CS 473: Fundamental Algorithms, Fall 2011

Discussion 13

November 15, 2011

13.1 FROM SET COVER TO MONOTONE SAT.

Consider an instance I of a CNF formula specified by clauses $C_1, C_2, \ldots, C_k$ over a set of boolean variables $x_1, x_2, \ldots, x_n$. We say that I is ***monotone*** if each term in each clause consists of a nonnegated variable i.e. each term is equal to x_i, for some i, rather than $\overline{x_i}$ (i.e., no negations are allowed). They could be easily satisfied by setting each variable to 1. For example, suppose we have three clauses $(x_1 \vee x_2), (x_1 \vee x_3), (x_3 \vee x_2)$. These could be satisfied by setting all three variables to 1, or by setting x_1 and x_2 to 1 and x_3 to 0.

Given a monotone instance of CNF formula, together with a number k, the problem Monotone Satisfiability asks whether there is a satisfying assignment for the instance in which at most k variables are set to 1.

The Set Cover problem asks, given a collection $\mathcal{F}$ of subsets $S_1, S_2, \ldots, S_m$ of a ground set $U = \{1, \ldots, n\}$, what is the minimum number of sets of $\mathcal{F}$ whose union is U?

(A) Given a decision instance of Set Cover (i.e., given S, $\mathcal{F}$, and a k – is there a cover of U by k subsets?), show a Karp reduction to Monotone Satisfiability.

(B) Show how to solve the optimization version of Set Cover (i.e., you are given $U, \mathcal{F}$, and you have to compute the minimum number of sets of $\mathcal{F}$ that cover the ground set) by an algorithm performing a polynomial number of calls to a solver of Monotone Satisfiability.

13.2 BUILDING 3CNF FORMULAS.

(A) Consider the following boolean function f and g defined by a truth table. Generate a 3CNF formulas that computes these two functions.

x	y	z	$f(x, y, z)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

(i)

x	y	z	$g(x, y, z)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

(ii)

(B) Given an arbitrary boolean formula $f(x, y, z)$, describe how to convert it into an equivalent 3CNF formula.

(C) Argue that any boolean formula with n variables can be converted into a n-CNF formula (i.e., CNF formula where every clause has at most n variables).

13.3 Reducing from 3-coloring to SAT.

SAT is a decision problem that asks whether a given boolean formula in conjunctive normal form (CNF) has an assignment that makes the formula true. The 3-Coloring problem is a decision problem that asks given an undirected graph G, can its vertices be colored with three colors, so that every edge touches vertices with two different colors? Give a polynomial time reduction from 3-coloring to 3SAT.

Comment: Observe that to prove the hardness of 3-Coloring we showed the reduction in the other direction.

13.4 Turning 3SAT into an optimization problem.

Consider the optimization problem MAX SAT.

Problem: MAX SAT

Instance: Set U of variables, a collection C of disjunctive clauses of literals where a literal is a variable or a negated variable in U. ***Question***: Find an assignment that maximized the number of clauses of C that are being satisfied.

(A) Prove that MAX SAT is NP-Hard.

(B) Prove that if each clause has exactly three literals, and we randomly assign to the variables values 0 or 1, then the expected number of satisfied clauses is $(7/8)M$, where $M = |C|$.

(C) Show that for any instance of MAX SAT, where each clause has exactly three different literals, there exists an assignment that satisfies at least $7/8$ of the clauses.

(D) Let (U, C) be an instance of MAX SAT such that each clause has $\geq 10 \cdot \log_2 n$ distinct variables, where n is the number of clauses. Prove that there exists a satisfying assignment; namely, there exists an assignment that satisfy all the clauses of C.

Comment: As such, SAT is easy if all the clauses are large. It is also easy (see the 2SAT problem) if every clause has two variables. Specifically, as the clauses of a SAT become bigger, the problem becomes easier. As such, intuitively 3SAT, where every clause has exactly three variables, is the hardest version of this problem.

13.5 Solve 2SAT in linear time.

In the 2SAT problem, you are given a set of clauses, where each clause is the disjunction (OR) of two literals (a literal is a boolean variable or the negation of a Boolean variable). You are looking for a way to assign a value `true` or `false` to each of the variables so that *all* clauses are satisfied—that is, there is at least one true literal in each clause.

(A) As an example, here's an instance of 2SAT:

$$(x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_3) \wedge (x_1 \vee x_2) \wedge (\overline{x}_3 \vee x_4) \wedge (\overline{x}_1 \vee x_4) \quad .$$

This instance has a satisfying assignment: set x_1, x_2, x_3, and x_4 to `true`, `false`, `false`, and `true`, respectively.

(a) Are there other satisfying truth assignments of this 2SAT formula? If so, find them all.

(b) Give an instance of 2SAT with four variables, and with no satisfying assignment.

(B) We can solve 2SAT efficiently by reducing it to the problem of finding the strongly connected components of a directed graph. Given an instance I of 2SAT with n variables and m clauses, construct a directed graph $G_I = (V, E)$ as follows.

- G_I has $2n$ nodes, one for each variable and its negation.
- G_I has $2m$ edges: for each clause $(\alpha \vee \beta)$ of I (where α, β are literals), G_I has an edge from the negation of α to β, and one from the negation of β to α.

Note that the clause $(\alpha \vee \beta)$ is equivalent to either of the implications $\overline{\alpha} \Rightarrow \beta$ or $\overline{\beta} \Rightarrow \alpha$. In this sense, G_I records all implications in I.

(C) Carry out this construction for the instance of 2SAT given above, and for the instance you constructed in (b).

(D) Show that if G_I has a strongly connected component containing both x and $\overline{x}$ for some variable x, then I has no satisfying assignment.

(E) Now show the converse of (d): namely, that if none of G_I's strongly connected components contain both a literal and its negation, then the instance I must be satisfiable. (*Hint*: Assign values to the variables as follows: repeatedly pick a sink strongly connected component of G_I. Assign value `true` to all literals in the sink, assign `false` to their negations, and delete all of these. Show that this ends up discovering a satisfying assignment.)

(F) Conclude that there is a linear-time algorithm for solving 2SAT.