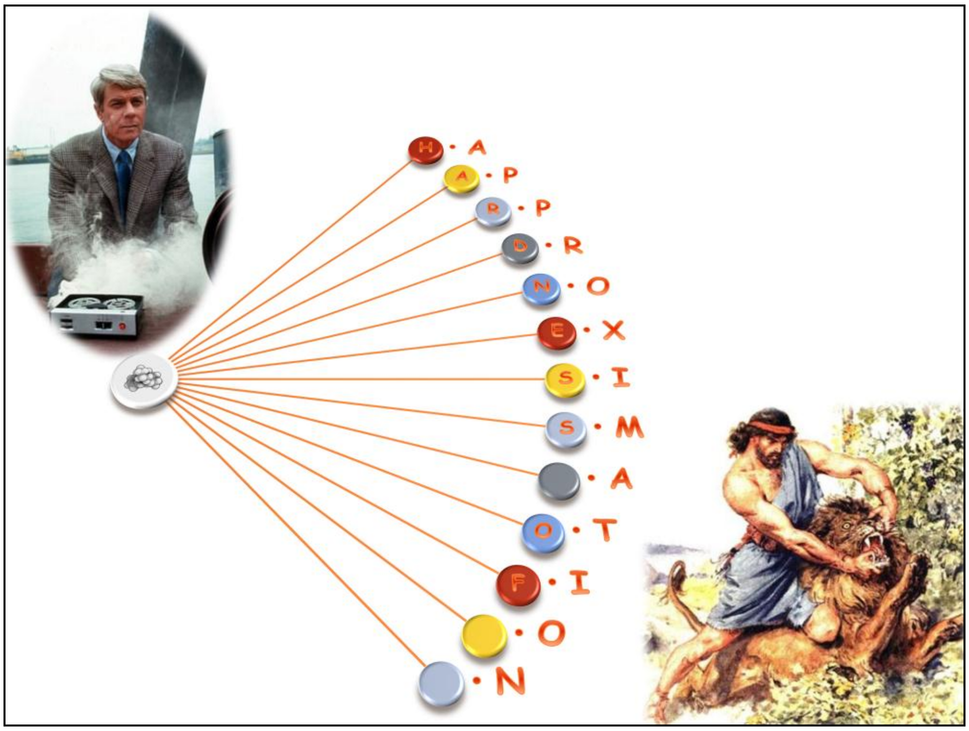
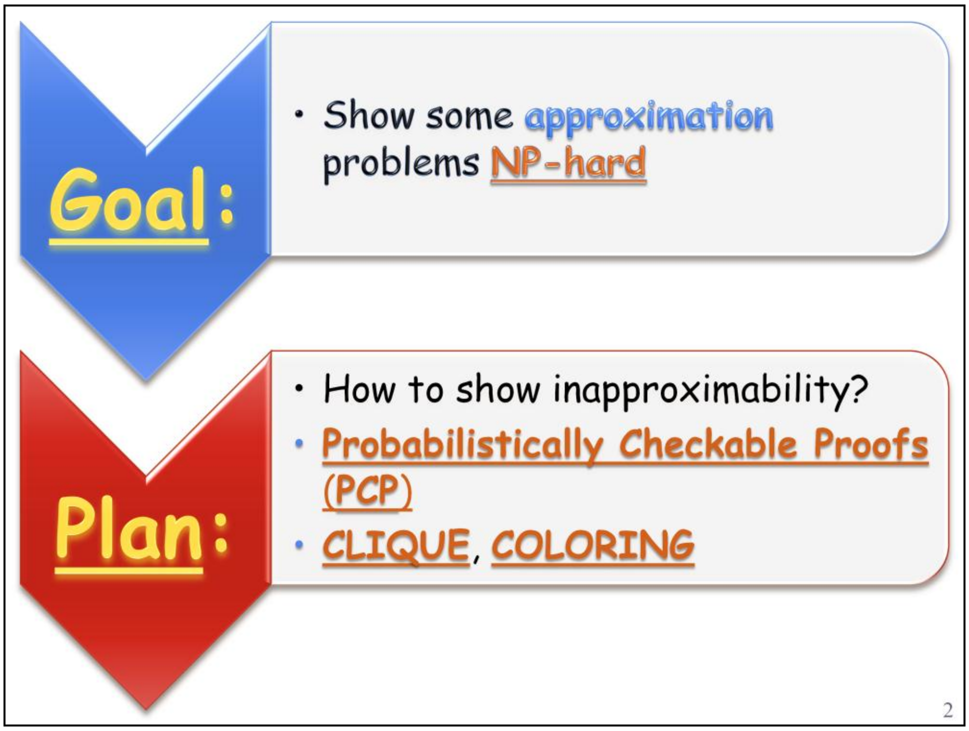


In this lecture we discuss the complexity of approximation problems, and show how to prove they are NP-hard.



Goal:

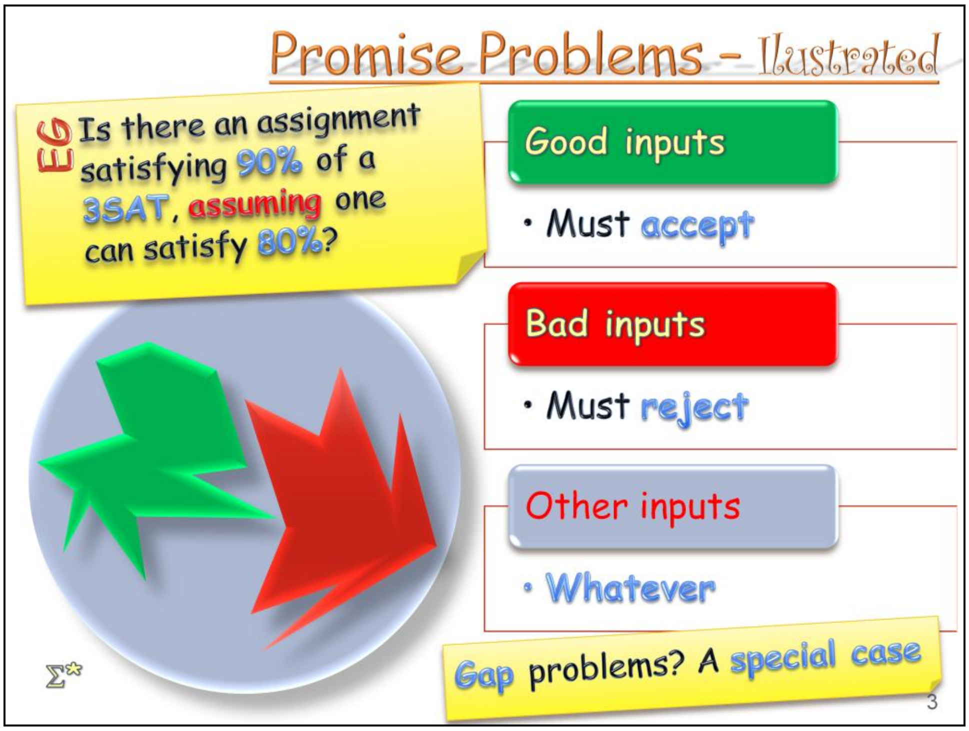
- Show some approximation problems NP-hard

Plan:

- How to show inapproximability?
- Probabilistically Checkable Proofs (PCP)
- CLIQUE, COLORING

2

We will show how one can prove such results and then apply this technique to some approximation problems.



Promise Problems – Ilustrated

EG Is there an assignment satisfying 90% of a 3SAT, assuming one can satisfy 80%?

Good inputs
- Must accept

Bad inputs
- Must reject

Other inputs
- Whatever

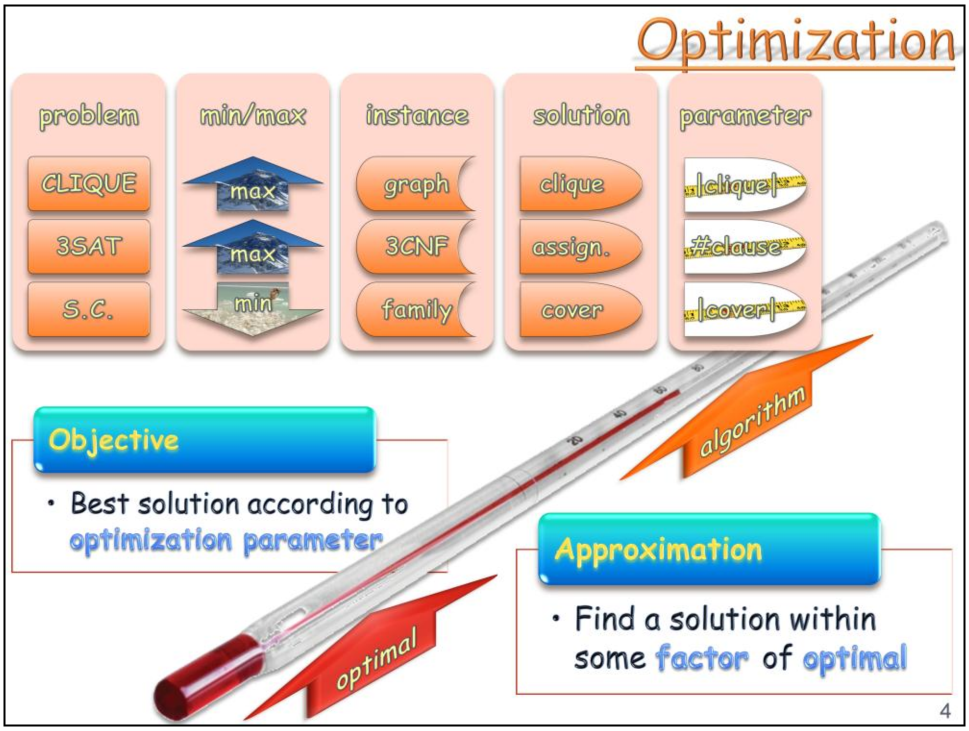
Gap problems? A special case

Let us start with a general definition of promise problems:

These are problems in which the algorithm is required to accept some inputs and reject others, however, unlike in algorithm for languages, some inputs are "don't care", and the algorithm may return whatever answer.

The problem is therefore easier, and showing such a problem is NP-hard implies all languages that agree with it on the good and bad inputs are NP-hard as well.

Gap problems are a special case of promise problems.


Optimization

problem	min/max	instance	solution	parameter
CLIQUE	max	graph	clique	|clique|
3SAT	max	3CNF	assign.	#clause
S.C.	min	family	cover	|cover|

Objective

- Best solution according to optimization parameter

Approximation

- Find a solution within some factor of optimal

algorithm
optimal

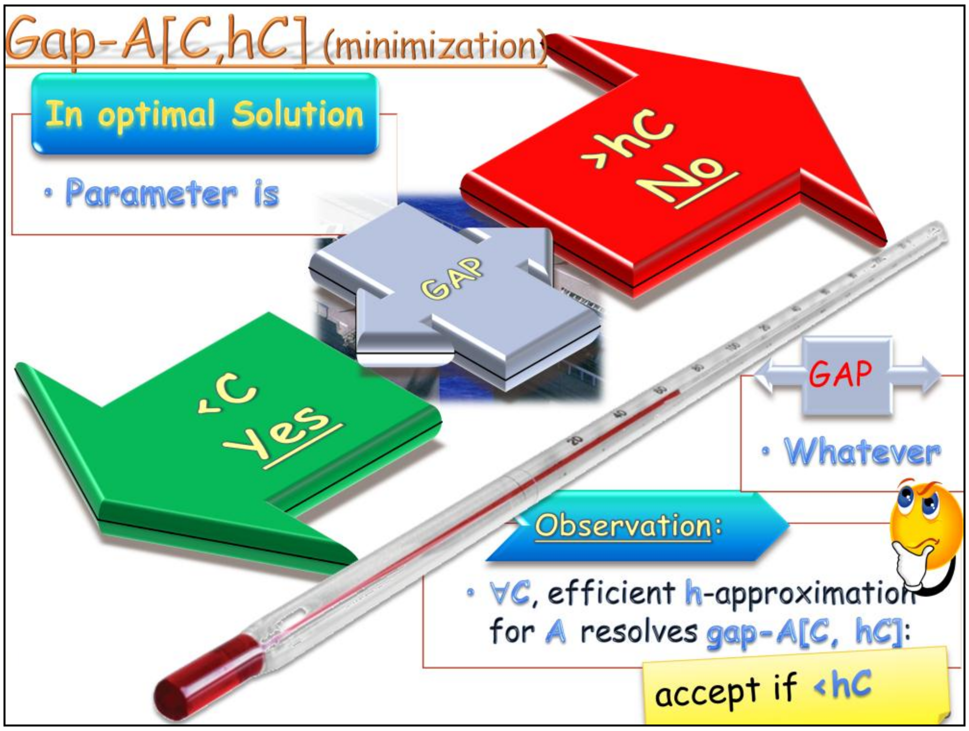
4

Let us recall the general framework of optimization problems --- either minimization or maximization problems.

These problems allow some type of *solutions* and measure those solutions according to the *optimization parameter*.

The goal is to find the *best* solution according to the parameter --- one which either minimizes or maximizes the parameter.

An *approximation* algorithm is guaranteed to find a solution which, although not optimal, is nevertheless within the *approximation factor* from optimal.



Gap-A[C,hC] (minimization)

In optimal Solution

- Parameter is

- Whatever

Observation:

- ∀C, efficient h-approximation for A resolves gap-A[C, hC]:

accept if <hC

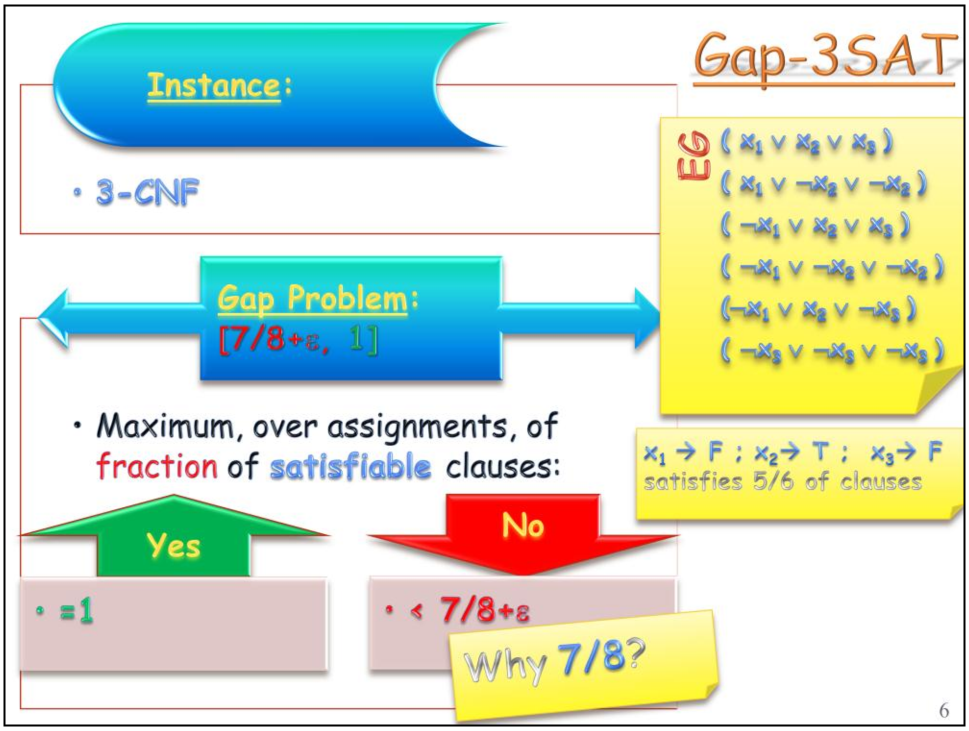
Gap problems partitions all *input* instances into *3 categories according* to the optimized parameter of *best solution*.

It introduces two *thresholds*:

- If the optimal solution is better than the good threshold, the input is good.
- If it is worse than the bad solution the input is bad.
- In the in between ***gap*** the input is a "don't care" input.

An approximation algorithm whose factor of approximation is better than the ratio between the two thresholds can be easily adapted to solve the gap problem.

Hence, if the gap problem is NP-hard then so is the approximation problem.



Gap-3SAT

Instance:

- 3-CNF

Gap Problem:
$[7/8+\varepsilon,\ 1]$

- Maximum, over assignments, of fraction of satisfiable clauses:

Yes
- $=1$

No
- $< 7/8+\varepsilon$

EG
$(x_1 \vee x_2 \vee x_3)$
$(x_1 \vee \neg x_2 \vee \neg x_2)$
$(\neg x_1 \vee x_2 \vee x_3)$
$(\neg x_1 \vee \neg x_2 \vee \neg x_2)$
$(\neg x_1 \vee x_2 \vee \neg x_3)$
$(\neg x_3 \vee \neg x_3 \vee \neg x_3)$

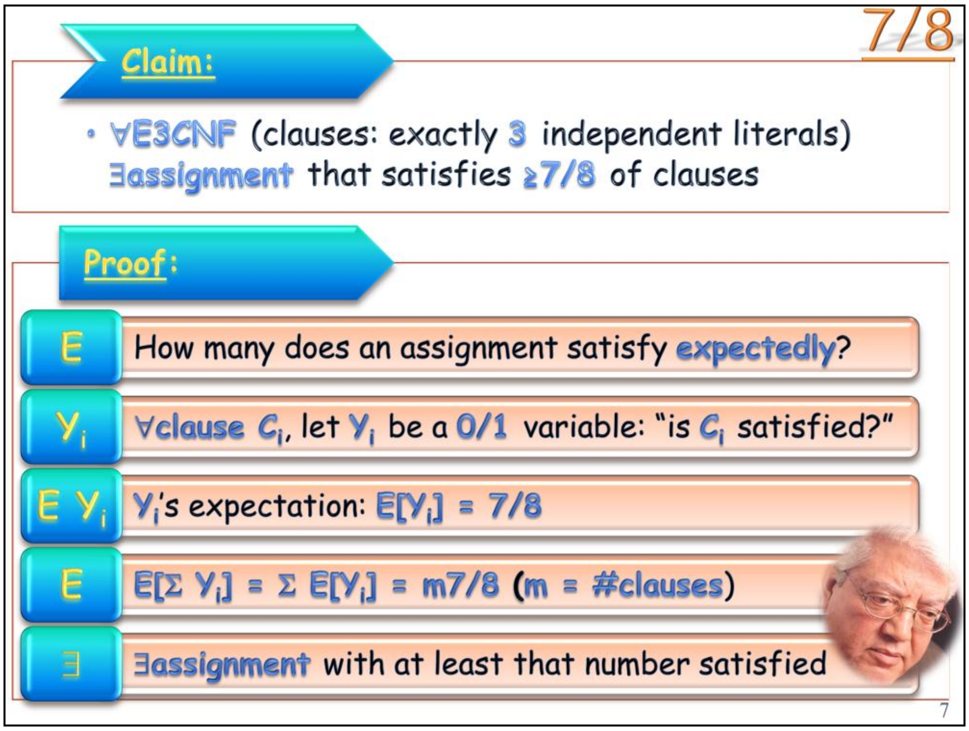
$x_1 \rightarrow F$; $x_2 \rightarrow T$; $x_3 \rightarrow F$
satisfies 5/6 of clauses

Why 7/8?

6

As far as 3SAT is the grandparent of all NP-complete problems, the following gap version can serve the same role for gap problems:

- The input is the same, a 3CNF formula
- The good threshold is 1, namely, the same as in 3SAT all formulas that are completely satisfiable.
- Bad inputs are those for which only 7/8+epsilon fraction of the clauses can be satisfied by any given assignment.
  We will next see why we chose the 7/8 fraction.



7/8

Claim:

- ∀E3CNF (clauses: exactly 3 independent literals) ∃assignment that satisfies ≥7/8 of clauses

Proof:

E	How many does an assignment satisfy expectedly?
Y_i	∀clause C_i, let Y_i be a 0/1 variable: "is C_i satisfied?"
E Y_i	Y_i's expectation: $E[Y_i] = 7/8$
E	$E[\Sigma Y_i] = \Sigma E[Y_i] = m7/8$ (m = #clauses)
∃	∃assignment with at least that number satisfied

7

In general, any E3SAT (that has clauses with exactly 3 independent literals) has an assignment that satisfies 7/8 of the clauses.

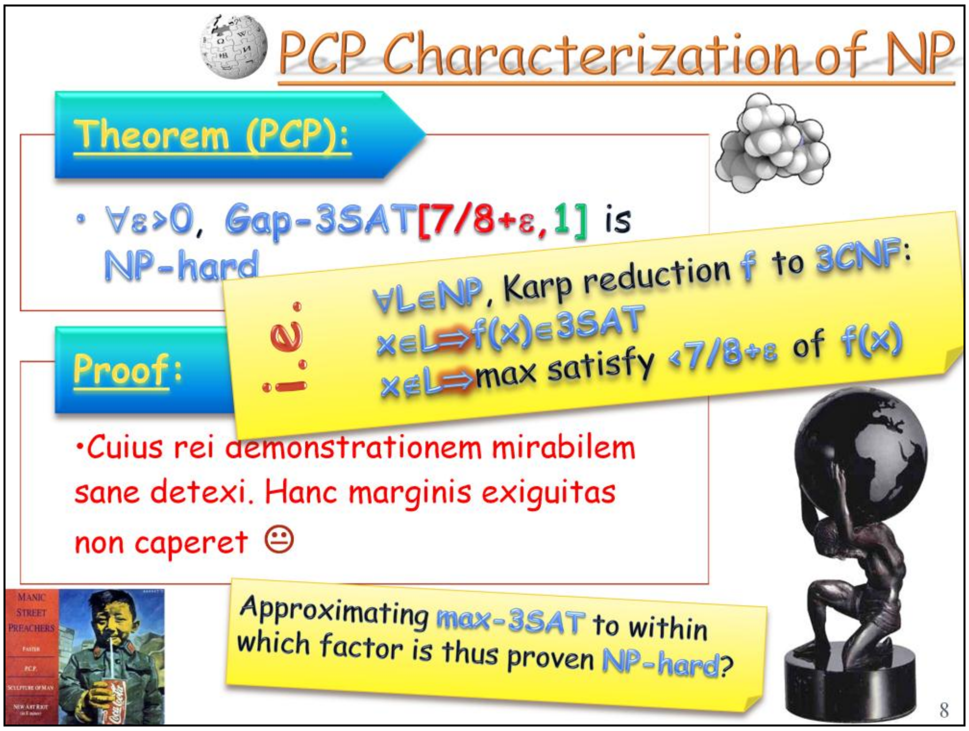
To show that, we apply a very simple *probabilistic method* argument: we show the probability, over a random assignment to the variables, to satisfy 7/8 of clauses, is positive, hence there must be such an assignment.

What is the *average*, uniformly *over all assignments* of the fraction of clauses satisfied? We show it is 7/8.

To see that, assume a variable Y_i that is 1 if clause C_i is satisfied and 0 otherwise. For any i, the average value of Y_i is 7/8, as 7 of all 8 possible assignments to its 3 variables satisfy it.

Now look at the average number of clauses satisfied: it is the same as the sum of averages (by linearity of expectation or in simple terms change of order of summation) --- that sum is of course simply 7/8 of the number of clauses.

Now, by the law of averages, there *must be* at least *one assignment* that achieves this average.



PCP Characterization of NP

Theorem (PCP):

- $\forall \varepsilon > 0$, Gap-3SAT$[7/8+\varepsilon, 1]$ is NP-hard

i.e.: $\forall L \in NP$, Karp reduction f to 3CNF:
$x \in L \Rightarrow f(x) \in 3SAT$
$x \notin L \Rightarrow$ max satisfy $< 7/8+\varepsilon$ of $f(x)$

Proof:

- Cuius rei demonstrationem mirabilem sane detexi. Hanc marginis exiguitas non caperet ☹

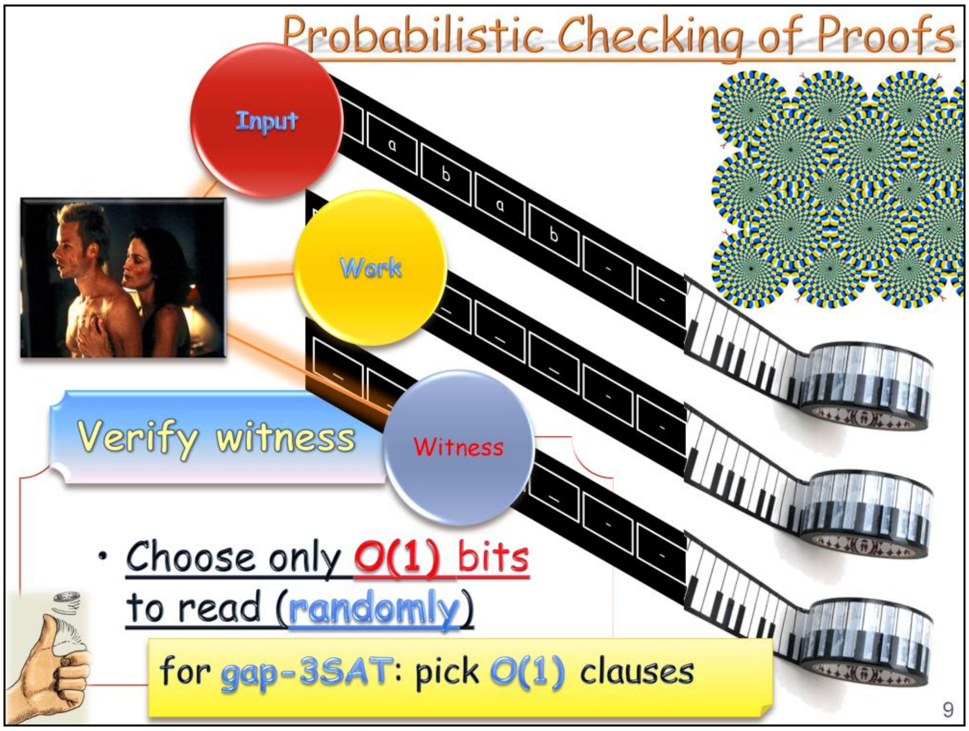
Approximating max-3SAT to within which factor is thus proven NP-hard?

8

Now, let us go back to gap-3SAT and state one of the versions of the *PCP theorem*, which is, that the problem is NP-hard. That is, any NP problem can be Karp-reduced to a 3CNF instance, so that if the input is good the outcome of the reduction (as in Cook/Levin theorem) is a completely satisfiable formula. If the *input is bad*, however, the reduction results in a formula for which the *maximum* satisfiable fraction of clauses is only slightly above *7/8*.

This can be viewed as an alternative, much stronger *characterization of NP* than the one of Cook/Levin.

The proof of this theorem is possibly the most elaborated in Computer Science with a matching impact, and not too many mathematical proofs beat it in that respect. It is hence way beyond the scope of this course. Nevertheless, we'll assume it is true and proceed to show some of its fundamental implications.



Probabilistic Checking of Proofs

Input
Work
Witness
a
b
a
b

Verify witness

- Choose only O(1) bits to read (randomly)

for gap-3SAT: pick O(1) clauses

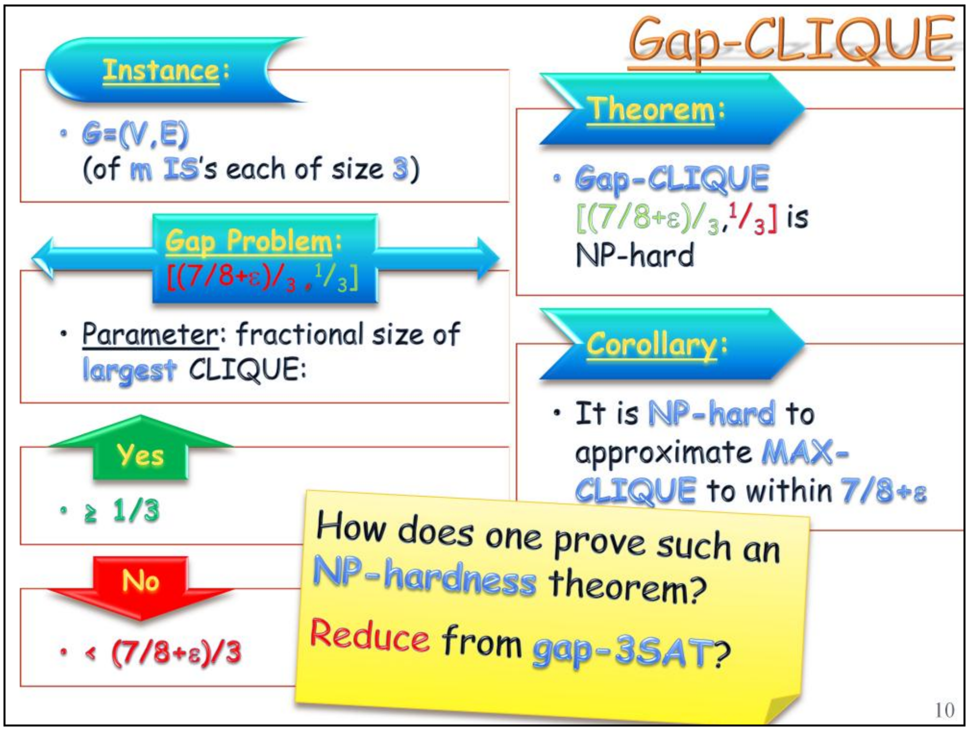
9

Recall the characterization of NP we used before, namely as all languages for which a witness of membership can be verified efficiently.

Now consider a very limited verifier for the membership proof --- one who is only allowed to read a constant number of bits of the witness proof. Nevertheless, it may choose which bits to read by flipping *random* coins, and then may *err* with small probability (accept a bad input, that is, fail to discover an error in the membership proof).

Our gap-3SAT above is accepted by this framework: assuming the membership proof is simply an assignment to the formula's variables, the verifier can choose a constant number of clauses and see if they are satisfiable. A satisfying assignment passes this test with probability 1. In case no assignment can satisfy 7/8+epsilon of the clauses the probability of all chosen clauses to be satisfied becomes arbitrarily small.

It therefore follows (assuming the PCP theorem) that all languages in NP have membership proof that can be verified probabilistically reading only a constant number of their bits.



Gap-CLIQUE

Instance:

- $G=(V,E)$ (of m IS's each of size 3)

Gap Problem: $[(7/8+\varepsilon)/_3, {}^1/_3]$

- Parameter: fractional size of largest CLIQUE:

Yes

- $\geq 1/3$

No

- $< (7/8+\varepsilon)/3$

Theorem:

- Gap-CLIQUE $[(7/8+\varepsilon)/_3, {}^1/_3]$ is NP-hard

Corollary:

- It is NP-hard to approximate MAX-CLIQUE to within $7/8+\varepsilon$

How does one prove such an NP-hardness theorem?

Reduce from gap-3SAT?

10

Now, let us consider other approximation problems, e.g. max-CLIQUE.

How about approximating it?

Look at a special type of graphs: consisting of m pair-wise disjoint independent sets, each containing 3 vertexes.

If we prove hardness for this special case, it still applies to the general case (however we may later utilize this special structure).

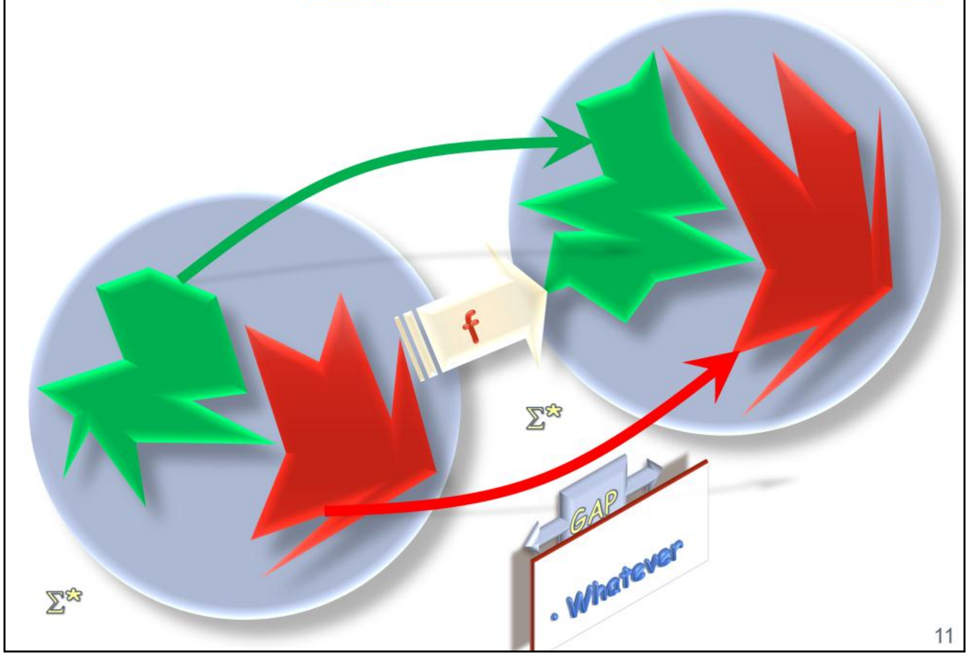
Now, define a gap problem, where good inputs have a clique with a representative in every independent set, while the largest clique in a bad input is of size less than 7/8+epsilon of that number.

This problem is NP-hard.

To prove such a theorem, we need to introduce a special type of reduction, and apply one such from gap-3SAT to it.

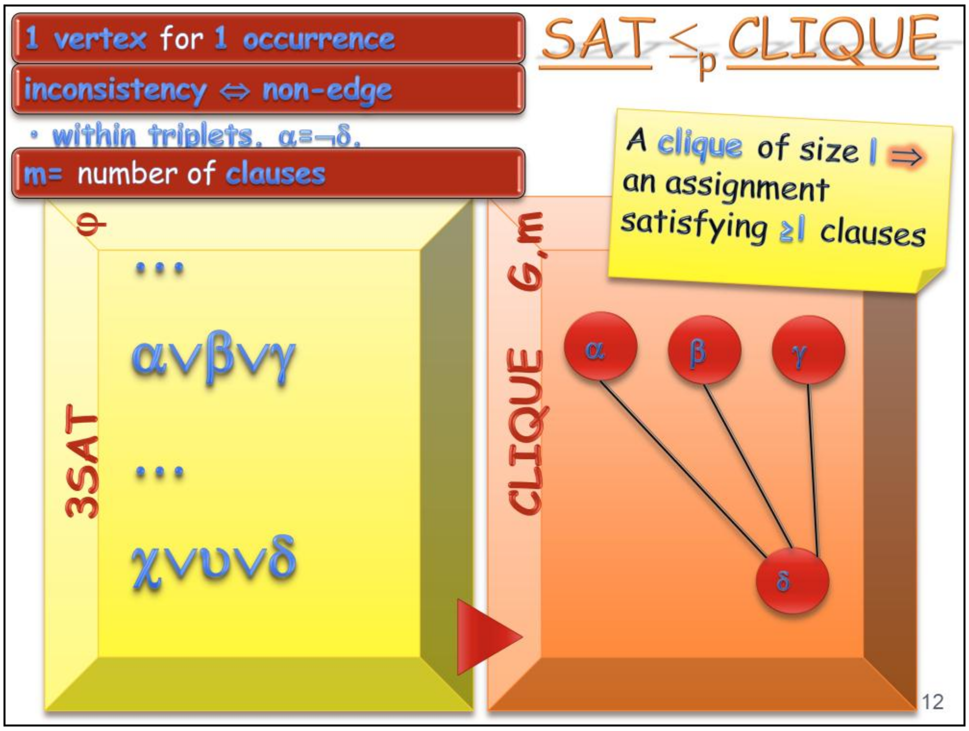
As a corollary max-CLIQUE is NP-hard to approximate to within the corresponding factor.

Gap Preserving Reduction



A *gap-preserving* Karp-reduction, from one gap-problem to another, is one that takes

- good inputs to good inputs,
- bad inputs to bad one, and
- "don't care" inputs to whatever inputs it cares to take it to.



1 vertex for 1 occurrence

inconsistency ⇔ non-edge

- within triplets, α≡¬δ,

m= number of clauses

SAT $\leq_p$ CLIQUE

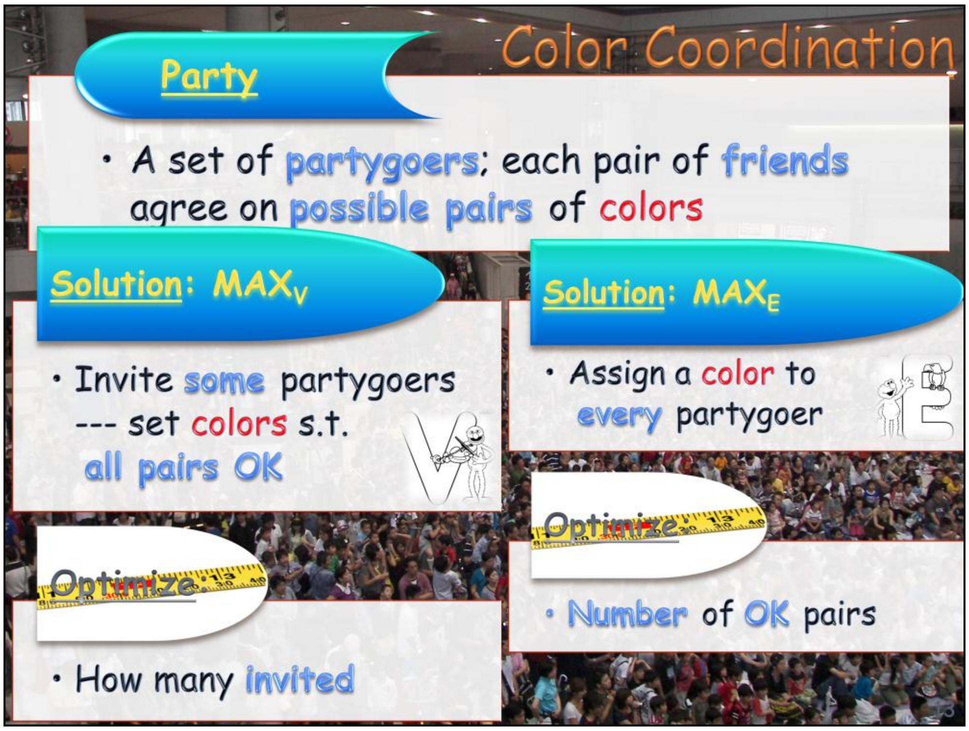
A clique of size l ⇒ an assignment satisfying ≥l clauses

We now revisit the same reduction we used to show that CLIQUE is NP-hard, and prove it to be a gap preserving reduction, between gap-3SAT to gap-CLIQUE with the same gap.

Completeness is clearly the same (it is exactly the same statement).

As to *soundness*: note that a clique of l vertexes can be utilized to construct an assignment, by making TRUE all literals appearing in it, which must satisfy all clauses for which the clique has a representative in their corresponding triplet.

Hence, the assignment satisfies at least l/m clauses.



Color Coordination

Party

- A set of partygoers; each pair of friends agree on possible pairs of colors

Solution: MAX_V

- Invite some partygoers --- set colors s.t. all pairs OK

Optimize:

- How many invited

Solution: MAX_E

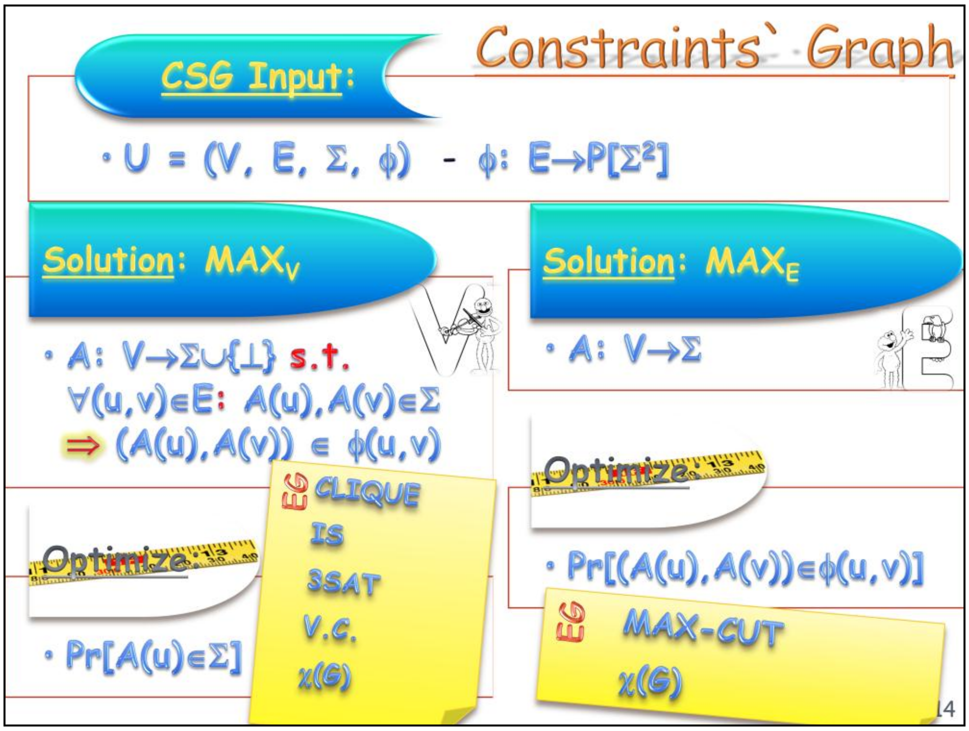
- Assign a color to every partygoer

Optimize:

- Number of OK pairs

Assume you'd like to *invite* a group of people to a *party*.
As it turns out, however, every pair of friends of those partygoers may have very particular conditions as to *what to wear* to the party, in particular, which *pairs of choices* of clothes for both of them constitute an acceptable pair.

One now either invites everyone, doing best to minimize conflict, or tries to figure out the largest set of guests, and a choice of clothes to them all so as to avoid any conflict.



Constraints` Graph

CSG Input:

- $U = (V, E, \Sigma, \phi)$ - $\phi: E \rightarrow P[\Sigma^2]$

Solution: MAX_V

- $A: V \rightarrow \Sigma \cup \{\perp\}$ s.t. $\forall (u,v) \in E: A(u), A(v) \in \Sigma \Rightarrow (A(u), A(v)) \in \phi(u,v)$

Optimize

- $Pr[A(u) \in \Sigma]$

EG
CLIQUE
IS
3SAT
V.C.
$\chi(G)$

Solution: MAX_E

- $A: V \rightarrow \Sigma$

Optimize

- $Pr[(A(u), A(v)) \in \phi(u,v)]$

EG
MAX-CUT
$\chi(G)$

14

Let us now generalize some optimization problems we have discussed earlier, by introducing the *constraints graph problem*:

The input is a graph, a set of possible values for the vertexes (colors), and a set of constraints, specifying for each edge which pairs of colors are allowed for its ends.

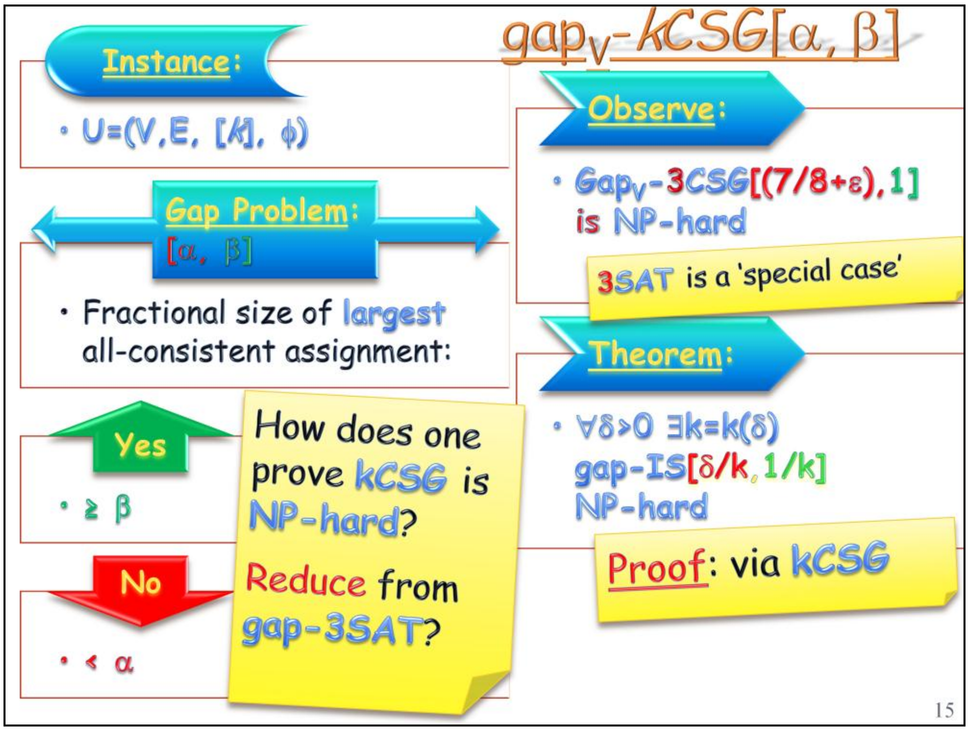
There are two possible variants of the problem:

In the first, we allow vertexes to remain uncolored, however require that all constraints between colored vertexes are satisfied. In this case, the *parameter* to maximize is the fraction of *vertexes colored*.

The other, more natural, variant, colors all vertexes and the maximized parameters is the fraction of edges whose constraints are satisfied.

Numerous optimization problems we've looked at can be formulated as a special case of this general definition.

See if you can identify for those where is the relevant gap for which the problem is easy, and where it may be hard.



gap_V-$kCSG[\alpha, \beta]$

Instance:

- $U=(V,E,[k],\phi)$

Gap Problem: $[\alpha, \beta]$

- Fractional size of largest all-consistent assignment:

Yes

- $\geq \beta$

No

- $< \alpha$

How does one prove kCSG is NP-hard?

Reduce from gap-3SAT?

Observe:

- Gap_V-3CSG[(7/8+ε),1] is NP-hard

3SAT is a 'special case'

Theorem:

- $\forall\delta>0\ \exists k=k(\delta)$ gap-IS[δ/k,1/k] NP-hard

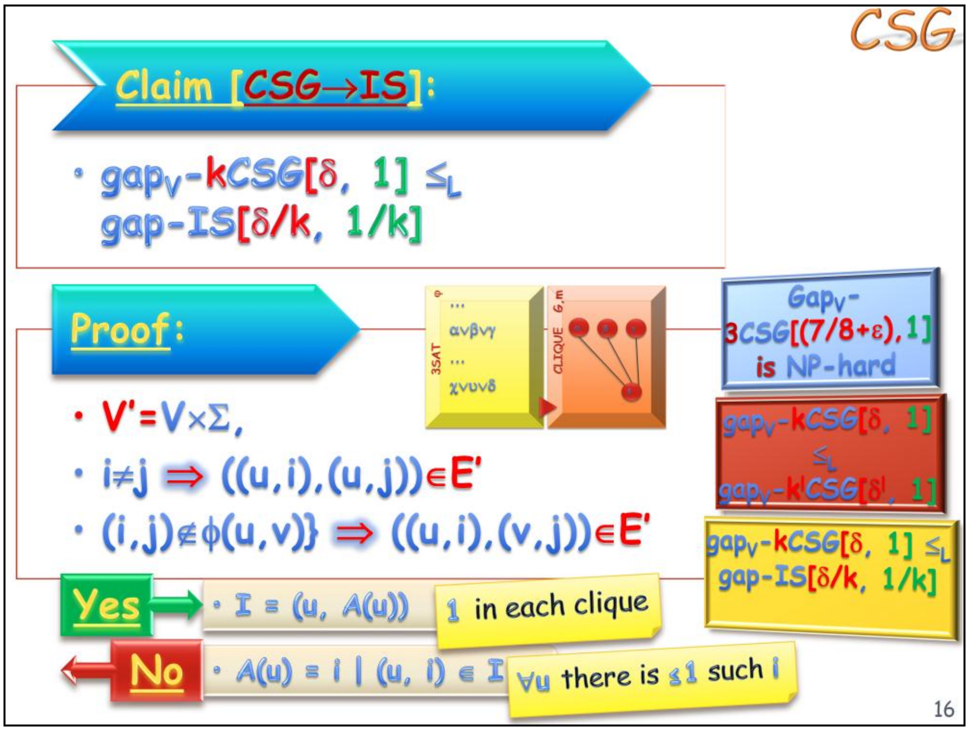
Proof: via kCSG

15

Now, concentrating on the first time of optimization, let us define the appropriate gap problem, for two arbitrary thresholds, and where the number of colors is limited (denoted by k).

One can immediately observe that gap_V-3CSG (gap; optimizing vertexes; 3 colors) for the thresholds 7/8+ε and 1, is NP-hard.

Next, let us go back to showing hardness for approximating max-IS (or max-CLIQUE) to within any constant is NP-hard (and where the fractional size of the IS becomes smaller as the factor becomes larger).



Claim [CSG→IS]:

- $\text{gap}_V\text{-kCSG}[\delta, 1] \leq_L \text{gap-IS}[\delta/k, 1/k]$

Proof:

- $V' = V \times \Sigma$,
- $i \neq j \Rightarrow ((u,i),(u,j)) \in E'$
- $(i,j) \notin \phi(u,v)\} \Rightarrow ((u,i),(v,j)) \in E'$

Yes → • $I \equiv (u, A(u))$ 1 in each clique

No ← • $A(u) \equiv i \mid (u, i) \in I$ $\forall u$ there is ≤ 1 such i

φ
...
α∨β∨γ
...
χ∨υ∨δ
3SAT
CLIQUE
G,m

$\text{Gap}_V\text{-3CSG}[(7/8+\varepsilon),1]$ is NP-hard

$\text{gap}_V\text{-kCSG}[\delta, 1] \leq_L \text{gap}_V\text{-k'CSG}[\delta', 1]$

$\text{gap}_V\text{-kCSG}[\delta, 1] \leq_L \text{gap-IS}[\delta/k, 1/k]$

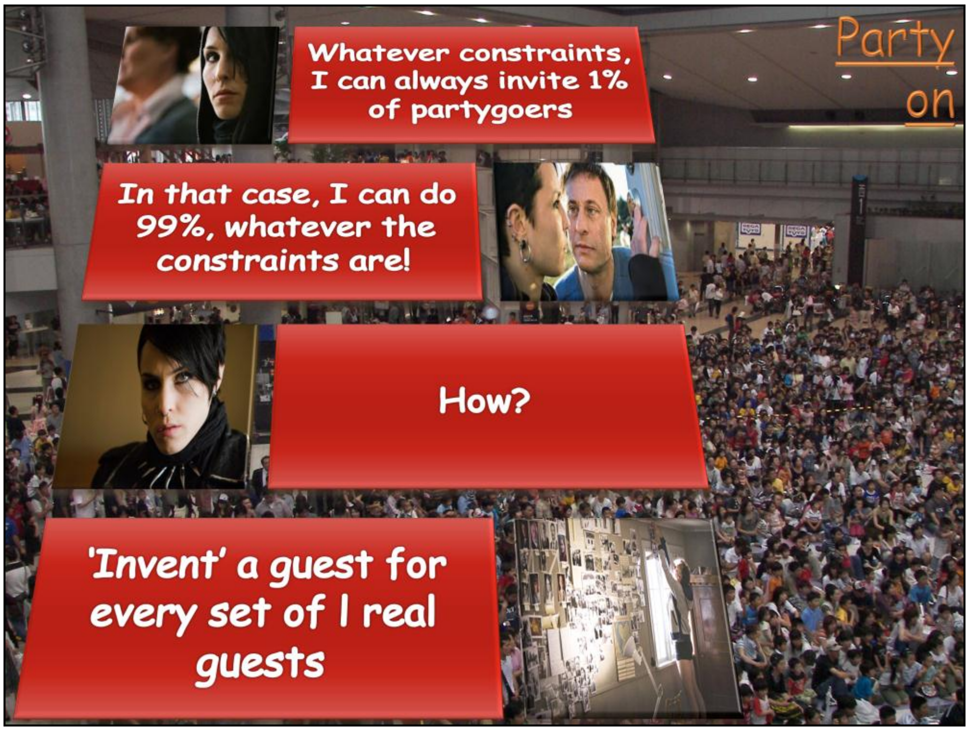
16

When looking at the gap versions for the problem, even when the alphabet (colors) set is of size 3, for the appropriate gap, the problem is NP-hard, which is the case as 3SAT can be reduced to it.

So we have established NP-hardness of gap_V-3CSG.

Another important observation is that any constraint problem can be directly reduced to an Independent Set (or CLIQUE) problem, by specifying a vertex, for each pair of a vertex and color of the constraints graph, and incorporating the appropriate edges.

This is in fact the reduction we've just seen from 3SAT to CLIQUE --- it turns out to be a general reduction from constraints problems to IS or CLIQUE.



Party on

Whatever constraints, I can always invite 1% of partygoers

In that case, I can do 99%, whatever the constraints are!

How?

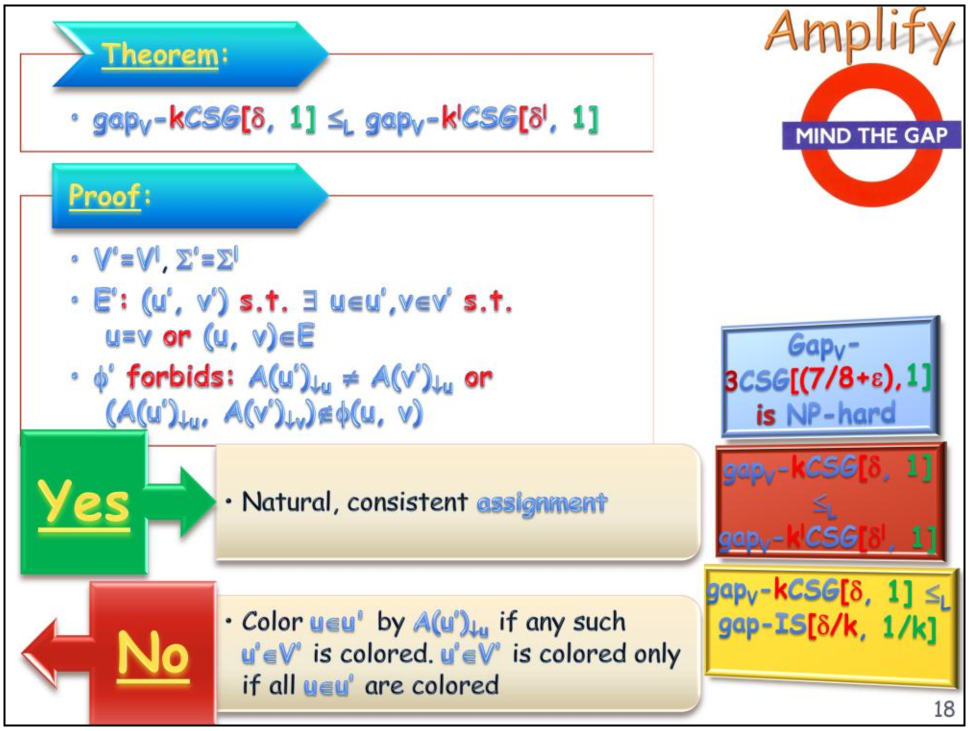
'Invent' a guest for every set of l real guests

Suppose someone says they can approximate the CSG problem as follows:

Given there is a coloring to all vertexes that satisfies all constraints, they can color some δ fraction of the vertexes satisfying all constraints.

Can one use such an algorithm to color a larger fractions?

Following the party allegory from above, all one needs to do is 'invent' a new, virtual party, in which guests are all sets of l guests of the real party, clothes assign one option to every member of the set, and constraints are natural; then apply the algorithm to that new party.



Amplify

MIND THE GAP

Theorem:

- $\text{gap}_V\text{-kCSG}[\delta, 1] \leq_L \text{gap}_V\text{-k}^l\text{CSG}[\delta^l, 1]$

Proof:

- $V' = V^l$, $\Sigma' = \Sigma^l$
- E': (u', v') s.t. $\exists\, u \in u', v \in v'$ s.t. $u = v$ or $(u, v) \in E$
- ϕ' forbids: $A(u')_{\downarrow u} \neq A(v')_{\downarrow u}$ or $(A(u')_{\downarrow u}, A(v')_{\downarrow v}) \notin \phi(u, v)$

Yes

- Natural, consistent assignment

No

- Color $u \in u'$ by $A(u')_{\downarrow u}$ if any such $u' \in V'$ is colored. $u' \in V'$ is colored only if all $u \in u'$ are colored

$\text{Gap}_V\text{-3CSG}[(7/8+\varepsilon), 1]$ is NP-hard

$\text{gap}_V\text{-kCSG}[\delta, 1] \leq_L \text{gap}_V\text{-k}^l\text{CSG}[\delta^l, 1]$

$\text{gap}_V\text{-kCSG}[\delta, 1] \leq_L \text{gap-IS}[\delta/k, 1/k]$

18

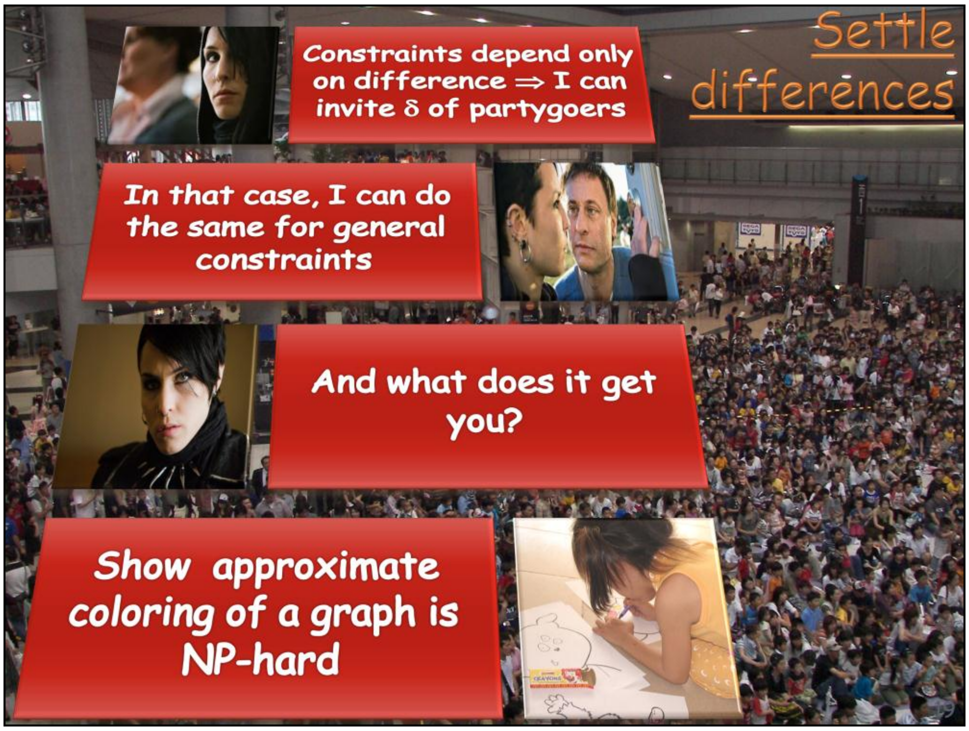
It turns out we can in general "amplify" a constraints problem:

The trick goes as follows. Given a constraints graph, construct a new graph in which every vertex corresponds to a sequence of l vertexes of the original graph. Each new color assigns a regular, old color to every vertex in the l-sequence. The new constraints prohibit any inconsistency in coloring that may occur. Namely, if two new vertexes are colored so that the same original vertex is colored differently, or if colors to the two ends of an original edges do not satisfy the constraint corresponding to that edge.

Given a consistent coloring of the original graph, one can naturally extend it to the new graph:

suppose you have a coloring A' of V^l. Let A color a vertex v of the original graph according to any l-sequence u' containing v. Colors assigned by A are everywhere consistent. If A colors δ fraction of V, A' must had avoided any l-sequence u' containing any vertex in V uncolored by A, which implies it must be of size at most δ^l.

Can one similarly amplify the other variant of the CSG problem?



Settle differences

Constraints depend only on difference ⇒ I can invite δ of partygoers

In that case, I can do the same for general constraints

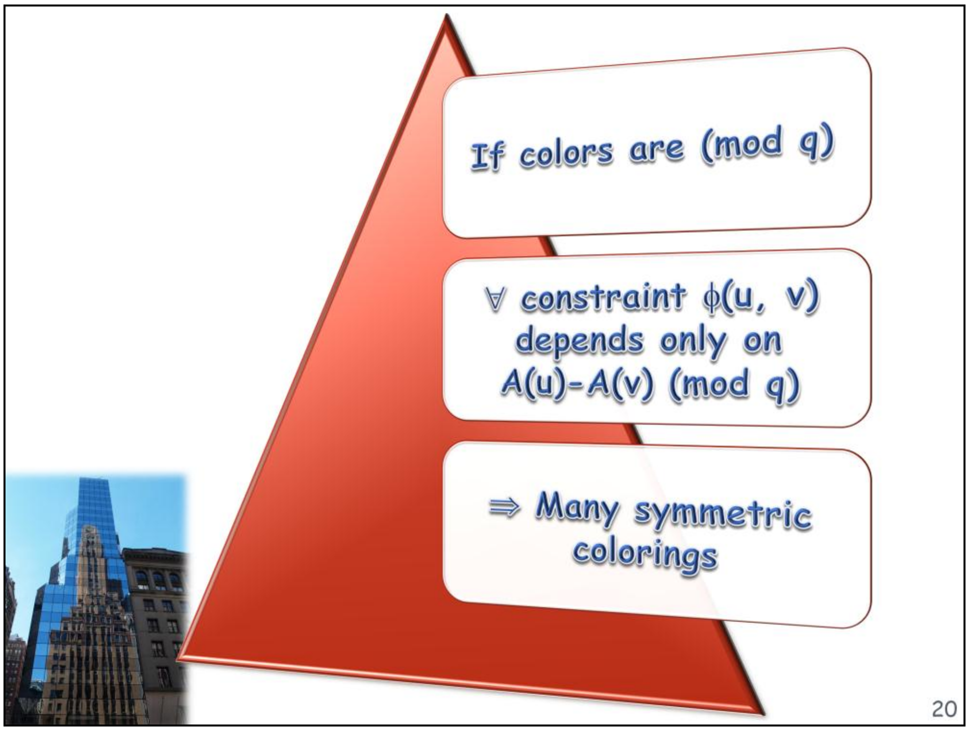
And what does it get you?

Show approximate coloring of a graph is NP-hard

Suppose now someone claims an algorithm that can color a constraints graph, provided the constraints take a very special form:

- Colors are {0..q-1} for some q, and any constraint specifies which pairs of differences are allowed between the two colors at the end of the edge.

We'll show this can be used to color a general CSG and furthermore is very useful to show hardness of approximate coloring of a graph.



If colors are (mod q)

∀ constraint $\phi(u, v)$ depends only on A(u)-A(v) (mod q)

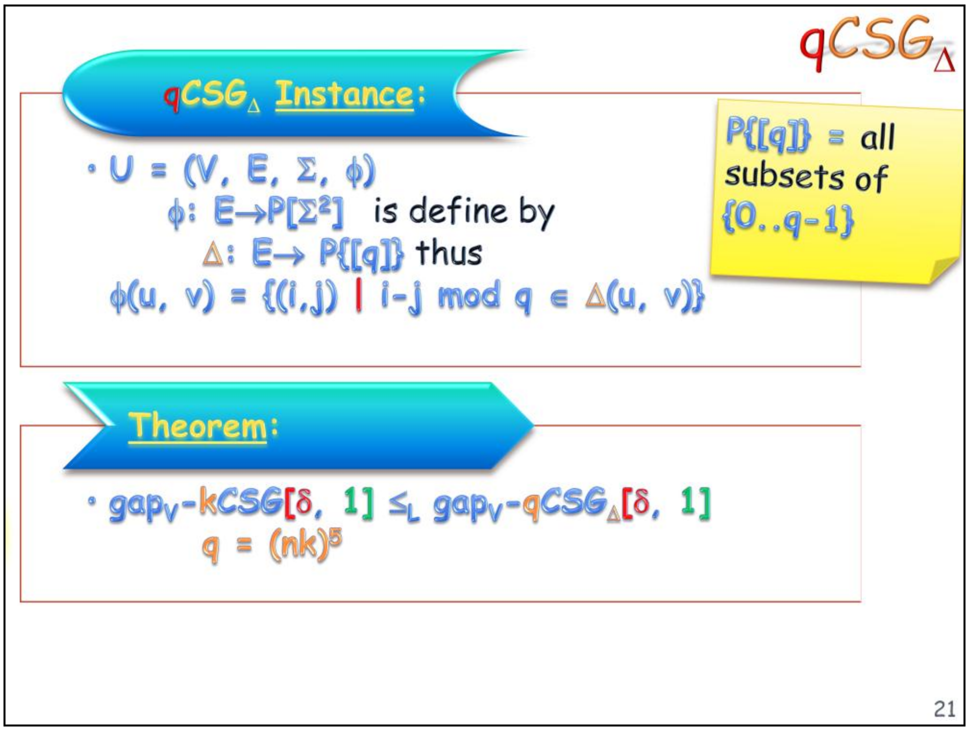
⇒ Many symmetric colorings

20

Essentially, we change the problem so that there are many symmetric solutions?

Can we transform the general CSG problem to one in which colors are 0…q-1 and all constrains simply specify some differences (mod q) that are allowed?

This will imply adding any d (mod q) to all colors results in a coloring satisfying the same constraints.



$q\mathrm{CSG}_\Delta$

$q\mathrm{CSG}_\Delta$ Instance:

- $U = (V, E, \Sigma, \phi)$
  $\phi: E \to P[\Sigma^2]$ is define by
  $\Delta: E \to P\{[q]\}$ thus
  $\phi(u, v) = \{(i,j) \mid i-j \bmod q \in \Delta(u, v)\}$

$P\{[q]\}$ = all subsets of $\{0..q-1\}$

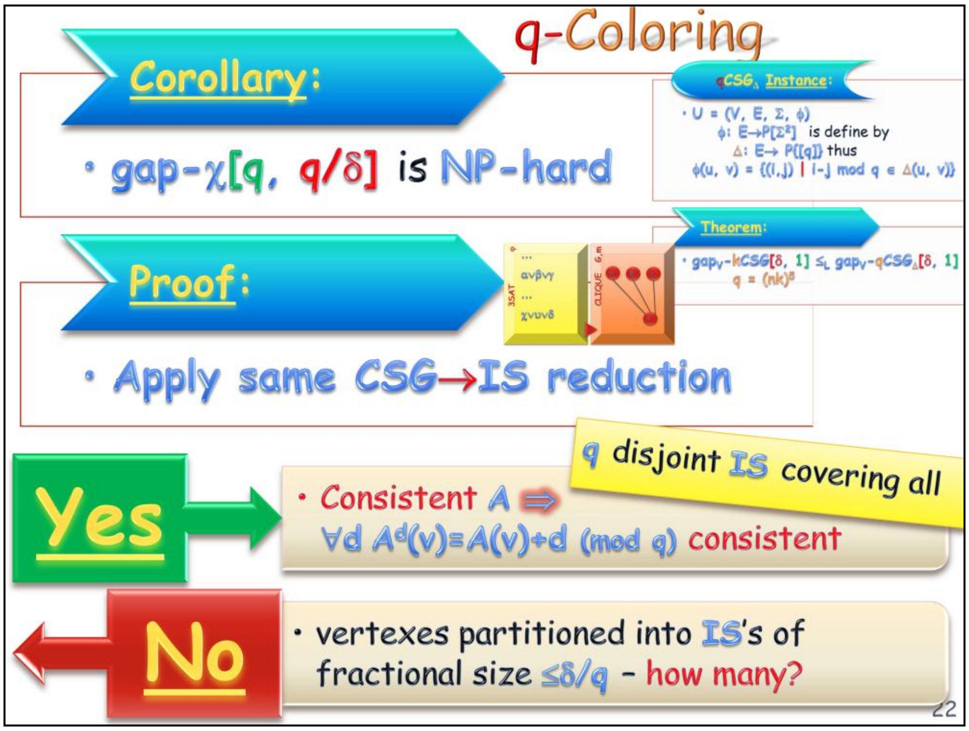
Theorem:

- $gap_V\text{-}k\mathrm{CSG}[\delta, 1] \leq_L gap_V\text{-}q\mathrm{CSG}_\Delta[\delta, 1]$
  $q = (nk)^5$

21

Here is a definition of the special form of CSG, namely, $q\mathrm{CSG}_\Delta$:
Colors are 0…q-1 and for each edge Δ specifies satisfying differences between colors mod q.

We will next prove one can reduce any CSG to such a CSG (maintaining the set of vertexes as is) while making the number of new colors be polynomial in the number of vertexes times the number of old colors.



q-Coloring

Corollary:

- gap-$\chi[q, q/\delta]$ is NP-hard

Proof:

- Apply same CSG→IS reduction

Yes

- Consistent $A \Rightarrow$ $\forall d\ A^d(v) = A(v) + d \pmod q$ consistent

q disjoint IS covering all

No

- vertexes partitioned into IS's of fractional size $\leq \delta/q$ – how many?

qCSG$_\Delta$ Instance:

- $U = (V, E, \Sigma, \phi)$
  $\phi: E \to P[\Sigma^2]$ is define by
  $\Delta: E \to P\{[q]\}$ thus
  $\phi(u, v) = \{(i,j) \mid i-j \bmod q \in \Delta(u, v)\}$

Theorem:

- $\text{gap}_V\text{-}k\text{CSG}[\delta, 1] \leq_L \text{gap}_V\text{-}q\text{CSG}_\Delta[\delta, 1]$
  $q = (nk)^5$

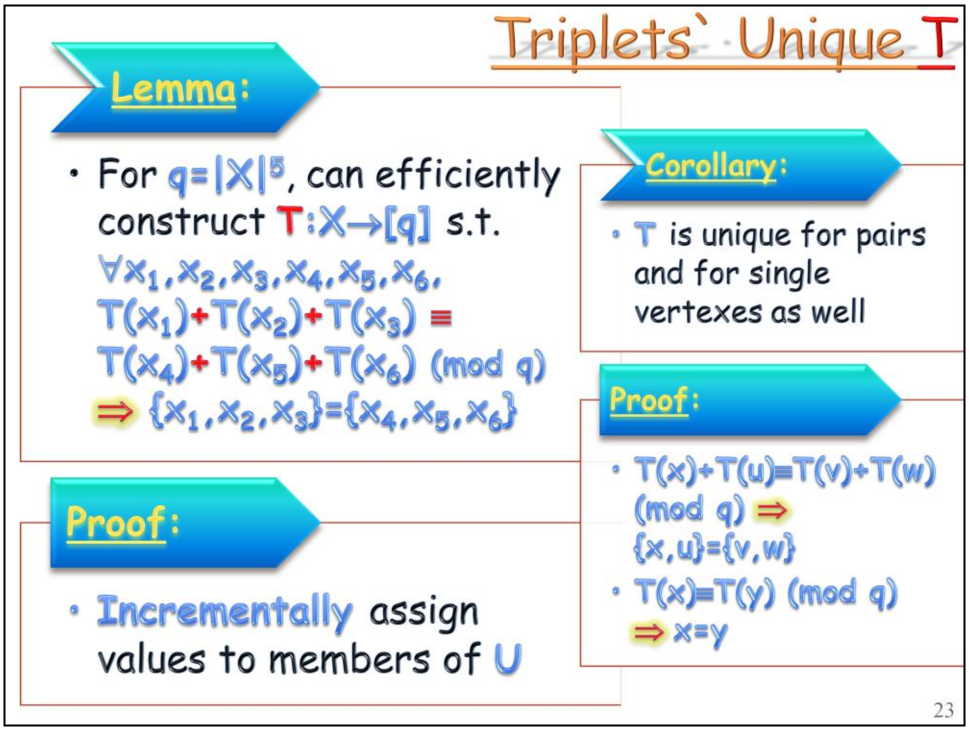
22

Before proving the theorem, let us note that as a corollary, the Chromatic Number of a graph is hard to approximate to within any constant. (and assuming stronger hardness results for approximating IS implies even stronger results --- can you see what would the factor be?).

To see why the corollary is true, let us consider the coloring number of the graph resulting from the CSG-to-IS reduction we studied earlier:

In case the CSG$_\Delta$ instance is completely satisfiable, look at all shifts of the coloring –--where one adds the shift value d (mod q) to the colors of all vertexes--- and observe these are all good colorings. Each of these, when translated to an IS in the IS-graph forms an IS; so that they are all pair-wise disjoint. Hence, their union covers all the graph, namely, colors it with q colors.

In case the CSG$_\Delta$ instance maximal good coloring colors only δ of the vertexes, the fractional size of the resulting IS is δ/q, hence a cover by IS's must consist of at least 1 over that in order to cover all vertexes. In that case, the chromatic number is at least q/δ.



Triplets` Unique T

Lemma:

- For $q=|X|^5$, can efficiently construct $T: X \rightarrow [q]$ s.t. $\forall x_1, x_2, x_3, x_4, x_5, x_6$, $T(x_1)+T(x_2)+T(x_3) \equiv T(x_4)+T(x_5)+T(x_6) \pmod{q} \Rightarrow \{x_1, x_2, x_3\} = \{x_4, x_5, x_6\}$

Proof:

- Incrementally assign values to members of U

Corollary:

- T is unique for pairs and for single vertexes as well

Proof:

- $T(x)+T(u) \equiv T(v)+T(w) \pmod{q} \Rightarrow \{x,u\}=\{v,w\}$
- $T(x) \equiv T(y) \pmod{q} \Rightarrow x=y$

23

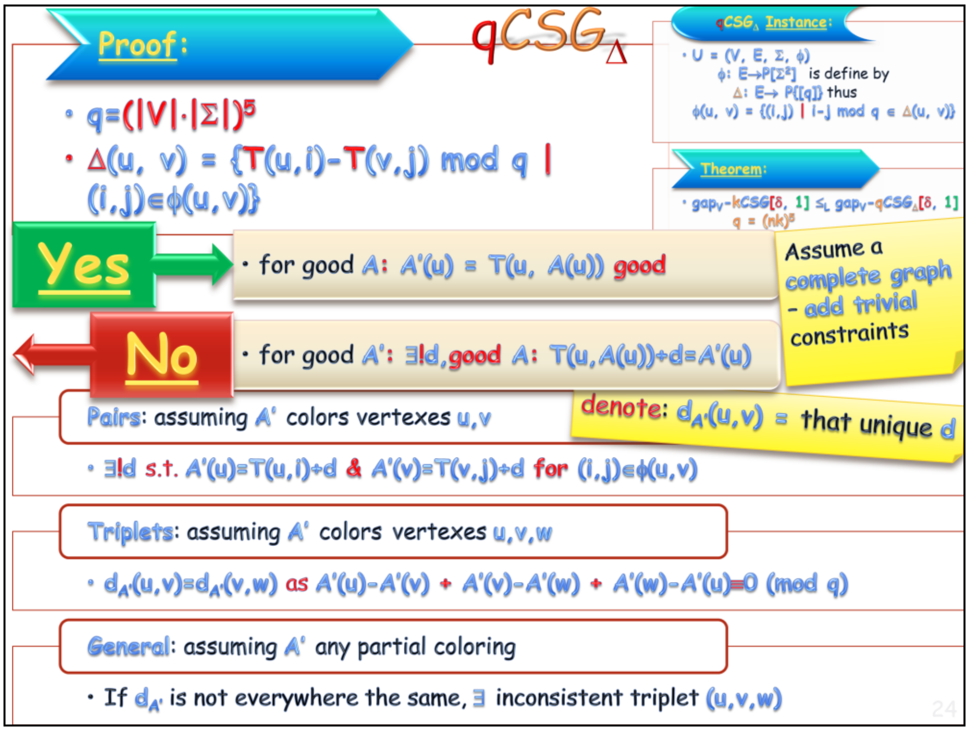
For the purpose of this reduction, let us introduce a mapping T, which assigns a number mod q to each element in some given set X, and so that the sum of values to each triplet clearly identifies that triplet (are unique):

Namely, take any three elements (with repetition) of X, apply the mapping T to them and add mod q --- the number you get is unique to that triplet.

One can incrementally and simplistically construct such a mapping, as long as q is at least the size of the set X to the power 5.

At each step the elements that are prohibited correspond to 5 elements whose values disallow that number. So, for such a q, there's always a number allowed.

Note that such a mapping is also unique for pairs (add the same elements to both pairs) and of course for single elements.



Proof:

$qCSG_\Delta$

- $q = (|V| \cdot |\Sigma|)^5$
- $\Delta(u, v) = \{T(u,i) - T(v,j) \bmod q \mid (i,j) \in \phi(u,v)\}$

$qCSG_\Delta$ Instance:

- $U = (V, E, \Sigma, \phi)$
  $\phi: E \to P[\Sigma^2]$ is define by
  $\Delta: E \to P\{[q]\}$ thus
  $\phi(u, v) = \{(i,j) \mid i-j \bmod q \in \Delta(u, v)\}$

Theorem:

- $gap_V\text{-}kCSG[\delta, 1] \le_L gap_V\text{-}qCSG_\Delta[\delta, 1]$
  $q = (nk)^5$

Yes
- for good A: $A'(u) = T(u, A(u))$ good

No
- for good A': $\exists! d$, good A: $T(u, A(u)) + d = A'(u)$

Assume a complete graph - add trivial constraints

denote: $d_{A'}(u,v)$ = that unique d

Pairs: assuming A' colors vertexes u,v

- $\exists! d$ s.t. $A'(u) = T(u,i) + d$ & $A'(v) = T(v,j) + d$ for $(i,j) \in \phi(u,v)$

Triplets: assuming A' colors vertexes u,v,w

- $d_{A'}(u,v) = d_{A'}(v,w)$ as $A'(u) - A'(v) + A'(v) - A'(w) + A'(w) - A'(u) \equiv 0 \pmod q$

General: assuming A' any partial coloring

- If $d_{A'}$ is not everywhere the same, $\exists$ inconsistent triplet (u,v,w)

24

Time to prove the Theorem, that is, show a reduction from a general CSG problem to one in which each constraint is derived by some set of allowed differences (CSG_Δ).

For starters, let us assume the graph is a complete graph, that is, there is a constraint between any two vertexes (otherwise, simply add a trivial constraint).

Now, set q to be the range necessary for the mapping T so as to map all pairs (v, i) for any vertex v and color i ($X=V\times\Sigma$). For any pair u,v let the allowed differences be all T(u, i)-T(v, j) where u,i and v,j are consistent.

This is the reduction --- let's proceed to the proof of correctness.

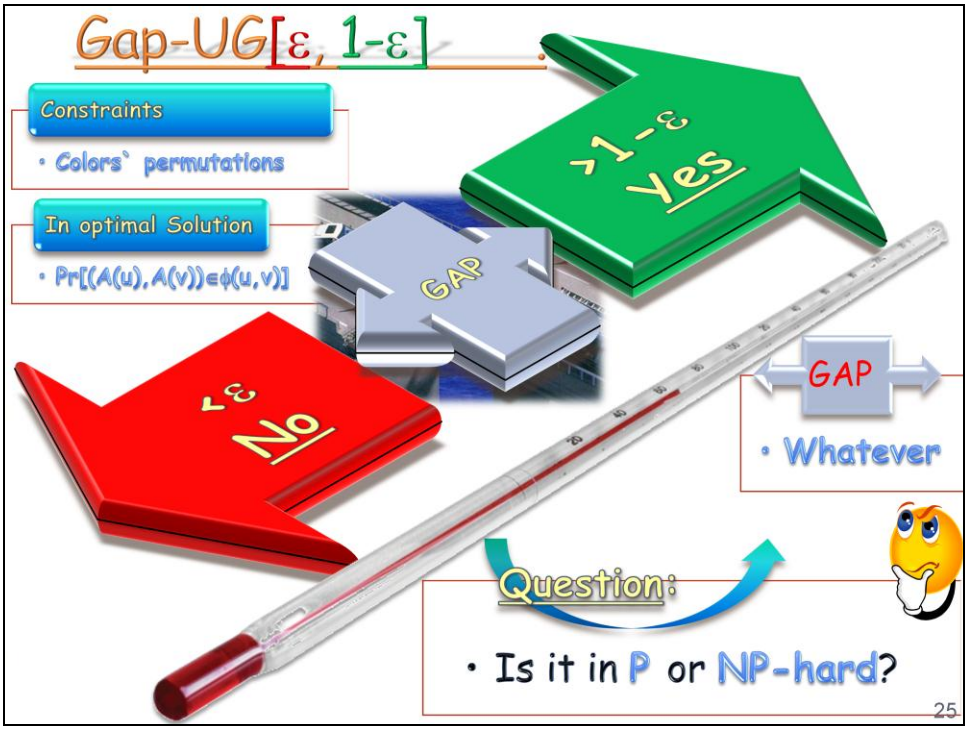
Completeness is rather easy – given a coloring A to the original graph, let the coloring for the constructed graph A' color each vertex v by T(v, A(v)).

As to **soundness**, let us prove, given an assignment A' to the constructed graph, that there is a global shift d, so that if one subtracts d from all colorings of A', and then applies T^{-1}, one gets a coloring of the original graph --- every coloring has an origin.

2: Assign a shift $d_{A'}(u,v)$ to every pair u,v so that both u and v are colored by A'; then show these shifts are all the same. This shift is well defined (there is exactly one such shift) as otherwise T is not unique for pairs.

3: Now look at triplets u,v,w all of which are colored by A'. The shift for u.v must be the same as the shift for v,w --- otherwise T would not be unique for triplets (the sum of the three difference is clearly 0 – every element has one positive and one negative occurrences).

∞: Finally, for a general set of colored vertexes, if the shifts are not everywhere consistent, there must be a vertex v, so that the shift for u,v and the shift for v,w are not consistent --- that's an inconsistent triplet, which cannot exist as we just proved.



Gap-UG[ε, 1-ε]

Constraints

- Colors` permutations

In optimal Solution

- Pr[(A(u),A(v))∈φ(u,v)]

>1-ε
Yes
GAP
<ε
No

GAP

- Whatever

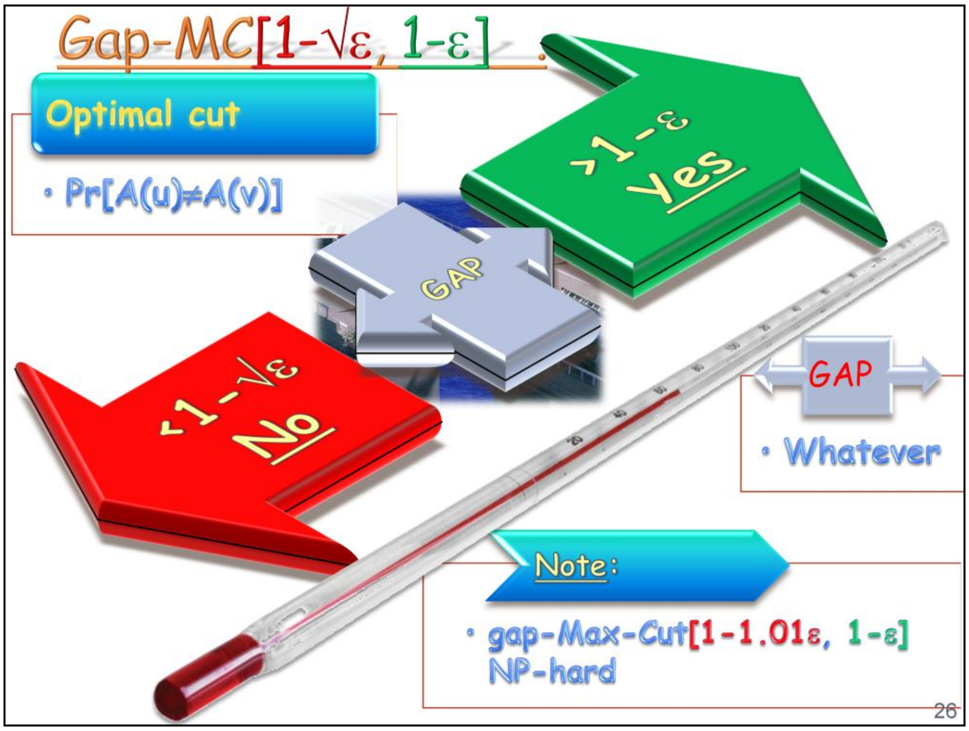
Question:

- Is it in P or NP-hard?

25

Let us introduce the UG problems, where constraint are 1-to-1 mapping between colors.

- Not known to be solvable in poly-time
- Neither known to be NP-hard
- In fact, not know to be as hard as any other



Gap-MC[1-√ε, 1-ε]

Optimal cut

- Pr[A(u)≠A(v)]

>1-ε
Yes
GAP
<1-√ε
No

GAP

- Whatever

Note:

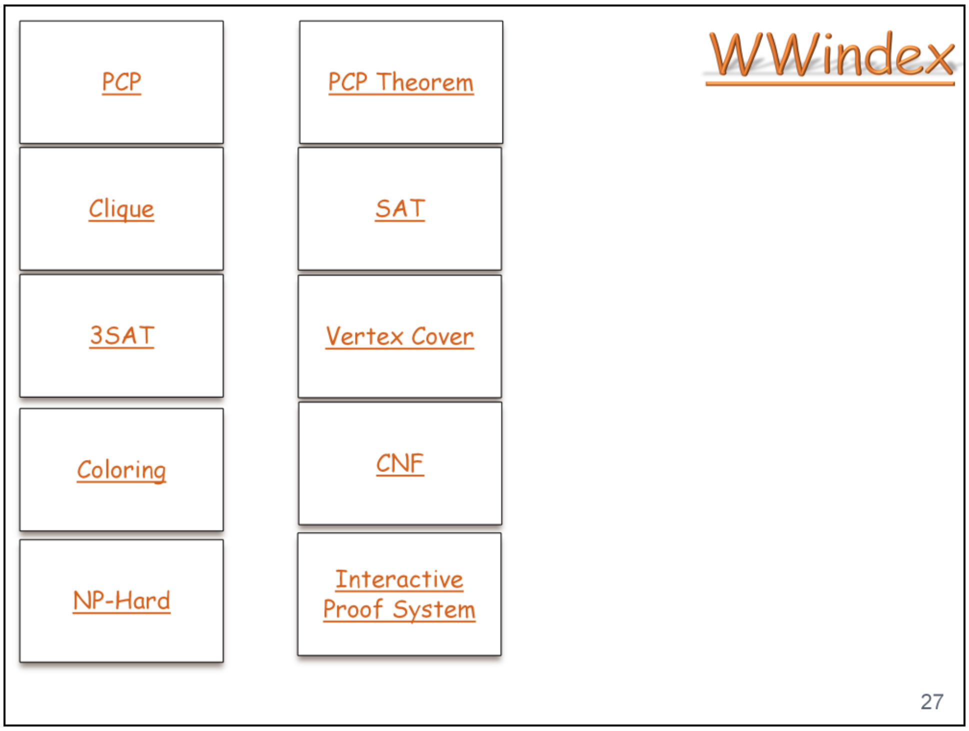
- gap-Max-Cut[1-1.01ε, 1-ε] NP-hard

26

How about MaxCut?

The best one can achieve ---applying Semi-Definite-Programming---
is 1-sqrt(epsilon) assuming a cur of fractional size 1-epsilon.

NP-hardness is know only for a much stronger approximation ratio.
UG-hardness?



WWindex

- PCP
- Clique
- 3SAT
- Coloring
- NP-Hard
- PCP Theorem
- SAT
- Vertex Cover
- CNF
- Interactive Proof System