

Lecture 20

Lecturer: Andre Wibisono

Scribe: John Lazarsfeld

1 Overview

This lecture continues our introduction on complexity and the concept of polynomial time reductions. We mostly follow Section 8.2 in the KT textbook.

1.1 Recap From Lecture 19

In the previous lecture, we defined the notion of a **reduction** between two problems X and Y . In particular, we say that Y can be *reduced in polynomial time* to X , denoted by $Y \leq_p X$ if (informally):

- (i) We can reduce (transform) an instance of Y into an instance of X in polynomial time.
- (ii) An algorithm that solves instances of X can be used (a polynomial number of times) to solve instances of Y .

Recall that using this definition of reducibility allows us to characterize the relative “hardness” of two problems. For example, if $Y \leq_p X$, then we consider X to be “at least as hard as” Y , since any computational lower bounds on Y will also apply to X . In the other direction, if we can solve X efficiently, then $Y \leq_p X$ also implies that we can solve Y efficiently.

In previous lectures we’ve seen several examples of these reductions:

- Bipartite-Matching \leq_p Max-Flow
- Independent-Set \leq_p Vertex-Cover
- Vertex-Cover \leq_p Independent-Set
- Vertex-Cover \leq_p Set-Cover

Note also that these polynomial reductions are *transitive* in the following sense:

Lemma 1. *If $Z \leq_p Y$ and $Y \leq_p X$, then $Z \leq_p X$.*

The proof of this statement follows from composing algorithms for X and Y : we can solve an instance of Z using an algorithm for Y (since $Z \leq_p Y$), and we can simulate an algorithm for Y via an algorithm for X (since $Y \leq_p X$).

1.2 Today's Focus and Motivation

With the transitivity property of Lemma 1 in mind, today we will introduce a new family of problems involving Boolean formulae known as *satisfiability* problems. We will work toward showing that 3-SAT (a special type of satisfiability problem) reduces to Independent-Set, i.e.:

$$3\text{-SAT} \leq_p \text{Ind-Set} .$$

The main consequence of this new reduction is that, given the transitivity lemma and the list of reductions established in previous lectures:

$$3\text{-SAT} \leq_p \text{Ind-Set} \leq_p \text{Vertex-Cover} \leq_p \text{Set-Cover} .$$

This highlights the generality of 3-SAT (and Boolean satisfiability problems in general) for modeling other types of problems, including graph and set-based problems. Thus understanding the properties of these satisfiability problems is an essential component of complexity theory.

We will proceed by introducing the Satisfiability problem (and its variants 3-SAT and 2-SAT), and then we will show that $3\text{-SAT} \leq_p \text{Ind-Set}$.

2 Satisfiability

We begin by introducing the components of a Boolean formula. First, suppose we have a set of n Boolean variables

$$x_1, x_2, \dots, x_n \in \{0, 1\} .$$

A **term** (or **literal**) t is one of the variables x_i or its negation \bar{x}_i . A **clause** C is a disjunction (a logical OR) of distinct terms

$$C = t_1 \vee t_2 \vee \dots \vee t_\ell .$$

The *length* of C is defined as the number of terms in C . For each term t_i , we have $t_i \in \{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n\}$. A Boolean **formula** f in **Conjunctive Normal Form (CNF)** is a conjunction (a logical AND) of clauses

$$f = C_1 \wedge C_2 \wedge \dots \wedge C_k .$$

The length of f is defined as the number of clauses k .

Given a Boolean formula f in CNF over a set of variables x_1, \dots, x_n , a natural question to ask is whether f has a **satisfying assignment**? This is an assignment of 0 or 1 to each x_i such that the formula f evaluates to True.

Since f is a formula in CNF (an AND of clauses), a satisfying assignment must ensure *all* clauses C_j evaluate to True. Since each clause C_j is an OR of terms, this is equivalent to ensuring

at least one term t_i in each clause C_j evaluates to True. This yields the definition of the Satisfiability problem (SAT):

SAT – Input: CNF formula $f = C_1 \wedge \cdots \wedge C_k$
 Output: Yes iff there exists a satisfying assignment to f .

Note that each clause C_i in f can contain any number of terms. However, we can define analogous problems where the structure of f is more constrained:

3SAT – Input: CNF formula $f = C_1 \wedge \cdots \wedge C_k$ where each C_i has length 3
 Output: Yes iff there exists a satisfying assignment to f .

Similarly:

2SAT – Input: CNF formula $f = C_1 \wedge \cdots \wedge C_k$ where each C_i has length 2
 Output: Yes iff there exists a satisfying assignment to f .

Consider the following examples of Boolean formulae:

(1) $f = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_2 \vee \overline{x_3})$.

The assignment $\{x_1 = 0, x_2 = 0, x_3 = 0\}$ satisfies f .

(2) $f = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2}) \wedge (x_2 \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_1}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$

Here, you can check that f has no satisfying assignment.

Note that given an assignment of variables $X \in \{0, 1\}^n$ (where $X_i \in \{0, 1\}$ is the assignment of the variable x_i), it is easy to verify whether f evaluates to True. However, in the SAT, 3-SAT, and 2-SAT problems, our goal is to *find* a satisfying assignment X . This is a combinatorial search problem, where the space of all possible assignments $\{0, 1\}^n$ has exponential size. Additionally, although the notation here is abstract, note that a Boolean formula can describe many “real-world” constraint satisfaction problems.

3 Reducing 3-SAT to Independent-Set

Recall that given a graph $G = (V, E)$, an independent set S of size k is a subset of k vertices such that no two vertices share an edge. We now develop the proof of the following lemma:

Lemma 2. *3-SAT \leq_p Independent-Set.*

We will assume that we have an algorithm to solve Independent set, and we start by describing a procedure to convert instances of 3-SAT into instances of Independent-Set:

- (1) Assume we are given an instance of 3-SAT $f = C_1 \wedge \dots \wedge C_k$ over the variables $\{x_1, \dots, x_n\}$.
- (2) Construct a graph $G = (V, E)$ consisting of $3k$ vertices grouped into k triangles, where each triangle corresponds to 1 clause.

Specifically, for $i = 1, \dots, k$, construct three vertices v_{i1}, v_{i2}, v_{i3} , where v_{ij} corresponds to the j 'th term from clause C_i .

For each $i = 1, \dots, k$, add edges between all v_{ij} . Additionally, for each pair of vertices that correspond to terms (in different clauses) that are negations of each other (i.e., $\overline{x_1}$ and x_1), add an edge between the two vertices.

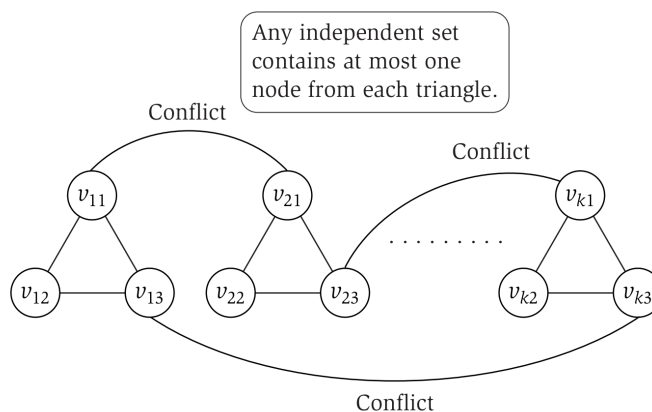


Figure 1: Example graph from 3-SAT to Independent-Set reduction. (Source: KT Figure 8.3).

Figure 1 gives an example of such a construction, and we now proceed to prove the following claim:

Claim 1. *The instance of 3-SAT is satisfiable if and only if the graph G has an independent set of size at least k .*

As usual, we prove each direction of the claim separately:

(\Rightarrow) *If the 3-SAT instance is satisfiable, then G has an independent set of size at least k .*

Assume that the instance of 3-SAT is satisfiable, meaning there exists some assignment of the variables x_i satisfying f . Then (by definition of the construction) each triangle of G contains at least 1 vertex whose corresponding term in f evaluates to 1 under this assignment. Let S be a set consisting of one such vertex from each triangle (which can be chosen arbitrarily if more than one exists in a triangle).

We claim that S is independent: if there were an edge between two vertices u, v in S (which must correspond to terms from two different clauses), then the corresponding terms must be negations of each other (as this was the criterion for adding an edge between the two

corresponding vertices when constructing G). However, this cannot be possible if both vertices were added to S , since these correspond to terms that evaluated to 1.

Thus S is an independent set (since no two vertices in S share an edge) with k vertices.

(\Leftarrow) *If G has an independent set of size at least k , then the 3-SAT instance is satisfiable.*

Suppose that G has an independent set S of size at least k . By construction of G , the size of S must be equal to k , because otherwise S would have to contain at least two vertices from one triangle (which all share edges). This means S must consist of exactly one vertex from each triangle in G .

Using this fact, we will construct a satisfying assignment to f as follows: first, set the terms corresponding to each vertex in S to be 1. For example, if $v \in S$ corresponds to a term x_i , then set $x_i = 1$. If v corresponds to a term \bar{x}_j , then set $x_j = 0$. Because S is independent, there cannot be two vertices in S whose corresponding terms contradict each other, since these vertices share edges in G . Thus our assignment of the corresponding x_i 's from S is consistent.

For the remaining vertices in G , we can set the corresponding variables arbitrarily to 1.

The result is that each clause in f evaluates to 1, since each clause contains one vertex from S whose corresponding term was set to 1. Thus the assignment we constructed satisfies f .