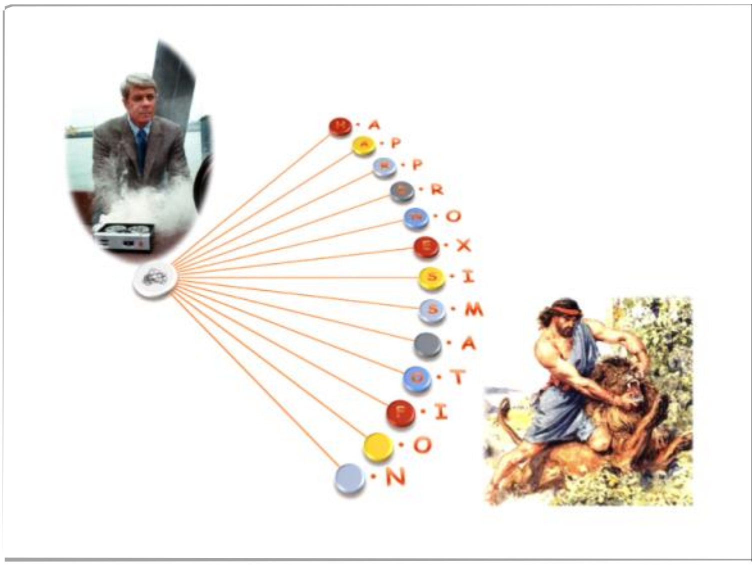
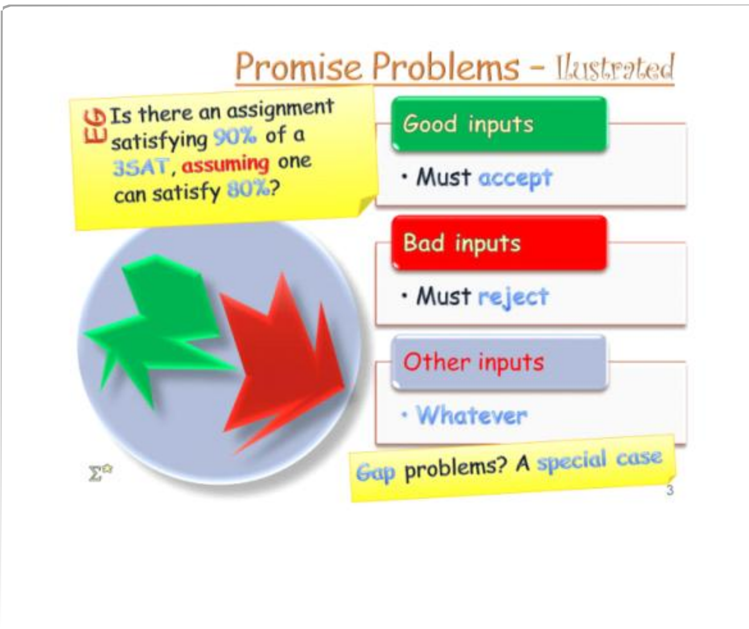
PCP Notes



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

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

PCP Theorem

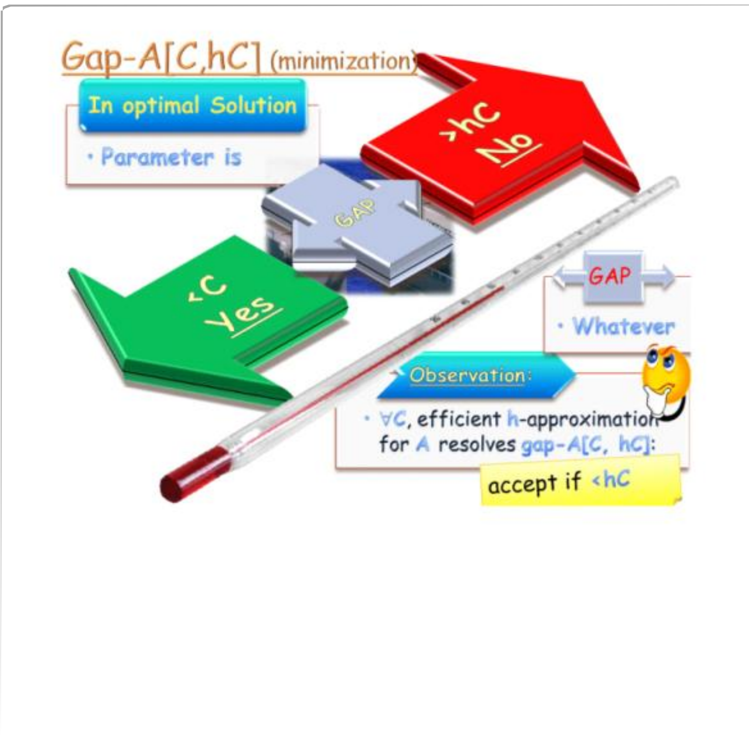


Let us start with a general definition of *promise problems*: these are problems in which the algorithm is required to accept some inputs and reject others, however, unlike in algorithm for languages, some inputs are "don't care", and the algorithm may return whatever answer.

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.

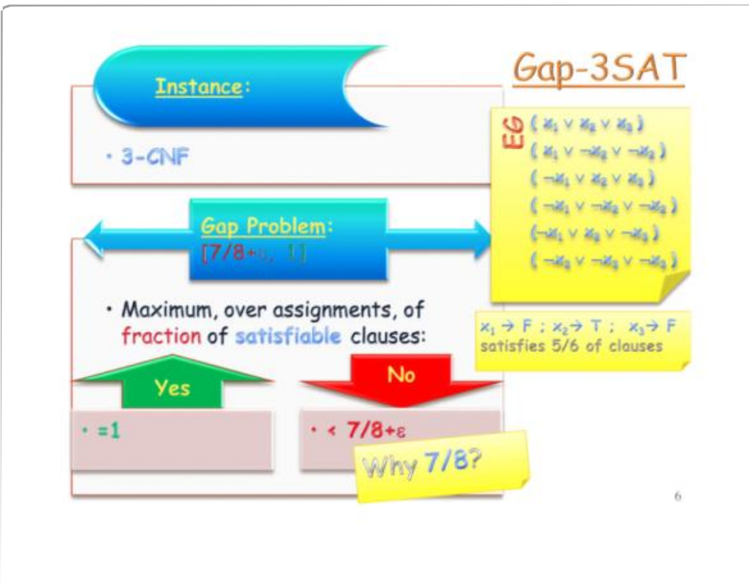
Let us recall the general framework of *optimization* problems --- either *minimization* or *maximization* problems. These problems allow some type of solution and measures those solutions according to the *optimization parameter*.
The goal is to find the *best* solution *according* to the *parameter* --- 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 problems partitions the input instances into 3 categories according to the optimized parameter of their 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* threshold 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.

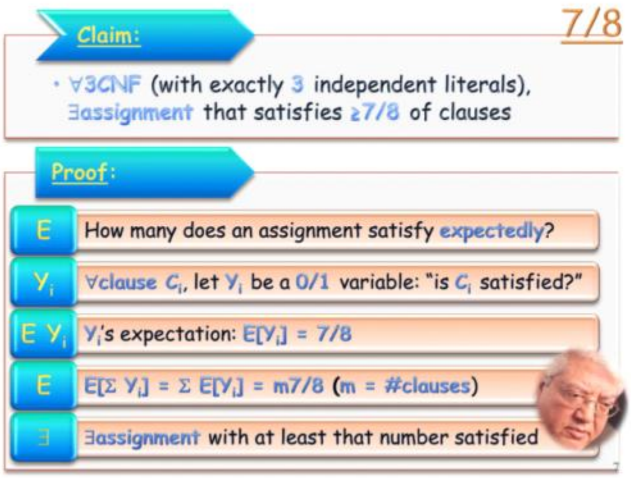


As much as 3SAT is the grandparent of all NP-complete problems, the gap 3SAT problem can serve the same role for gap problems.

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+ε fraction of the clauses can be satisfied by any given assignment.

We will next see why we chose the 7/8 fraction.

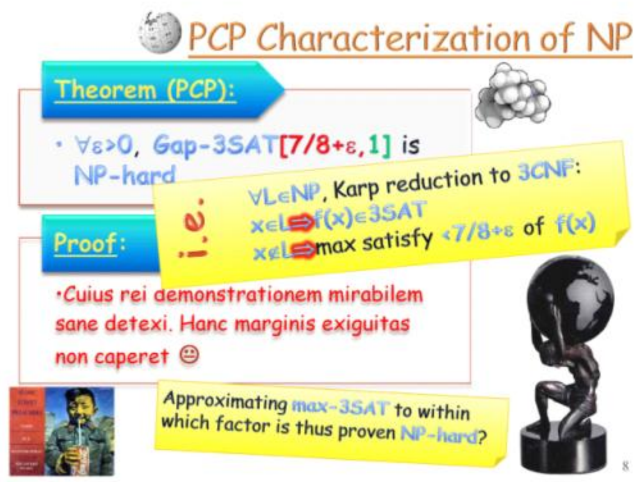


In general, any 3SAT instance that has clauses with exactly 3 independent literals, has an assignment that *satisfies 7/8* of the clauses.

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.

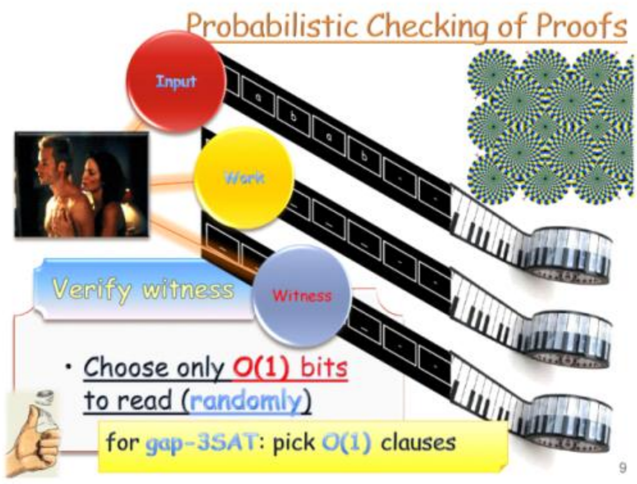
To see that, assume a variable Yi that is 1 if clause Ci is satisfied and 0 otherwise.
For any i, the average value of Yi is 7/8, as 7 of all 8 possible assignments to its 3 variables satisfy it.
Now look at the average number of clauses satisfied: it is the same as the sum of averages (by linearity 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.



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 is a completely satisfiable formula (as in Cook/Levin theorem). 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*.

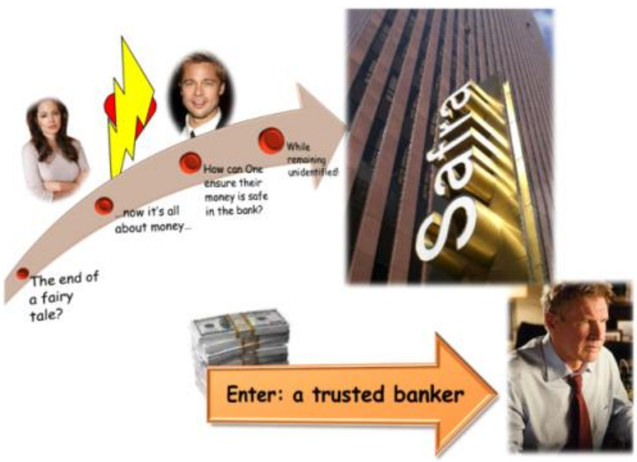
This can be viewed as an alternative, much stronger characterization of NP than the one of Cook/Levin.

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.



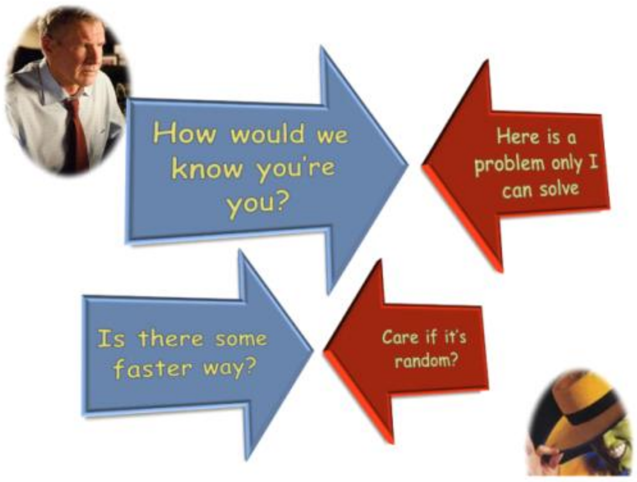
Recall the characterization of NP we used before, namely as all languages for which a witness of membership can be verified efficiently.
Now consider a very limited verifier for the membership proof --- one who is only allowed to *read a constant number of bits of the witness proof*.
Nevertheless, it 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.

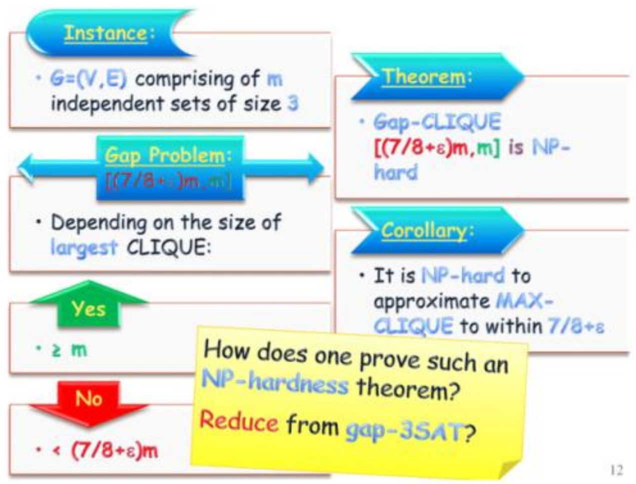


Here's a possible scenario which can paint a nice picture of the subject matter.

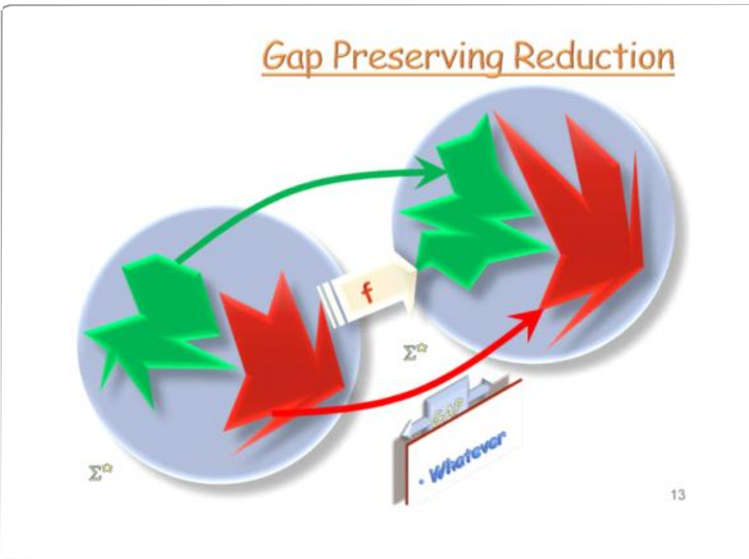
Suppose someone wants to deposit some money in a bank account, without being identified.



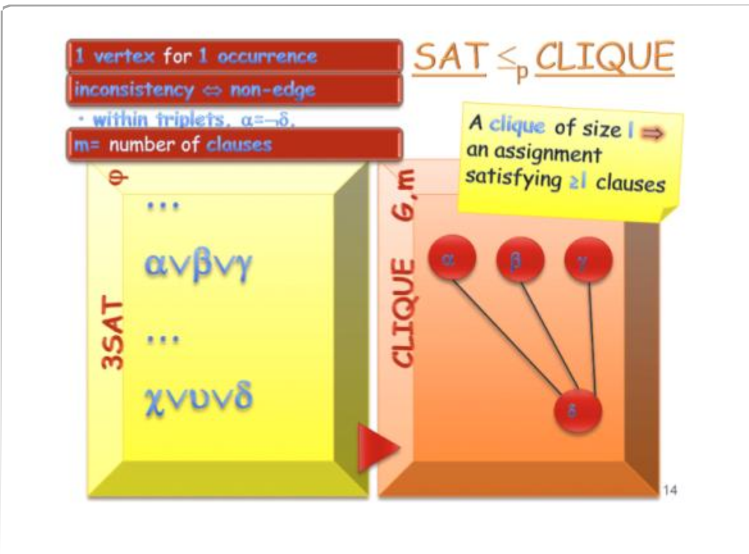
Associated with the bank account there will be a secret known only to the owner of the account (whose responsibility is not to reveal it to anyone unauthorized).
The natural candidate for such a secret is an input to a hard computational problem, which is designed so as for the account owner to know the solution for.
The owner, wanting to carry out some transaction, would prove to the bank clerk she/he knows the solution for the specific input.
Now, to make the process much faster, we can assume the solution is encoded so that the bank clerk can read only very few bits and thus verify the correctness of the solution, albeit, with some negligible probability of error, which can be made arbitrarily small.



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 from gap-3SAT to approximation of Max-CLIQUE.
As a corollary max-CLIQUE is NP-hard to approximate to within the corresponding factor.

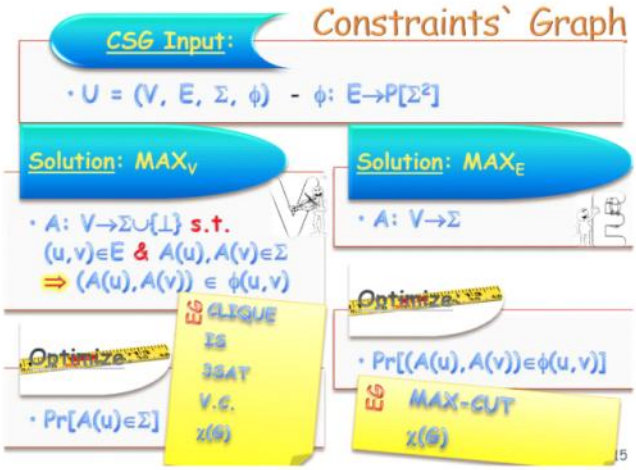


A gap-preserving Karp-reduction, from one gap-problem to another, is one that takes good inputs to good inputs, bad inputs to bad one, and takes "don't care" inputs to whatever inputs it happens to.



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

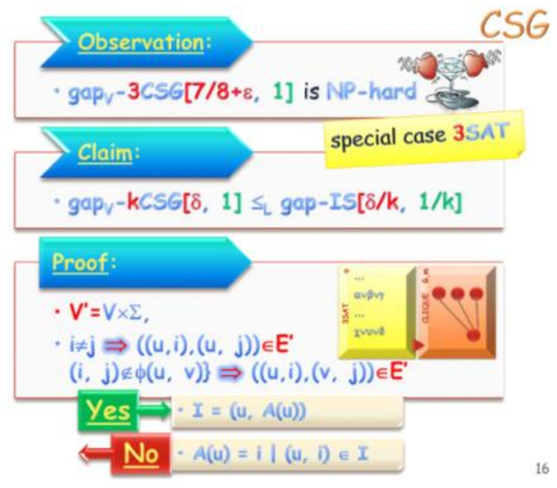
Completeness is clearly the same (it is exactly the same statement).
As to *soundness*: note that a clique of l vertexes can be utilized to construct an assignment, by making TRUE all literals appearing in it --- this must satisfy all clauses the clique has a representative in. Hence, the assignment satisfies l clauses.



Let us now generalize some optimization problems we have discussed earlier, by introducing the *constraints graph (CSG)* 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 at its ends.
There are two possible variants of the problem:
In the first, we allow vertexes to remain uncolored, however require all constraints between colored vertexes be 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 fall under this general definitions.
See if you can identify for those where is the relevant gap for which the problem is easy, and where it may be hard.



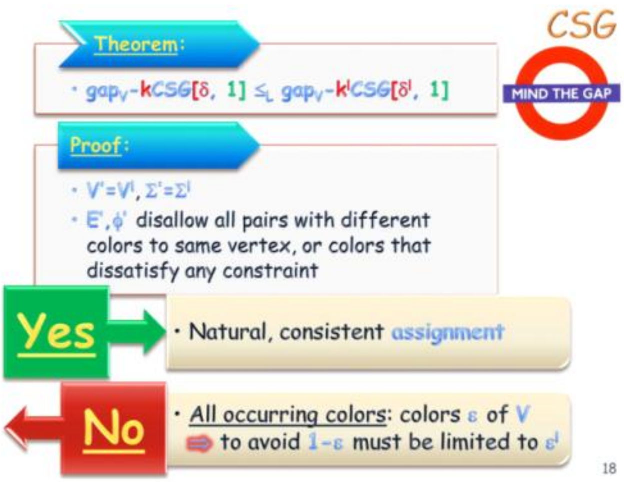
When looking at gap-CSG, even when the alphabet (colors) set is of size 3, for the appropriate gap, the problem is NP-hard, which is the case as 3SAT can be reduced to it.
In fact, we have just established NP-hardness of gap-V-CSG above.

Another important observation is that any CSG problem can be directly reduced to a Independent Set (or CLIQUE) problem as follows:
Have a vertex for each pair of a vertex and color of the CSG instance, and incorporate the appropriate edges (are these pairs of vertex/color consistent?).

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

Here is an obvious problem with our picturesque motivating story:
How can we prevent the bank clerk from going to another branch and solving our problem there to an unsuspecting other clerk?

Well, can we at least design a protocol in which the account owner reveals only a small fraction of the colors to the vertexes of the constraints graph?

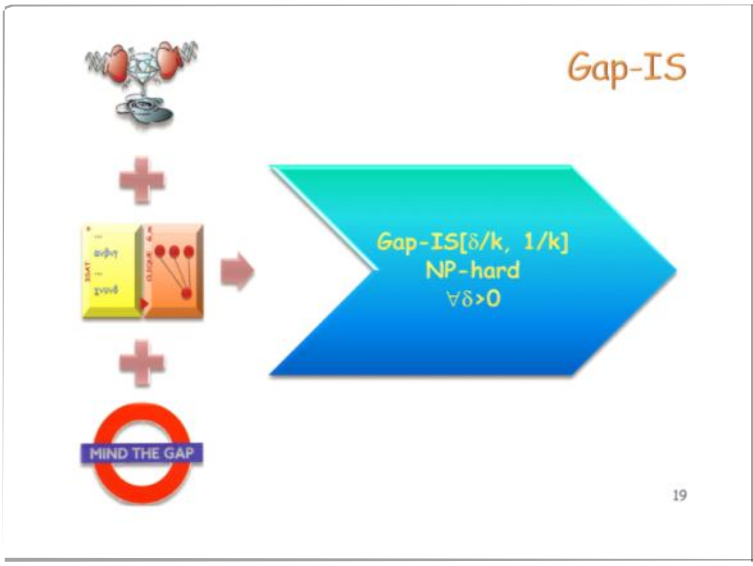


It turns out we can in general "amplify" any CSG problem. The reduction goes as follows:

Given a CSG instance, construct a new instance in which

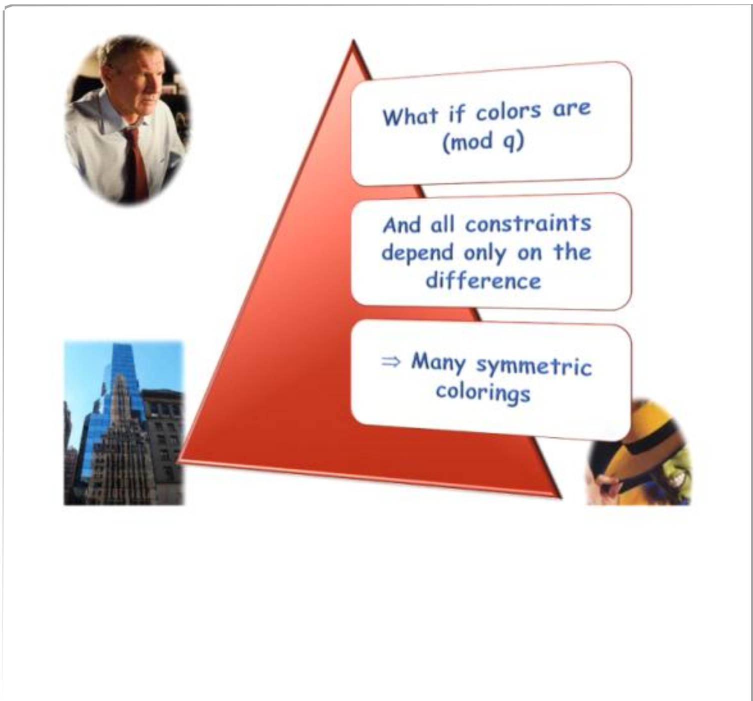
- Every vertex corresponds to a sequence of l vertexes of the original graph.
- The new set of colors will assign a vertex with coloring for all l original vertexes.
- The new constraints consist of any inconsistency in coloring that may occur. Namely, if two new vertexes are colored so that the same 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.

Let us now consider the first variant, where one tries to maximize the fraction of vertexes colored.
Given a consistent coloring of the original graph, one can naturally extend it to the new graph.
Now, suppose you have a coloring of the new graph. Take all colorings of the original graph (appearing in any vertex corresponding to an l-sequence containing it). These colors are everywhere consistent.
If they happen to color an ε fraction of the original vertexes, the coloring of the new graph must avoid any l-sequence containing any of the 1-ε fraction of uncolored vertexes, which implies it must be of size at most ε^l.



The three last claims we prove, combined, implies that it is hard to approximate the IS (or CLIQUE) size to within any constant factor.

Make sure you see how come.

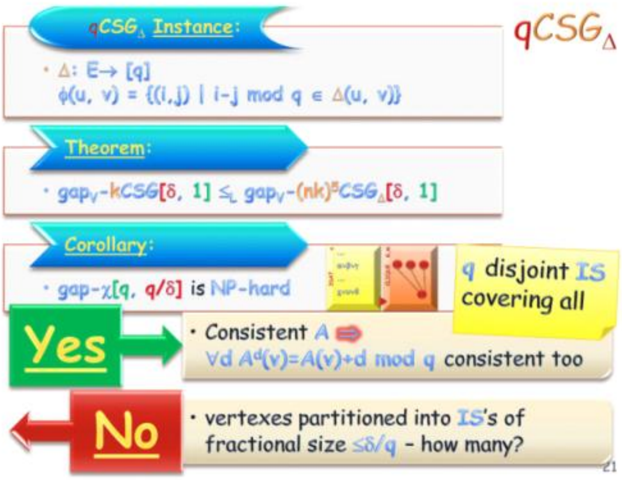


Does this prohibits the bank clerk from posing as the account owner?

Not really! Despite the fraction being arbitrarily small, the amount of information may be enough to reconstruct the entire coloring…

So, can 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.

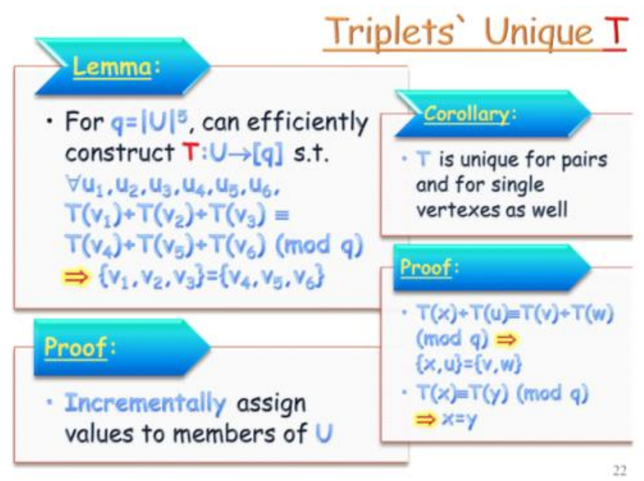


Here is a definition of the $qCSG_{\Delta}$.

Colors are 0…q-1 and for each edge Δ defines satisfying differences between colors (mod q).

We will next prove one can reduce any CSG to such a CSG while making the number of new colors polynomial in the number of vertexes times the number of old colors.

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. (Assuming IS is hard to approximate to within a constant power of the number of vertexes, this would imply an even stronger result --- can you see what would the factor be?).
To see why this 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_{Δ} 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 coloring as well. 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_{Δ} instance maximal good coloring colors only a δ fraction of the vertexes, a cover by IS's must consist of at least $1/\delta$ IS's in order to cover all vertexes. In that case, the chromatic number of the resulting graph is at least q/δ.



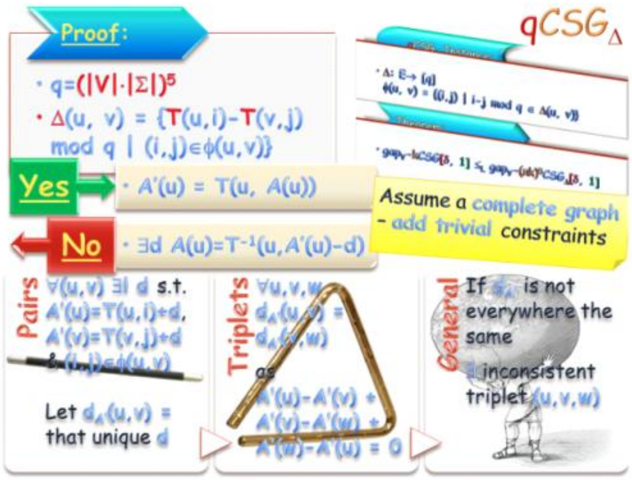
For the purpose of this reduction, let us introduce a mapping, assigning each element in some set U with a number mod q, and so that the sum of triplets are unique.

Namely, take any three elements (with repetition) of U, 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 $|U|^5$.

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 (mod q) allowed.

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



Now it's 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)

Before we start with the reduction let us assume the graph is a complete graph, that is, there is a constraint between any two vertexes. If two vertexes have no constraint 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 ($U=V\times\Sigma$).

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)).

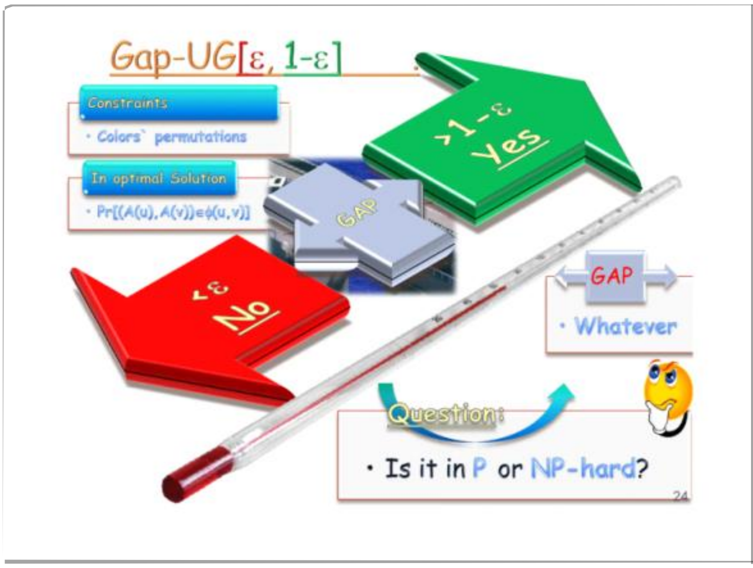
As to soundness, let us prove –given an assignment A' to the new, constructed graph– that there is a global shift d, so that if one subtract d from all colorings of A', and then inverse T, one gets a coloring of the original graph.

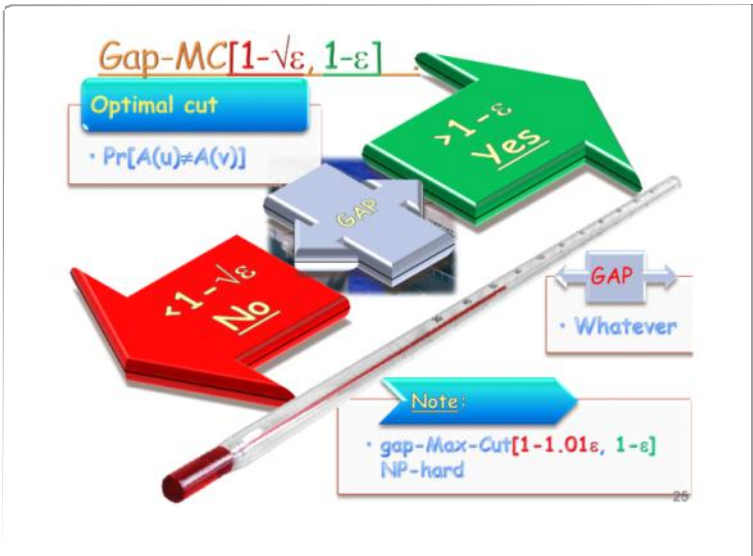
Let us first assign a shift $d_{A'}(u,v)$ to every pair u,v so that both u and v are colored by A' – then show these shifts are all the same.

The shift is how much one should subtract from both colors so as to get it to values of T. This shift is well defined (there is exactly one such shift) as otherwise T is not unique for pairs.

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. But 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?
24


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
25

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