Brief Intro to Hardness of Approximation

03 May 2021 10:01

Q: What is the Class NP?

Defn: defined for <u>decision problems</u>

e.x is given a graph, is it 3-COLOURABLE?

Here $\mathcal{L} = \{ G : G \text{ is 3-colourable} \}$

[NP is a class of languages which admits a poly-time non-deterministic turing machine.]

In contrast P is the class of languages identifiable with deterministic poly-time turing machine

Equivalent viewpoint of the Class NP in a "prover - verifier language"

NP is the class of languages L st

$\exists$ deterministic polynomial time Algo (VERIFIER)

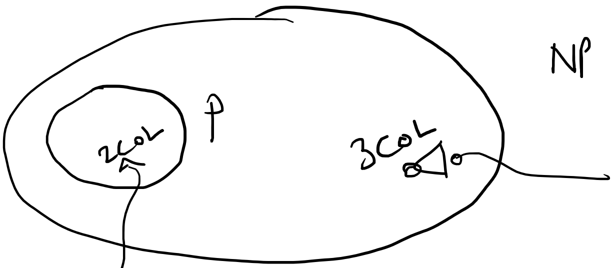
s.t

$\forall x \in L$, $\exists$ proof $\pi(x)$ of poly($|x|$) bits such that

$$V(x, \pi(x)) = 1$$

$\forall\, x \notin L$, no proof should make V accept

(i.e. $\forall\, \pi(x)$, $V(x, \pi(x)) = 0$)



Given graph, is it 3-colourable?

2COL: Given a graph, is it 2-colourable?

$L = \{$ 2-colourable graphs $\}$

$\boxed{\text{Is } P = NP}$ Most longstanding problems in complexity theory.

{ Widely believed that $P \neq NP$. }

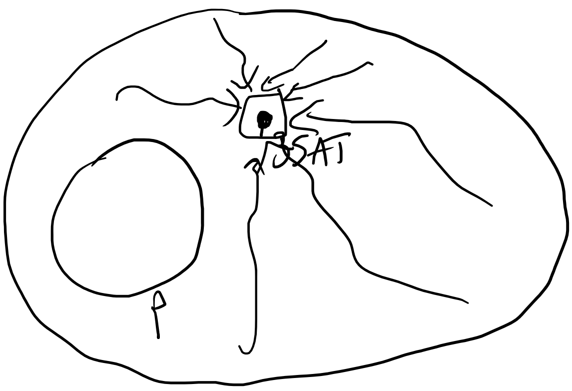
Cook-Levin Theorem [1971]

3SAT is <u>NP-Complete</u>.

Given a CNF where each clause is an OR of 3 literals (x_i or $\bar{x}_i$)

check if it is satisfiable or not.

When is a problem NP-complete?



A problem Q is NP Complete if

a) it is in NP

b) every other problem in NP can be reduced to Q in poly-time.

Meaning,

if we discover a poly-time algo for 3SAT, then we can obtain poly-time algos for any problem in NP by using Cook-Levin Thm.

This notion of NP completeness gave a way of showing how "hard" some decision problems are.

3COL is NP-Complete

How would we show something like this?

a) 3COL ∈ NP [Proof is the coloring]

b) Every other problem in NP can reduce to 3COL in poly-time?

(b) seems tricky, but its not due to Cook-Levin simply reduce 3SAT ~~to~~ 3COL in polytime

(ie)

given $\mathcal{I}$ of 3SAT, output a graph $G(\mathcal{I})$ in poly-time such that

$$\mathcal{I} \text{ is satisfiable } \rightleftarrows G \text{ is 3-colorable.}$$

Great! We have a way of characterizing "hardness" of decision problems by showing NP-completeness.

For longest time, it was not clear how to extend these ideas / notions to optimization / approximation-type problems. ☹

Ex:

We know 3-SAT is NP-complete.

Let's look at Max-3SAT:

given an instance, find the

assignment which satisfies
max # clauses ?

- Can't admit poly-time algos (if $P \neq NP$)

Can it admit a PTAS?

(ie) $\forall$ constant $\varepsilon > 0$, can we get a $(1-\varepsilon)$-approximation algorithm which runs in poly-time ?

Similarly can ask for
Graph colouring.

Given graph G, colour it with fewest colours.

What's the best approximation algo for this ? Is there a $(1+\varepsilon)$-approximation ? Is there a 2-approximation ?

Another side of the story.

How to analyze the class NP from the perspective of the verifier ?

How all can we restrict the verifier and still admit the NP-class ?

(long story for the motivation)

One attempt [ALMSS '98]

Probabilistically Checkable Proof $\boxed{PCP}$

PCP[$r(n)$, $s(n)$] is the class of all languages L s.t. $\exists$ poly-time verifier V

$x \in L \Rightarrow \exists$ proof $\pi(x)$ of size poly(|x|) such that

V probes only $s(|x|)$ bits of the proof and accepts with prob. 1.

$x \notin L \Rightarrow \forall$ proofs, , V rejects after checking $s(|x|)$ bits of the proof with prob $\geq \frac{1}{2}$.

The verifier V is restricted to probe only $s(n)$ bits of the proof.

Hence we give it some power to toss $r(n)$ random coins.

So V can look at x and the $r(n)$ coins to determine which $s(n)$ bits to probe.

$PCP[r(n), s(n)]$

poly time randomized verifier which tosses $r(n)$ coins

$x \in L \Rightarrow$ $\exists$ proof $\pi(x)$, V probes $s(n)$ bits of proof π and accepts w.p. 1

$x \notin L \Rightarrow \forall$ proof π, V rejects wp $\geq 1/2$.

REMARKABLE THM [ALMSS '98]

$$NP = PCP[O(\log n), O(1)] \quad !!!$$

$\Rightarrow$ can check if $x \in L$ by looking at only constant bits of the proof!

Has beautiful connections to hardness of approximations

Opened the door to studying if 3SAT, COL have PTAS, etc.

<u>Goal</u>: Hardness of Approximation of Max3SAT

Prior to PCPs, was not known if
Max 3SAT admits a $(1-\varepsilon)$-approximation for any constant $\varepsilon > 0$.
(i.e.) PTAS ?

Emergence of PCPs $\Rightarrow$ PTAS NOT POSSIBLE
There is some fixed $\varepsilon > 0$ for which it is NP-hard to approximate Max3SAT to a factor better than $(1-\varepsilon)$.

Subsequent improvements to PCP Machinery
$\forall\ \varepsilon > 0$, it is NP-hard to get a $\left(\frac{7}{8}+\varepsilon\right)$-approx to Max3SAT.

And we know trivial $\frac{7}{8}$-approximation.

Q: What does it mean?
A: If you design a poly-time $\left(\frac{7}{8}+\varepsilon\right)$-approx for $\varepsilon > 0$ then we can design a poly-time algo for 3SAT.

PCP view of the NP class allows us to go from such approx - hardness to classical NP - completeness.

Q: How are we going to show APX - hardness of Max 3SAT ?

Idea: create an intermediate decision problem and show it is NP - complete.

<u>Problem</u>

Gap $3SAT_{c,s}$:

Given an instance I of 3SAT, output

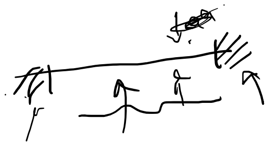
① Yes if there is an assignment satisfying $\geq c$ fraction of clauses

② NO if <u>NO</u> assignment can satisfy $\geq s$ fraction of clauses

③ YES/NO if intermediate values.

Ex : fr $c = 1$, $s = 0.9$

Gap $3SAT_{1, 0.9}$:



Intuitively easier than regular 3SAT, where we need to give correct answer in all regimes of satisfiability.

Using PCP Thm, we can show that
★ $\text{Gap 3SAT}_{1,1-\varepsilon}$ is NP-complete for some constant $\varepsilon > 0$.

Using ★, easy to show $(1-\varepsilon)$-hardness of Max 3SAT.

Sps $\exists$ A which is a $(1-\varepsilon)$-approx for Max 3SAT

Given I of $\text{Gap 3SAT}_{1,1-\varepsilon}$, run A on I.
if it finds an assignment satisfying $\geq (1-\varepsilon)$ fraction of clauses,
Output YES
- Else output NO

Todo
Show that $\text{Gap 3SAT}_{1,1-\varepsilon}$ is NP-complete

Proof
From PCP Theorem + reduction

We know that 3COL is NP-complete.

We'll reduce it to $Gap3SAT_{1,1-\varepsilon}$.

Traditional reductions don't give such "gaps"

PCP view comes to our rescue.

$$\boxed{3COL} \xrightarrow[\text{reduction}]{\text{poly-time}} \boxed{Gap3SAT_{1,1-\varepsilon}}$$

We'll use the PCP view of 3-coloring.

$\exists$ verifier V which tosses $O(\log n)$-random coins, and looks at $O(1)$ bits of the proof to decide if $G \in 3COL$.

- if $G \in 3COL$, $\exists$ ~~proof~~ st $\Pr_{coins}[V \text{ accepts}] = 1$
- if $G \notin 3COL$, $\forall$ proofs $\Pr_{coins}[V \text{ accepts}] \le 1/2$.

Our reduction will create an instance of $Gap3SAT_{1,1-\varepsilon}$ as follows :-

Imagine that proof provided is of the form

$x_1, x_2, x_3, \ldots X_N$

where $N = \underline{poly(n) \cdot \text{proof size}}$

Verifier tosses $O(\log n)$ - coins

$C_1 \quad C_2 \cdots C_r$ Probe   truth-table

$\rightarrow 0, 0, \dots, 0 \rightarrow \underbrace{x_1^1, x_2^1, \dots, x_5^1}$ $\rightarrow 0$, $\rightarrow 1$

$\rightarrow \vdots$

$\rightarrow \vdots$

$\rightarrow 1, 1, 1, 1, 1 \rightarrow x_1^k, x_2^k, \dots x_5^k$ $\rightarrow 0$, $\rightarrow 1$

We'll form a 3CNF from these truth tables.

We'll convert each truth table to a collection of clauses. (small #)

For example

x_1	x_{10}	x_{15}	x_{20}	output of V
				0
0	1	1	1	0
1	0	1	0	0
1	1	1	1	

otherwise accept.

Verifier is checking

$(\bar{x}_1 \wedge x_{10} \wedge x_{15} \wedge x_{20}) \vee$

reject $\Leftarrow$ $(x_1 \wedge \bar{x}_{10} \wedge x_{15} \wedge \bar{x}_{20}) \vee$
$(x_1 \wedge x_{10} \wedge x_{15} \wedge x_{20})$

accept $\Leftarrow$ $(x_1 \vee \bar{x}_{10} \vee \bar{x}_{15} \vee \bar{x}_{20}) \wedge$
$(\bar{x}_1 \vee x_{10} \vee \bar{x}_{15} \vee x_{20}) \wedge$
$(\bar{x}_1 \vee \bar{x}_{10} \vee \bar{x}_{15} \vee \bar{x}_{20})$

[Can use some small # auxilliary variables to make it a 3CNF.]

In total, we'll generate poly(n) poly(s) clauses
but s is constant.

This is the desired Gap 3SAT instance $\tilde{I}$

Final part of proof:

$G \in$ 3COL, then all clauses of $\tilde{I}$ are satisfied

$G \notin$ 3COL, then $\leq (1-\varepsilon)$-fraction of clauses of $\tilde{I}$ are satisfied

---

PCPs are intimately tied to hardness of Approx

How to show hardness of Apx?

For optimization problem P (let's say maximization)
Create a Gap $P_{c,s}$ problem (decision problem)
& show it is NP-complete

$\Rightarrow$ NP-Hardness of factor $\left(\frac{s}{c}\right)$ for problem P.

Try to find c, s such that $\frac{s}{c}$ is as small as possible

This gap is related to accept prob. of verifier in PCP theorem.

"Parallel Repetition theorem"

Lots of research to trade-off $r(n)$, $s(n)$, accept probability

Culminated in a very neat abstraction called label cover problems.

equivalent form of PCP Thm + Parallel Repetition where verifier probes only 2 locations of the proof.

[Catch: proof is no longer bit string, its over a larger alphabet]

Max Label Cover Problems:

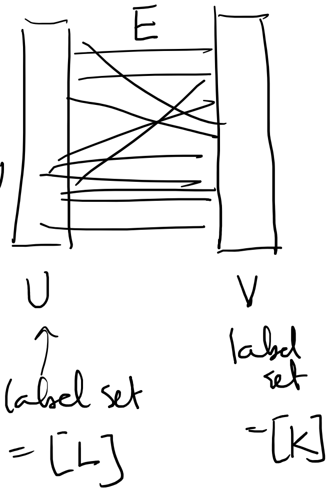
Given a bipartite graph $G = (U, V, E)$
and a "projection function" $f_{u \to v} : [L] \to [K]$
on each edge $(u,v) \in E$,

goal

Pick labels for each vertex (ie)

$\ell(u) \in [L]$ for all $u \in U$

$\ell(v) \in [K]$ for all $v \in V$



Max # edges for which labels are aligned wrt f.

(ie) $\boxed{f_{u \to v}(\ell(u)) = \ell(v)}$

Think of K and L as being constants

Gap Label Cover$_{1, \eta}$

Given an instance of label cover, can we decide if $\exists$ labelling which can satisfy all edges vs

if all labellings can satisfy $\leq \eta$ fraction of edges?

THEOREM:
For any constant $\eta > 0$, $\exists$ constant K, L bounded by poly$(1/\eta)$

Moreover

k, L bounded by poly$(1/\eta)$
Such that the GapLabelCover$_{1,\eta}$ is NP-complete.

Moreover, this holds even when $|U| = |V|$ and graph is regular

Many, if not most, hardness of approx. results come with the GapLabelCover as starting point.

$$\text{GapLabelCover}_{1,\eta} \xrightarrow{\text{reduction}} \text{GapP}_{c,s}$$

will get us some hardness for problem P.

Example Application (Hardness of Max-Coverage)

Problem:
Given a set system $(U, \mathcal{S})$ where U is universe of elements and $\mathcal{S}$ is collection of subsets of U, and given parameter k, goal: choose k sets from $\mathcal{S}$, say $S_1, S_2, \ldots, S_k \in \mathcal{S}$ to maximize

$$\left| \bigcup_{i=1}^{k} S_i \right|$$ (ie the # elts covered by them).

Algorithmic Ideas:-
① greedy algo: for $t = 1, 2, \ldots, k$. choose set which covers max # uncovered elts

② LP - rounding?

③ local search: Start with k random sets, and swap in a new set for an existing set if coverage improves

THM

$\boxed{\text{Greedy Algo is a } \left(1-\frac{1}{e}\right) \text{- approximation}}$

THM

For any fixed $\varepsilon > 0$, it is NP-complete to design a poly-time $\left(1-\frac{1}{e}+\varepsilon\right)$-approximation

PCP + Label Cover viewpoint

THM 2

For any fixed $\varepsilon > 0$, it is NP-complete to design a poly-time $(\ln n - \varepsilon)$-approx for Set cover. (Min # sets to cover all elts)

Recall: greedy algo is $\ln n$ - approx.

Next Lecture

Briefly outline the redn. from GapLabelCover $\rightarrow$ Gap Max Coverage.

to show slightly worse factors of $\left(\frac{3}{4}+\varepsilon\right)$ hardness of Approx.

Given an instance $\mathcal{I} = \{G = (U, V, E), \{f_{u \to v}\}\}$ of GapLabelCover problem, we'll create an instance $\mathcal{I}' = \{X, \mathcal{S}, k\}$ of GapCoverage s.t.

$$\circledast \begin{cases} \text{if} \quad Opt(\mathcal{I}) = 1 & \Longrightarrow \quad Opt(\mathcal{I}') = 1 \\ \quad Opt(\mathcal{I}') \leq \eta & \Longrightarrow \quad Opt(\mathcal{I}') \leq \frac{3}{4} + \varepsilon. \end{cases}$$

Here
Opt = fraction of satisfied edges

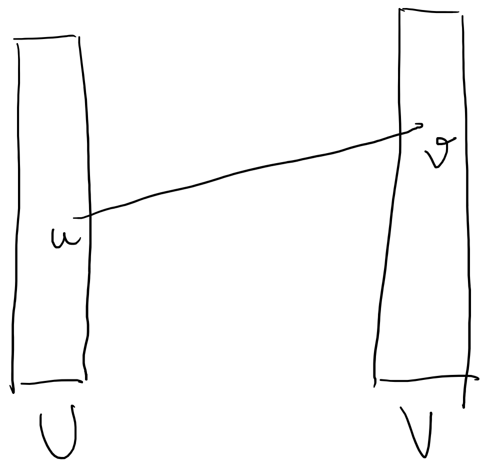
Here
Opt = fraction of covered elements.

From this,

NP-completeness of Gap Label Cover, we can get NP-completeness of Gap Coverage.

$\Rightarrow$ $\left(\frac{3}{4}+\varepsilon\right)$ - hardness of approx for Max Coverage.

<u>To do</u>
Come up with $\mathcal{I}'$ and prove $\circledast$ for it.



From $\mathcal{I}$, we want $\mathcal{I}'$ such that there is some correspondence between assigning a label l to u and picking a set in I'.

We'll create I' such that

there is a set $S_{u,\alpha}$ for all $u \in U$, $\alpha \in [L]$

and similarly

$S_{v,\beta}$ for all $v \in V$, $\beta \in [K]$

In total

$$\begin{aligned}\# \text{sets} &= |U| \cdot L + |V| \cdot K \\ &= |U| (L+K) \\ &= n(L+K)\end{aligned}$$

$\because |U| = |V| = n$ in input graph.

Next, we'll create some sort of association between edges in I with elements of I'.

For each edge $(u,v) \in G$, we'll create a number of elements.

2^K elements corresponding to K-bit strings

We'll refer to these elements as

$e_x^{(u,v)}$ where (u,v) is edge, x is a K-bit string

How many elements in I' have we created?

$$\#\text{edges} \cdot 2^k = d \cdot n \cdot 2^k$$

$\because$ graph is d-regular.

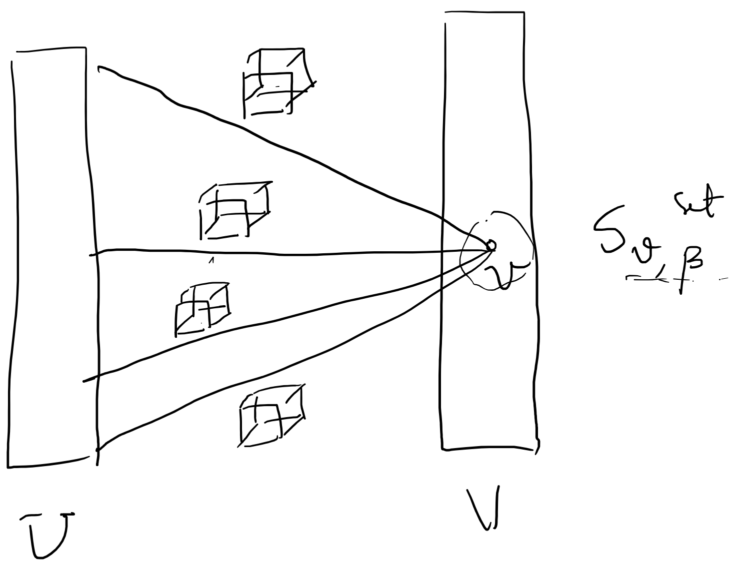
We've defined $\mathcal{X}$ (universe of elts) in I'

" " $\mathcal{S}$ (coll. of sets).

__Last__: When does a set $S_{u,\alpha}$ cover an element?

Ill^ly, when does a set $S_{v,\beta}$ cover an element?

Let's start with $S_{v,\beta}$.



__RULE 1:__ Firstly $S_{v,\beta}$ can only cover the elements on the

edges incident to v.

$\Rightarrow |S_{v,\beta}| \leq d \cdot 2^k$ (trivial bound)

Moreover,

Consider edge (u, v) and all elts $e_x^{(u,v)}$ for $x \in \{0,1\}^k$

RULE 2

Let's make $S_{v,\beta}$ cover all elts where the βth bit of $x = 1$.

In particular

$e_x^{(u,v)} \in S_{v,\beta}$ iff $x[\beta] = 1$
(β^{th} bit of x).

$\boxed{S_{v,\beta} \text{ is simply all such elts}}$

$\Rightarrow |S_{v,\beta}| = d \cdot 2^{k-1}$

Similarly, sets for all $v \in V$,
all $1 \leq \beta \leq k$.

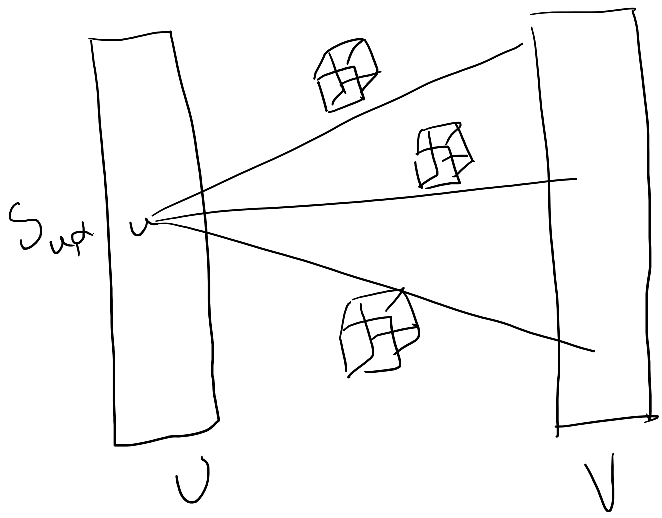
Next we'll define sets for U

Consider $S_{u,\alpha}$ for $u \in U$, $\alpha \in [L]$

$(u,v) \quad \ldots \quad x[\ldots(\alpha)] = 0$

$$e_x^{(u,v)} \in S_{u,\alpha} \text{ iff } x[f_{u,v}(\alpha)] = 0$$

(ie) the $f_{u,v}(\alpha)^{th}$ bit of $x = 0$.



$$|S_{u,\alpha}| = d \cdot 2^{k-1}$$

Also set k (in Gap k coverage) to be '2n'.

That defines I'. Why is it useful?

CLAIM ①

I is satisfiable, I' is fully coverable

(ie) $OPT(I) = 1 \Rightarrow OPT(I') = 1$

Proof.

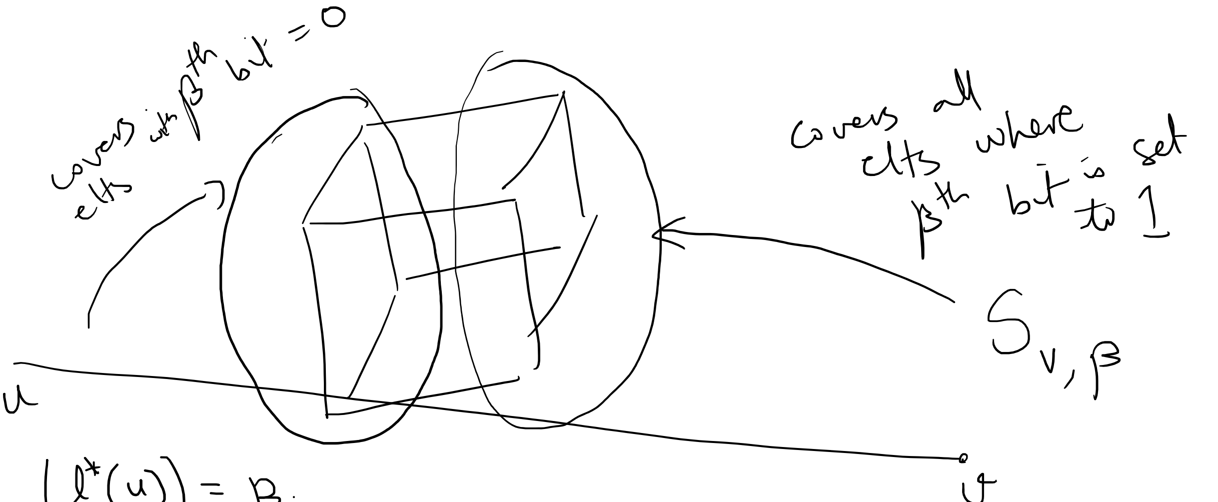
Let's consider the optimal labelling for I.

It can satisfy all edges.

(i.e.) $\ell^*(u)$, $\ell^*(v)$ are optimal labels
$u \in U$, $v \in V$

$\forall (u, v) \in G$, it holds that

$$\boxed{f_{u,v}(\ell^*(u)) = \ell^*(v)} = \beta$$
$(1 \le \beta \le K)$



$f_{u,v}(\ell^*(u)) = \beta$.

$\ell^*(v) = \beta$

Using this labeling ℓ^*, can we get a good sol^n for I' (GapKCoverage).

We can pick one set per vertex acc. ℓ^* labeling.

$\forall u \in U$, choose $S_{u, \ell^*(u)}$ and

$\forall\, v \in V$, choose $S_{v, \ell^*(v)}$.

$\#$sets selected $= 2n = k$.

$S_{v,\beta}$ covers all $e_x^{(u,v)}$ st $x[\beta] = 1$

$S_{u,\ell^*(u)}$ covers all $e_x^{(u,v)}$ st $x[\beta] = 0$.

$\Rightarrow$ together they cover all elts ☺

Remains to show:-

$$\mathrm{Opt}(I) \le \eta \Rightarrow \mathrm{Opt}(I') \le \frac{3}{4} + \varepsilon.$$

Instead we'll show

$$\mathrm{Opt}(I') > \frac{3}{4} + \varepsilon \Rightarrow \mathrm{Opt}(I) > \eta.$$

Way we'll show it:-

If there is a good sol^n for gap coverage covering $\geq \left(\frac{3}{4} + \varepsilon\right)$ fraction of elts, then we can recover a good labeling satisfying $\geq \eta$ fraction of edges.

High-level sketch :-

Let S^* be a good cover covering $\geq (\frac{\beta}{4}+\varepsilon)$ fraction of elts.

$|S^*| = k = 2n$.

Sps

Cheating assumption

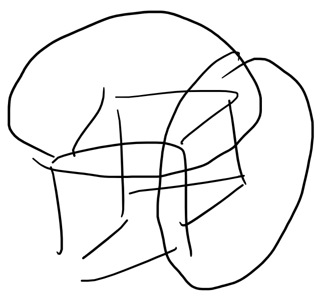
S^* is such that it picks one set per vertex of the graph G of I.

Then we ask : Can we get a good labeling for I.

If S^* contains $S_{u,\alpha}$, then assign label $l^*(u) = \alpha$

III^{ly} If S^* contains $S_{v,\beta}$ then assign label $l^*(v) = \beta$.

Then, claim : $\geq 4\varepsilon$ fraction of edges need to have a satisfied labeling.



Any badly labeled edge can cover only $\le 3/4$ fraction of elts.

But since S^* covers $\ge \left(\frac{3}{4}+\varepsilon\right)$ fraction of elts, there must be a good # of satisfied edges

$$(1-\delta)\cdot\frac{3}{4} + \delta\cdot 1 = \frac{3}{4}+\varepsilon$$

$$\frac{3}{4} + \frac{\delta}{4} = \frac{3}{4}+\varepsilon$$

$$\boxed{\delta = 4\varepsilon}$$