Problem Set 9 — Module 9

Shenyu Li

NOV 28th 2024

8.4 CLIQUE-3 Problem is in NP

i need proving that the CLIQUE-3 problem, where every vertex in the graph has a degree at most 3, is in NP. The CLIQUE-3 problem is a restricted version of the general CLIQUE problem.

Step 1: Definition of CLIQUE-3

The CLIQUE-3 problem asks if there exists a clique (a complete subgraph) of size k in a given graph, where every vertex has a degree at most 3.

Step 2: Verifying a Solution

To prove that the problem is in NP, we need to show that given a solution (i.e., a set of vertices that form a clique of size k), we can verify in polynomial time whether it is indeed a valid clique. The verification process involves:

1. Checking that the subgraph induced by the set of vertices is a complete subgraph, i.e., every pair of vertices in the subset has an edge between them.
2. Ensuring that the degree of each vertex in the clique does not exceed 3.

Both of these checks can be performed in polynomial time. Specifically, verifying the complete subgraph requires checking all pairs of vertices for an edge, which takes $O(k^2)$ time, where k is the size of the proposed clique. Checking that the degree of each vertex does not exceed 3 can be done by inspecting each vertex's neighbors, which is also a polynomial-time operation.

afrer all

Since the solution can be verified in polynomial time, the CLIQUE-3 problem is in NP.

8.5 Reduction from 3D MATCHING to SAT

The 3D MATCHING problem is defined as follows: given three sets X, Y, and Z, and a set of 3-element tuples, each containing one element from each of the sets, the task is to select a subset of these tuples such that every element in X, Y, and Z is included in exactly one tuple.

reduce the 3D MATCHING problem to a Boolean satisfiability (SAT) problem as follows:

Step 1: Variables for Tuples

For each tuple $t_i = (x_i, y_i, z_i)$, define a Boolean variable x_i, which will be assigned the value 1 if the tuple t_i is selected in the matching and 0 otherwise. This is the basic structure for representing the problem as a SAT problem.

Step 2: Conflict Constraints

Since choosing a matching from a set of tuples, ensure that no two tuples share an element. That is, for any two tuples $t_i = (x_i, y_i, z_i)$ and $t_j = (x_j, y_j, z_j)$, they cannot both be selected if they contain the same element in any of the sets X, Y, or Z. For example, if t_i and t_j both contain $x_i = x_j$, ensure that at least one of them is not selected.

For this, the following conflict constraints:

$$\neg x_i \lor \neg x_j$$

for each pair of conflicting tuples t_i and t_j that share an element in X, Y, or Z. This ensures that if one tuple is selected, the other cannot be.

Step 3: Coverage Constraints

Next, ensure that every element in the sets X, Y, and Z appears in exactly one selected tuple.

For each element in X, Y, and Z, define clauses that express the constraint that each element is covered by exactly one tuple. For example:

- For each element $x_i \in X$, create a clause that ensures that at least one tuple containing x_i is selected:

$$x_{i1} \lor x_{i2} \lor \cdots \lor x_{in}$$

  where $x_{i1}, x_{i2}, \ldots$ represent the variables corresponding to the tuples containing x_i.

- Similarly, for each element $y_i \in Y$, and each element $z_i \in Z$, create corresponding clauses ensuring that each element is included in exactly one tuple.

These clauses ensure that all elements in X, Y, and Z are covered by the selected tuples.

Step 4: Final SAT Formula

Finally, combine all the conflict and coverage clauses into a single SAT formula. The SAT formula is composed of a conjunction of clauses:

$$\phi = \bigwedge_{\text{all tuples}} (\neg x_i \vee \neg x_j) \wedge \bigwedge_{\text{all elements in X, Y, Z}} (x_{i1} \vee x_{i2} \vee \cdots \vee x_{in})$$

This formula ensures that the selected tuples form a valid matching, with no element being repeated across tuples, and every element from X, Y, and Z is covered exactly once.

Step 5: Polynomial Time Reduction

The reduction from 3D MATCHING to SAT involves constructing a SAT formula with one Boolean variable for each tuple, and clauses to enforce the conflict and coverage constraints. Since the number of variables and clauses in the resulting SAT formula is linear in the size of the input (i.e., the number of tuples), the reduction can be performed in polynomial time.

Thus, the 3D MATCHING problem is polynomial-time reducible to SAT.

8.8 EXACT 4SAT Problem:

Given a Boolean formula in CNF where each clause contains exactly 4 literals and each variable appears at most once in each clause, determine whether there exists an assignment of truth values to the variables such that the formula is satisfied.

The goal is to show that **EXACT 4SAT** is NP-complete by reducing the 3SAT problem to it.

Reduction from 3SAT to EXACT 4SAT

start with a given 3SAT instance and show how to transform it into an EXACT 4SAT instance. The steps are as follows:

Step 1: Remove duplicate literals

In the 3SAT problem, if a clause contains the same literal more than once (for example, $x \vee x \vee y$), simply remove the duplicates. This does not affect the satisfiability of the formula because having duplicate literals in a clause does not change the truth value of that clause.

For instance, the clause $(x \vee x \vee y)$ simplifies to $(x \vee y)$.

Step 2: Eliminate contradictory literals (both a literal and its negation in the same clause)

If a clause contains both a literal and its negation (e.g., $x \vee \neg x \vee y$), this clause is trivially satisfied regardless of the assignment, as either x or $\neg x$ will always be true. thus remove any clauses where both a literal and its negation appear, and simplify the clause accordingly.

For example, the clause $(x \vee \neg x \vee y)$ simplifies to (y).

Step 3: Add auxiliary variables to ensure exactly 4 literals per clause

Now, ensure that each clause contains exactly 4 literals. In the 3SAT instance, each clause contains 3 literals, so introduce **dummy variables** (also known as auxiliary variables) to add one extra literal to each clause. These dummy variables do not affect the satisfiability of the clause because they can be assigned arbitrary truth values.

For instance, if a clause in the 3SAT formula is $(x \vee y \vee z)$, introduce a new dummy variable w and form the new clause $(x \vee y \vee z \vee w)$. This new clause still remains satisfiable under the same assignment for x, y, and z, and the assignment of w can be chosen freely.

Step 4: Ensure that each variable appears at most once per clause

In EXACT 4SAT, each variable can appear at most once per clause. If any variable appears multiple times in a clause in the 3SAT instance, resolve this by introducing additional dummy variables. For example, if a clause is $(x \vee x \vee y)$, rewrite it as $(x \vee y \vee w)$, where w is a new dummy variable. This ensures that each variable appears only once in the clause, as required by the EXACT 4SAT problem.

Step 5: Resulting formula is an EXACT 4SAT instance

After performing these steps, transformed the 3SAT instance into an EXACT 4SAT instance. The resulting formula has the following properties:

- Each clause contains exactly 4 literals.
- Each variable appears at most once in each clause.
- The satisfiability of the formula is preserved.

Thus, reduced the 3SAT problem to the EXACT 4SAT problem, and since 3SAT is NP-complete, this shows that EXACT 4SAT is also NP-complete.