ratio of $1 - \frac{1}{e}$ has been proven to be tight under NP-hardness. Such approximation ratio can be achieved through the following greedy algorithm.

Algorithm 4.1: $\left(1 - \frac{1}{e}\right)$-approximation for MC

Given an instance of MC $(\mathcal{U}, C, k)$, the algorithm returns a $\left(1 - \frac{1}{e}\right)$-approximation for the instance.

1: **function** ApproxMaxCover($\mathcal{U}$, C, k)
2: $S := \varnothing$
3: $T_0 := \varnothing$
4: **for** $i \in [k]$ **do**
5: $S_i \in \arg\max_{S_j \in C - S} |S_j - T_{i-1}|$ ▷ S_i maximizes the number of "new" elements
6: $S = S \cup \{S_i\}$
7: $T_i = T_{i-1} \cup S_i$
8: **end for**
9: **return** S
10: **end function**

Theorem 4.1: ApproxMaxCover's approximation ratio

Given an instance (U, C, k) of MC, and an optimal solution S^* to MC on $(\mathcal{U}, C, k)$, let $S :=$ ApproxMaxCover$(\mathcal{U}, C, k)$. Then, it holds that

$$\left|\bigcup_{S_j \in S} S_j\right| \geq \left(1 - \frac{1}{e}\right) \left|\bigcup_{S_j \in S^*} S_j\right|$$

Proof. Consider an instance $(\mathcal{U}, C, k)$ of MC, and an optimal solution S^* to it; moreover, let $X^* := \bigcup_{S_j^* \in S^*} S_j^*$ be the set of elements of $\mathcal{U}$ covered by S^*.

Let $S :=$ ApproxMaxCover$(\mathcal{U}, C, k)$; hence, we have that $T_i = \bigcup_{j=1}^{i} S_j$. Additionally, let $t_i := |T_i|$ and $\mu_i := |X^*| - t_i$. Hence, we have that

- $t_0 = |T_0| = |\varnothing| = 0$
- $t_k = |T_k| = \left|\bigcup_{S_j \in S} S_j\right|$ is the value of S
- $\mu_0 = |X^*| - t_0 = |X^*| - 0 = |X^*|$ is the value of the optimal solution S^*

Therefore, to prove the approximation ratio of the statement it suffices to show that $t_k \geq \left(1 - \frac{1}{e}\right) \mu_0$.

Claim: $\forall i \in [k] \quad |S_i - T_{i-1}| \geq \frac{\mu_{i-1}}{k}$

Proof of the Claim. Since S^* is an optimal solution that covers $|X^*|$ elements of $\mathcal{U}$, there must be at least one set $S_j^* \in S^*$ that covers at least □

i'm missing something here

Claim: $\forall i \in [0, k] \quad \mu_i \leq \left(1 - \frac{1}{k}\right)^i |X^*|$

Proof of the Claim. We will prove the claim by induction on i. In particular, if $i = 0$, we have that

$$\mu_0 = |X^*| \left(1 - \frac{1}{k}\right)^0 |X^*| = 1 \cdot |X^*|$$

hence the base case trivially holds. Then, assuming the inductive hypothesis, we can prove the inductive step as follows

non capisco (TODO dentro allo split)

$$\begin{aligned}
\mu_{i+1} &= |X^*| - t_{i+1} \\
&= |X^*| - |T_{i+1}| \\
&= |X^*| - TODO \\
&= |X^*| - |T_i| - |S_{i+1} - T_i| \\
&\leq \mu_i - \frac{\mu_i}{k} \qquad \text{(for the previous claim)} \\
&= \left(1 - \frac{1}{k}\right) \mu_i \\
&\leq \left(1 - \frac{1}{k}\right) \left(1 - \frac{1}{k}\right)^i |X^*| \qquad \text{(by inductive hypothesis on } \mu_i\text{)} \\
&= \left(1 - \frac{1}{k}\right)^{i+1} |X^*|
\end{aligned}$$

□

Thanks to this claim, recalling that for any $k \geq 1$ it holds that $\left(1 - \frac{1}{k}\right)^k \leq e^{-1}$, we get

$$\mu_k \leq \left(1 - \frac{1}{k}\right)^k |X^*| \leq \frac{1}{e} |X^*|$$

Lastly, since $\mu_k = |X^*| - |T_k|$, we get that

$$|X^*| - |T_k| = \mu_k \leq \frac{1}{e} |X^*| \iff |T_k| \geq \left(1 - \frac{1}{e}\right) |X^*| \iff t_k \geq \left(1 - \frac{1}{e}\right) \mu_0$$

□

4.1 Submodular functions

We are now ready to introduce **submodular functions**, which play a central role in *submodular optimization* that we introduced at the beginnig of the chapter.

Definition 4.2: Modular functions

Given $n \in \mathbb{N}$, let $f : \mathcal{P}([n]) \to \mathbb{R}$ be a function; given two sets $S, T \subseteq [n]$, we say that f is

- a **submodular function** if $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$
- a **modular function** if $f(S) + f(T) = f(S \cup T) + f(S \cap T)$
- a **supermodular function** if $f(S) + f(T) \leq f(S \cup T) + f(S \cap T)$

For instance, for any $n \in \mathbb{N}$ the *cardinality* function, i.e. $f(S) = |S|$ is modular, because $\forall S, T \subseteq [n] \quad f(S \cup T) = f(S) + f(T) - f(S \cap T)$ by the inclusion-exclusion principle. We observe that submodular and supermodular functions are modular, therefore any property that holds for modular functions is true for the other two types as well.

The main property of modular functions is the property to "know the full description of the function" through at most $n + 1$ values. For example, if f is modular, then the value of $f(\{1, 2, 3\})$ can be computed through $f(\varnothing)$, $f(\{1\})$, $f(\{2\})$ and $f(\{3\})$ as follows

$$f(\{1,2,3\}) = f(\{1,2\}) + f(\{3\}) - f(\varnothing) = (f(\{1\}) + f(\{2\}) - f(\varnothing)) + f(\{3\}) - f(\varnothing)$$

Proposition 4.1

For any $n \in \mathbb{N}$, the function $f : \mathcal{P}([n]) \to \mathbb{R}$ is modular if and only if there exist $x, w_1, \ldots, w_n$ such that for all $S \subseteq [n]$ it holds that $f(S) = z + \sum_{i \in S} w_i$.

Proof. Let f be a modular function defined on $n \in \mathbb{N}$, and fix a set $S \subseteq [n]$; in particular, let $S = \{1, \ldots, k\}$. By modularity of f, we have that

$$\begin{aligned}
f(S) &= f(\{1, \ldots, k\}) \\
&= f(\{2, \ldots, k\}) + f(\{1\}) - f(\varnothing) \\
&= (f(\{3, \ldots, k\}) + f(\{2\}) - f(\varnothing)) + f(\{1\}) - f(\varnothing) \\
&= f(\{k\}) + \sum_{i \in S - \{k\}} (f(\{i\}) - f(\varnothing)) \\
&= f(\varnothing) + \sum_{i \in S} f(\{i\}) - f(\varnothing) \\
&= z + \sum_{i \in S} w_i
\end{aligned}$$

where $z. = f(\varnothing)$ and $w_i := f(\{i\}) - f(\varnothing)$ for all $i \in S$.

Vice versa, suppose that there are values $z', w'_1, \ldots, w'_n$ such that for all $S \subseteq [n]$ it holds

that $f(S) = z' + \sum_{i \in S} w'_i$. Fix two sets $S, T \subseteq [n]$; then, we have that

$$\begin{aligned} f(S) + f(T) &= \left(z' + \sum_{i \in S} w'_i \right) + \left(z' + \sum_{i \in T} w'_i \right) \\ &= \left(z' + \sum_{i \in S \cap T} w'_i + \sum_{i \in S - T} w'_i \right) + \left(z' + \sum_{i \in S \cap T} w'_i + \sum_{i \in T - S} w'_i \right) \\ &= \left(z' + \sum_{i \in S \cap T} w'_i \right) + \left(z' + \sum_{i \in S - T} w'_i + \sum_{i \in S \cap T} w'_i + \sum_{i \in T - S} w'_i \right) \\ &= f(S \cap T) + f(S \cup T) \end{aligned}$$

hence f is modular. □

The importance of submodularity in optimization stems from its inherent **diminishing returns** property. In economics, diminishing returns describe the phenomenon where the incremental output of a production process decreases as the quantity of a single input increases, while all other inputs are held constant.

Definition 4.3: Diminishing returns

Given $n \in \mathbb{N}$, let $f : \mathcal{P}([n]) \to \mathbb{R}$; given $S \subseteq [n]$ and $x \in [n] - S$, the **return** of x on S for f is defined as

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

We say that f has **diminishing returns** when it holds that

$$\forall A \subseteq B \subseteq [n], x \in [n] - B \quad \Delta_f(x \mid A) \geq \Delta_f(x \mid B)$$

Theorem 4.2

For any $n \in \mathbb{N}$, the function $f : \mathcal{P}([n]) \to \mathbb{R}$ is submodular if and only if it has diminishing returns.

Proof. Let $f : \mathcal{P}([n]) \to \mathbb{R}$ be a submodular function for some $n \in \mathbb{N}$, and fix $A \subseteq B \subseteq [n]$ and $x \in [n] - B$; by submodularity of f, we have that

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

Now, since $A \subseteq B$, and $x \notin B$, we have that

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

Lastly, rearranging the terms, we get that

$$\Delta_f(x \mid A) := f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B) =: \Delta_f(x \mid B)$$

TODO □

do the other direction

4.1.1 TODO

Why did we present the Max Cover problem at the beginning of the chapter? MC is a **submodular optimization** problem, i.e. the problem can be described in terms of a submodular function.

Definition 4.4: Covering function

Given $t \in \mathbb{N}$, let $\mathcal{U} = [t]$ be the universe set, and let $C = \{A_1, \dots, A_n\}$ be a collection of subsets $A_i \subseteq \mathcal{U}$; the **covering function** $f : \mathcal{P}([n]) \to \mathbb{R}$ is defined as follows:

$$\forall S \subseteq [n] \quad f(S) := \left| \bigcup_{j \in S} A_j \right|$$

For example, given $t = 4$ and $A_1 = \{1, 2, 4\}$, $A_2 = \{2, 3\}$, $A_3 = \{1, 4\}$ — hence $n = 3$ and $C = \{A_1, A_2, A_3\}$ — we have that

$$f(\{1, 3\}) = \left| \bigcup_{j \in \{1,3\}} A_j \right| = |A_1 \cup A_3| = |\{1, 2, 4\}| = 3$$

and so on. In other words, the set S is used to *index* the sets $A_j \in C$.

Proposition 4.2

The covering function is non-negative, monotone increasing and submodular.

Proof. Given a set $X \subseteq [n]$, note that

$$f(X) = \sum_{i \in [t]} \mathbb{1}\left[\exists j \in X \quad i \in A_j\right] = \sum_{i \in [t]} f_i(X)$$

where for each $i \in [t]$ the subfunction $f_i : \mathcal{P}([n]) \to \mathbb{R}$ is defined precisely as

$$f_i(X) := \begin{cases} 1 & \exists j \in X \quad i \in A_j \\ 0 & \text{otherwise} \end{cases}$$

Claim: For all $i \in [t]$, the subfunction f_i is non-negative, monotone increasing and submodular.

Proof of the Claim. TODO □ da finire

TODO □ da finire

Since MC is NP-Hard, this proposition implies that the problem of maximizing non-negative, monotone increasing, and submodular functions over k elements is NP-Hard as well. In fact, the $\left(1 - \frac{1}{e}\right)$-approximation for MC that we previously discussed is a specific

instance of a more general algorithm, first presented by Nemhauser, Wolsey, and Fisher [NWF78], that aims at maximizing a Non-Negative and Monotone Increasing (NN-MI) submodular function.

TODO da finire

Bibliography

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, et al. "Proof verification and the hardness of approximation problems". In: *Journal of the ACM* 45.3 (May 1998), 501–555. ISSN: 1557-735X. DOI: 10.1145/278298.278306. URL: http://dx.doi.org/10.1145/278298.278306.

[ARV09] Sanjeev Arora, Satish Rao, and Umesh Vazirani. "Expander flows, geometric embeddings and graph partitioning". In: *Journal of the ACM* 56.2 (Apr. 2009), 1–37. ISSN: 1557-735X. DOI: 10.1145/1502793.1502794. URL: http://dx.doi.org/10.1145/1502793.1502794.

[Bou85] J. Bourgain. "On lipschitz embedding of finite metric spaces in Hilbert space". In: *Israel Journal of Mathematics* 52.1–2 (Mar. 1985), 46–52. ISSN: 1565-8511. DOI: 10.1007/bf02776078. URL: http://dx.doi.org/10.1007/BF02776078.

[Cha00] Moses Charikar. "Greedy Approximation Algorithms for Finding Dense Components in a Graph". In: *Approximation Algorithms for Combinatorial Optimization*. Springer Berlin Heidelberg, 2000, 84–95. ISBN: 9783540444367. DOI: 10.1007/3-540-44436-x_10. URL: http://dx.doi.org/10.1007/3-540-44436-X_10.

[ER22] P. Erdős and A. Rényi. "On random graphs. I." In: *Publicationes Mathematicae Debrecen* 6.3–4 (July 2022), 290–297. ISSN: 0033-3883. DOI: 10.5486/pmd.1959.6.3-4.12. URL: http://dx.doi.org/10.5486/PMD.1959.6.3-4.12.

[Kar72] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations*. Springer US, 1972, 85–103. ISBN: 9781468420012. DOI: 10.1007/978-1-4684-2001-2_9. URL: http://dx.doi.org/10.1007/978-1-4684-2001-2_9.

[Kho02] Subhash Khot. "On the power of unique 2-prover 1-round games". In: *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. STOC02. ACM, May 2002. DOI: 10.1145/509907.510017. URL: http://dx.doi.org/10.1145/509907.510017.

[Lov75] L. Lovász. "On the ratio of optimal integral and fractional covers". In: *Discrete Mathematics* 13.4 (1975), 383–390. ISSN: 0012-365X. DOI: 10.1016/0012-365x(75)90058-8. URL: http://dx.doi.org/10.1016/0012-365X(75)90058-8.

[LR88] Tom Leighton and Satish Rao. "An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approxima-

tion algorithms". In: *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society. 1988, pp. 422–431.

[NWF78] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. "An analysis of approximations for maximizing submodular set functions—I". In: *Mathematical programming* 14 (1978), pp. 265–294.

[Rag08] Prasad Raghavendra. "Optimal algorithms and inapproximability results for every CSP?" In: *Proceedings of the fortieth annual ACM symposium on Theory of computing*. STOC '08. ACM, May 2008, 245–254. DOI: 10.1145/1374376.1374414. URL: http://dx.doi.org/10.1145/1374376.1374414.