

CAS 705, Computability and Complexity

Assignment 2

Xiao Shu (0901994)
shux@mcmaster.ca

October 22, 2009

Question 1

For a given graph $G = (V, E)$, we can verify whether $S, T \in V$ is a cut and whether its size is no less than k by counting the number of edges crossing between S and T in polynomial time, therefore, $\text{MAXCUT} \in \text{NP}$. We next prove that MAXCUT is also NP-hard by reducing NP-complete problem NAE3SAT to it.

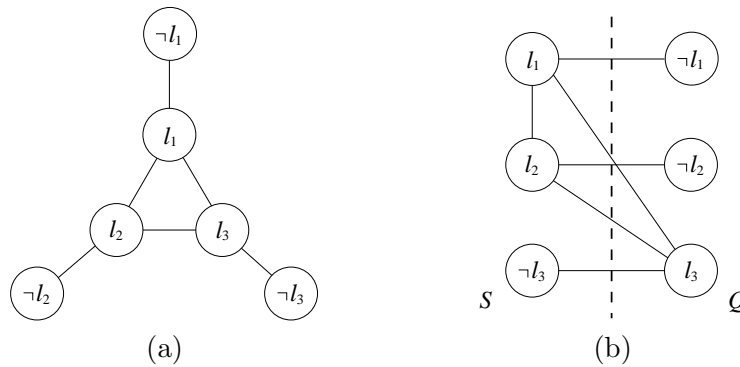


Figure 1: (a) The structure of G_i which is mapped from $c_i = (l_1 \vee l_2 \vee l_3)$. (b) A maximum cut (S, Q) of size 5 of G_i .

The reduction algorithm (transducer) begins with an input of NAE3SAT , $\phi = c_1 \wedge c_2 \wedge \dots \wedge c_n$. It first writes all the distinct variables in ϕ and their negations as the vertices of $G = (V, E)$ on the output tape. Then, for each clause $c_i = (l_1 \vee l_2 \vee l_3)$, the transducer connects every pair of literals and every literal to its negation in G by appending edge set $E_i = \{(l_1, l_2), (l_2, l_3), (l_3, l_1), (l_1, \neg l_1), (l_2, \neg l_2), (l_3, \neg l_3)\}$ to E on the output tape as showed in Figure 1. Note that if x is a variable and $l_i = \neg x$, then $\neg l_i = \neg \neg x = x$. After constructing G , the transducer outputs $k = 5n$ as the second parameter of MAXCUT . Obviously, all these steps can be done on a TM with log-space working tape.

The basic idea of this reduction algorithm is to create a graph $G_i = (V, E_i)$ for each clause c_i , such that a nae-assignment t of ϕ satisfies c_i iff. a cut (S, Q) of G_i is of size 5. $[\implies]$ Let $S = \{l | l \in V, t(l) = T\}$ and $Q = \{l | l \in V, t(l) = F\}$. Each literal in c_i and its negation must belong to different sides of the cut, and one literal must be on the different side to two other literals as t is a nae-assignment. Thus, (S, Q) is a cut of size 5. $[\impliedby]$ Let $t(l) = T$ if $l \in S$, or $t(l) = F$ otherwise. The cut (S, Q) being of size 5 implies that each

literal in c_i must be in different set as its negation and one literal must be in different set as the other two. Therefore, $t(l) = \neg t(\neg l)$ and not all literals in c_i get same assignment, i.e., t is a nae-assignment.

Furthermore, no cut can be larger than 5 as there is a triangle (l_1, l_2, l_3) in G_i . Combining this with the fact that there is no other edge in G besides edges in each G_i , we know that (S, Q) is a maximum cut of G if and only if (S, Q) is a maximum cut of each G_i . Therefore, t is a satisfying nae-assignment of $\phi \iff t$ is a satisfying nae-assignment of each $c_i \iff (S, Q)$ is a maximum cut of size 5 of each $G_i \iff (S, Q)$ is a maximum cut of size $5n$ of $G = \bigcup_{i=1}^n G_i$.

Question 2

(a) Without loss of generosity, we say a clause $(l_1 \vee l_2)$ in α if either $(l_1 \vee l_2)$ or $(l_2 \vee l_1)$ occurs in α . Then $(l_1, l_2) \in E \iff (\neg l_1 \vee l_2) \text{ in } \alpha \iff (\neg l_1 \vee \neg \neg l_2) \text{ in } \alpha \iff (\neg l_2, \neg l_1) \in E$. Thus, if (l_1, l_2, \dots, l_m) is a directed path in G_α , then $(\neg l_i, \neg l_{i-1}) \in E$ for each $1 < i \leq m$, which implies the existence of a directed path from $\neg l_m$ to $\neg l_1$. Now since $(l_1, l_2) \in E$ implies $(\neg l_1 \vee l_2)$, if l_1 is true, l_2 has to be true in order to satisfy α , and so does every node having a path from l_1 .

(b) [\Leftarrow] Based on previous result, assume that there is a directed cycle between x and $\neg x$, then they must be both true if one of them is true, which is not possible. [\Rightarrow] As long as there is no cycle between a variable and its negation in G_α , we can generate a truth assignment for α as follows,

1. Find an unassigned literal l_1 such that there is no path from l_1 to $\neg l_1$.
2. For l_1 and every unassigned l_i to which l_1 connects, assign $t(l_i) = T$ and $t(\neg l_i) = F$.
3. Repeat Step 1, until all literals get assignments.

Since there is no cycle between l_1 and $\neg l_1$, either l_1 does not connect to $\neg l_1$ or $\neg l_1$ does not connect to l_1 , thus, Step 1 can always find such literals, and algorithm will halt eventually. Assume for the sake of contradiction that the assignment t does not satisfy clause $(l_i \vee l_j)$ in α , i.e., $t(l_i) = F$ and $t(l_j) = F$. Since $(l_i \vee l_j)$ implies $(\neg l_i, l_j), (\neg l_j, l_i) \in E$, if either $t(\neg l_i) = T$ or $t(\neg l_j) = T$ is set, the other literal and its negation must be assigned in the same round by Step 2. Thus, there exists literal l_1 from Step 1 which connects to both $\neg l_i, \neg l_j$. However, this suggests that there is path from l_1 to $\neg l_i$ to l_j to $\neg l_1$, which contradicts to the requirement of Step 1.

(c) With previous results, we know that $\alpha \in 2\text{SAT}$ iff no variable and its negation of α are in the same strongly connected component of G_α . Since strongly connected components can be computed in $\Theta(|V| + |E|) = \Theta(|\alpha|)$ time,¹ and checking all variables and their negations requires $\Theta(|V|) = O(|\alpha|)$ time, 2SAT is linear-time decidable. The algorithm showed in (b) also has linear-time implementation, thus if $\alpha \in 2\text{SAT}$, it needs only linear-time to find a truth assignment of α .

Appendix

The solution to Question 1 shows that MAXCUT problem for a general undirected graph is NP-complete. Actually, we can push this a little further and prove that the simple

¹See Thomas H. Cormen, *Introduction to Algorithm*, pages 552-557

undirected graph version of MAXCUT is also NP-complete. Suppose graph $G = (V, E)$ is an output of the transducer from NAE3SAT to MAXCUT. We create a simple graph $G' = (V', E')$ such that, for each edge (x, y) in G , there are two more dummy nodes d_1, d_2 in V' and three more edges $(x, d_1), (d_1, d_2), (d_2, y)$ in E' replacing (x, y) . Then for a cut (S, Q) of G , we put each dummy node on the different side as its adjacent original node, and create a new cut (S', Q') of G' . It can be shown that, (S, Q) is maximum as long as (S', Q') is maximum, and in such case, (S', Q') has $2|E|$ more edges than (S, Q) . Thus, with a transducer $g(\langle G, k \rangle) = \langle G', k + 2|E| \rangle$ and the fact that MAXCUT is NP-complete, we can prove the NP-hardness of the simple undirected graph version of MAXCUT.

Another interesting result worth mentioning is that it is NP-complete to decide the existence of a satisfying assignment for a 3CNF formula, such that each clause of the assignment contains two variables with unequal values.² This can be proved by reducing it from 3SAT. Let transducer f being defined as follows, if two of the literals of c_i are just variables x, y themselves, let $c_i = (x \vee y \vee l) \xrightarrow{f} c'_i = (x \vee y \vee a) \wedge (\neg a \vee l \vee b) \wedge (a \vee b \vee \neg b)$; if two of the literals of c_i are negations of variables $\neg x, \neg y$, let $c_i = (\neg x \vee \neg y \vee l) \xrightarrow{f} c'_i = (\neg x \vee \neg y \vee \neg a) \wedge (a \vee l \vee \neg b) \wedge (a \vee b \vee \neg b)$. Except for the last clause $(a \vee b \vee \neg b)$ which forces a, b picking different values, the rest of the proof is similar to the NP-completeness proof of NAE3SAT.

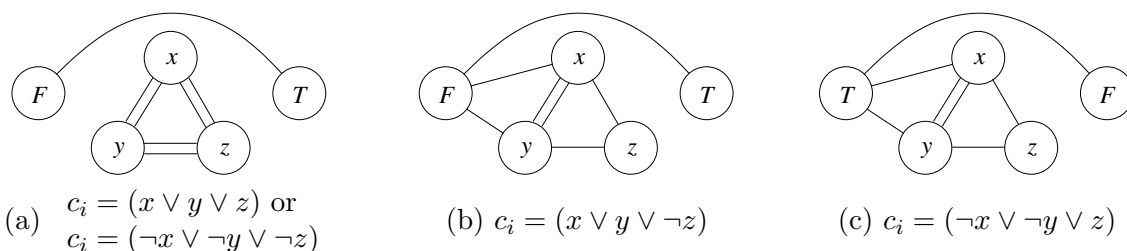


Figure 2: Each type of the clauses maps to a slightly different type of sub-graph of G by the transducer. A valid satisfying assignment of a clause can be translated to a maximum cut of size 5 of the corresponding sub-graph.

Like NAE3SAT, this problem reduces to MAXCUT with a log-space transducer as well. The transducer can be built on the same concept as the one in the solution to Question 1, except it has no negation vertices but two dummy nodes T, F to decide the assignment. Figure 2 shows how the transducer maps each type of clauses to a sub-graph of G . For example, $t(x) = T, t(y) = F, t(z) = F$ is a valid assignment of $c_i = (\neg x \vee \neg y \vee z)$. If we put the variables with true assignment on one side with T and the rest with F on the other, the result is a maximum cut $(\{x, T\}, \{y, z, F\})$ of G_i .

²I took a look at this problem because I misunderstood it as NAE3SAT which actually requires unequal literals in each clause rather than variables, and I found a way to reduce it to MAXCUT before realizing the mistake. It turns out that this is more intriguing than the original problem.