Reduction from 3 SAT to MAX CUT

CS 4820—March 2015

David Steurer

Recall that a *cut* of a graph is a subset of the notes that is neither empty nor the whole node set. The *capacity* of a cut is the sum of the capacities of the edges crossing the cut.

Problem (MAX CUT). Given an undirected graph G with nonnegative edge capacities and a parameter $c \in \mathbb{R}$, decide if there exists a cut in G with capacity at least c.

The problem of deciding if there exists a cut with capacity at most c is called MIN CUT. This problem has a polynomial time algorithm (for example, using network flows). In contrast, no polynomial time algorithm is known for MAX CUT. The following theorem explains this situation.

Theorem. MAX CUT is NP-hard.

We prove the theorem by a chain of reductions. We reduce from 3-SAT to NAE 4-SAT to NAE 3-SAT to MAX CUT. (The reason for going through NAE SAT is that both MAX CUT and nae sat exhibit a similar kind of symmetry in their solutions.)

Problem (NAE k-SAT). Given a set of clauses, each containing up to k literals, decide if there exists an assignment to the variables such that for every clause the Not-all-equal (NAE) predicate is satisfied, that is, not all literals in the clause have the same truth value.

Claim. 3-SAT reduces in polynomial time to NAE 4-SAT.

Proof. We will give a polynomial-time algorithm A that given a 3-SAT instance constructs an equivalent NAE 4-SAT instance. Given a 3-SAT instance φ, the algorithm A constructs a NAE 4-SAT instance $\varphi' = A(\varphi)$ by adding a variable z to every clauses. (The variable z is distinct from the variables that appear in φ.) For example, the 3-SAT clause $x_1 \vee x_3 \vee \neg x_4$ would be replaced by the NAE 4-SAT clause $\text{NAE}(x_1, x_3, \neg x_4, z)$.[1]

We are to show that φ is satisfiable if and only if φ' is satisfiable. If $x_1, \ldots, x_n$ is a satisfying assignment for φ, then the same assignment satisfies φ when we choose $z = 0$.[2] The reason is the following identity,

$$a \vee b \vee c = \text{NAE}(a, b, c, 0) \tag{1}$$

(In words, a disjunction of terms is true if and only not all terms are equal to 0.) To show the other direction, suppose $x_1, \ldots, x_n, z$ is a satisfying assignment of φ'. Notice that $\neg x_1, \ldots, \neg x_n, \neg z$ is also a satisfying assignment of φ' (because $\text{NAE}(a, b, c, d) = \text{NAE}(\neg a, \neg b, \neg c, \neg d)$). In one of these two assignments, the value assigned to the variable z is 0. This assignment corresponds to a satisfying assignment for φ (again using the identity above).

Claim. NAE 4-SAT reduces in polynomial time to NAE 3-SAT.

Proof. Given an NAE 4-SAT instance φ, we will construct an equivalent NAE 3-SAT instance φ' by splitting every NAE 4-SAT clause $C_i^{(1)} = \text{NAE}(a, b, c, d)$ in φ into two NAE 3-SAT clauses $C_i^{(2)} = \text{NAE}(a, b, w_i)$ and $C_i^2 = \text{NAE}(\neg w_i, c, d)$ that are linked together by an additional new variable w_i.

The correctness of the reduction follows from the following fact: Four Boolean values a, b, c, d are not all equal if and only if there exists a Boolean value w such that $\text{NAE}(a, b, w)$ and $\text{NAE}(\neg w, c, d)$.[3]

Claim. NAE 3-SAT reduces in polynomial time to MAX CUT.

Proof. Given a NAE 3-SAT instance φ, we will construct an equivalent MAX CUT instance (G, c). For every variable x_i of φ, we will add two vertices to G labeled by x_i and $\neg x_i$ and we will connect the two vertices by an edge. We assign capacity $M = 10 \cdot m$ to each of these "variable" edges. (Here, m is the number of clauses in φ and n is the number of variabes.) For every clause C in φ, we will add a "clause" triangle between the vertices corresponding to the terms in C. We assign capacity 1 to each of these "clause" edges.[4]

We claim that G contains a cut with capacity at least $n \cdot M + 2 \cdot m$ if and only if φ is satisfiable.

Suppose φ is satisfiable and consider any satisfying assignment. This assignment corresponds to a cut in G. (One side of the cut consists of all vertices labeled by terms that evaluate to 1 in the assignment. The other side of the cut consists of all vertices labeled by terms that evaluate to 0 in the assignment.) Since exactly one of terms x_i and $\neg x_i$ evalute to 1 in an assignment, all variable edges go across the cut, which contributes $n \cdot M$ to the capacity of the cut. Since the assignment satisfies φ, exactly two edges in every clause triangle go across the cut, which contributes $2 \cdot m$ to the capacity of the cut. In total the capacity of the cut is equal to $n \cdot M + 2 \cdot m$.

On the other, suppose that G contains a cut with capacity at least $n \cdot M + 2 \cdot m$. First, we claim that all variable edges go across this cut. The reason is that any cut that misses at least one of the variable edges has capacity at most $(n - 1) \cdot M + 3 \cdot m = n \cdot M + 3 \cdot m - 10m$, which is strictly smaller than $n \cdot M + 2m$. Next, we claim that exactly two edges of every clause triangle go across the cut. The reason is that no cut can separate three edges of a triangle and therefore if a cut separates fewer than two edges in one of clause triangles, then its capacity is strictly smaller than $n \cdot M + 2m$. Since all variable edge go across, this cut corresponds to an assignment for φ. Furthermore, since the cut separates exactly two edges per clause triangle, the corresponding assignment satisfies all clauses of φ.

Footnotes

1. Here, NAE is the Boolean operation that evaluates to TRUE if and only if not all of its inputs are equal.
2. We use 0 and 1 to abbreviate the Boolean values FALSE and TRUE.
3. I only know how to verify this fact by a somehwat cumbersome case distinction.
4. In this description, we assume that every clause contains three distinct variables. This assumption can be justified by a preprocessing step. Alternatively, we can modify the reduction slightly to accomodate such clauses.