Analysis of Algorithms

V. Adamchik CSCI 570 Fall 2016

Discussion 12 University of Southern California

NP Hardness

Polynomial Reduction

To reduce problem Y to problem X (we write $Y \leq_p X$) we want a function f that maps Y to X such that:

1) f is a polynomial time computable
2) $y \in I_Y$ (instance of Y) is YES if and only if $f(y) \in I_X$ is YES.

In plain form:

reduce an input of Y into an input of X,
solve X,
reduce the solution back to Y.

$Y \leq_p X$

If we can solve X, we can solve Y.

If we can solve X in polynomial time, we can solve Y in polynomial time.

Examples:

Image Segmentation $\leq_p$ Min-Cut

Survey Design $\leq_p$ Max-Flow

$Y \leq_p X$

If we can solve X, we can solve Y.

Negate this statement.

If we cannot solve Y, we cannot solve X.

We use this to prove NP hardness.

Examples: 3-SAT $\leq_p$ Independent Set
Independent Set $\leq_p$ Vertex Cover
Vertex Cover $\leq_p$ Set Cover

P and NP

P = set of problems that can be solved in polynomial time

NP = set of problems for which a solution can be verified in polynomial time

$P \subseteq NP$

Open question: does P = NP?

NP-Hard and NP-Complete

X is *NP-Hard*, if $\forall Y \in NP$ and $Y \leq_p X$.

X is *NP-Complete*, if X is NP-Hard and $X \in NP$.

Cook-Levin Theorem (1971)

SAT is NP-complete

No proof…

Cook received a Turing Award for this work.

Problem 1 (Set Packing)

We are given m sets $S_1, S_2, \dots S_m$ and an integer k. Our goal is to select k of the m sets such that none of the selected sets has any elements in common. Prove that this problem is NP-Complete.

For example, given the sets

$\{1, 3, 5\}, \{1, 2, 3\}, \{2, 4\}, \{2, 5, 7\}, \{6\}$

and the number 3.

Sets $\{1, 3, 5\}$, $\{2, 4\}$, and $\{6\}$ have no elements in common with one another.

Solution

Is it in NP?

We need to show we can verify a solution in polynomial time.

Given k sets. Consider all pairs (a,b), where a and b are from different sets.

This require $O(k^2)$ set comparisons. Since each set is finite, the total number of comparisons is polynomial.

Solution

Is it in NP-hard?

We need to show that ($Y \le_p$ Set Packing) for $\forall Y \in$ NP

Reduce from Independent Set.

$\Longrightarrow$)Assume that G has an independent set of size k.

For each vertex v_i in this set, take all incident edges.

Let us denote these edges by S_i set.

By construction, all S_i are pairwise disjoint sets.

Solution

Independent Set $\le_p$ Set Packing.

$\Longleftarrow$)Assume a set packing C of size k.

Create a graph. Define vertex v_i for each set in C.

There is an edge between v_a and v_b iff the sets a and b intersect.

Now, independent vertex set is a set of those vertices v_i.

Problem 2 (CNF)

Given a Conjunctive Normal Form (CNF)

$$(X_1 \lor \neg X_3) \land (X_1 \lor \neg X_2 \lor X_4 \lor X_5) \land \dots$$

with any number of clauses and any number of literals in each clause. Prove that CNF is polynomial time reducible to 3SAT.

Solution : CNF $\leq_p$ 3-SAT

Is it in NP-hard?

We need to convert any CNF into 3-SAT…

First take care clauses with one or two literals.

X replace by $(X \vee X \vee X)$

$(X \vee Y)$ replace by $(X \vee Y \vee X)$

For clauses with three literals, we do nothing

Solution : CNF $\leq_p$ 3-SAT

For clauses with four literals, we add a new variable

$(a \vee b \vee c \vee d)$ replace by $(a \vee b \vee x) \wedge (\neg x \vee c \vee d)$

For clauses with five literals

$(a \vee b \vee c \vee d \vee e)$ replace by

$(a \vee b \vee x) \wedge (\neg x \vee c \vee y) \wedge (\neg y \vee d \vee e)$

And so on…

Thus, original CNF is satisfiable iff 3SAT is satisfiable.

Problem 3 (The Steiner Tree)

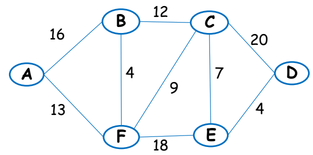
Given an undirected weighted graph $G=(V,E)$ with positive edge costs, a subset of vertices $R \subseteq V$, and a number C. Is there a tree in G that spans all vertices in R (and possibly some other in V) with a total edge cost of at most C ?

Prove that this problem is NP-complete.

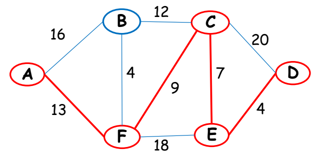
Solution

Example.

R = {A, F,D} and C = 34.



The Steiner Tree:



Is it in NP?

Solution

Is it in NP-Hard?

Vertex Cover $\leq_p$ Steiner Tree

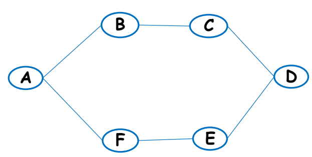
We can create a new graph G′ that starts with a copy of G=(E,V) and

1) add a new vertex for each edge, connected to the two endpoints
2) connect all vertices in V to all vertices in V
3) assign a cost of one to every edge

Solution

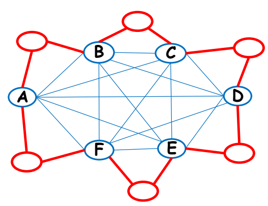
Example.

Graph G

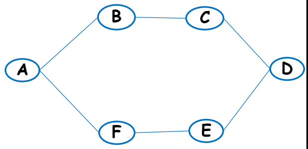


Graph G′ with E + V vertices

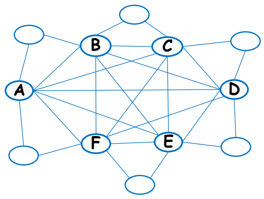
The set R in a Steiner tree is a set of new vertices (in red)



Solution

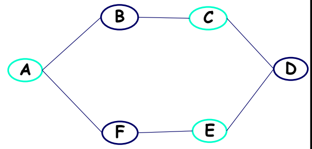


Claim: The new graph G' has a Steiner tree for R =E(G) and cost E+k-1 if and only if the original graph G has a vertex cover of size k.

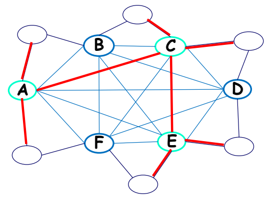


Solution

Proof. Let G has a vertex cover of size k.



A Steiner tree will consist of all new vertices R=E(G) plus the vertex cover.

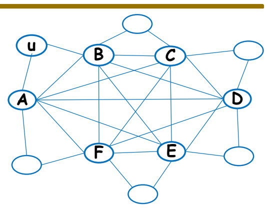


The tree cost is E+k-1.

Solution

Proof.

Let G' has a Steiner tree T for R =E(G) of cost E+k-1.



Remove all R vertices from the Steiner tree T to get

C = T\R = T\E vertices.

We claim that C is a vertex cover.

When we remove u, either A or B must be in C. Thus A or B will cover that edge.

The size of C is T-E = weight(T)+1-E=E+k-1+1-E=k.