CS483 Design and Analysis of Algorithms

Lectures 21-23 NP-Complete Problems

Instructor: Fei Li

lifei@cs.gmu.edu with subject: CS483

Office hours: STII, Room 443, Friday 4:00pm - 6:00pm or by appointments

Course web-site:

http://www.cs.gmu.edu/~lifei/teaching/cs483_fall08/

Figures unclaimed are from books "Algorithms" and "Introduction to Algorithms"

Announcements

Good News!

1. There is no class this Wednesday (November 26th).
2. Happy Thanksgiving!
3. We have a review class on December 3rd, the last class. We will cover the solutions of Assignment 5 - 8.
4. We will review materials discussed after the midterm.

Not-that-good News!

1. Assignment 8 is released today. The due date is December 3rd (Wednesday).

Again, Good News!

1. Assignment 8 is EASY!

Overview

Hard problems (**NP**-complete)	Easy problems (in **P**)
3SAT	2SAT, HORN SAT
TRAVELING SALESMAN PROBLEM	MINIMUM SPANNING TREE
LONGEST PATH	SHORTEST PATH
3D MATCHING	BIPARTITE MATCHING
KNAPSACK	UNARY KNAPSACK
INDEPENDENT SET	INDEPENDENT SET on trees
INTEGER LINEAR PROGRAMMING	LINEAR PROGRAMMING
RUDRATA PATH	EULER PATH
BALANCED CUT	MINIMUM CUT

Warmup Questions

Problem

If a problem A can be solved exactly using Divide-and-Conquer*, is A a* hard *problem?*

Problem

If a problem A can be solved exactly using the Greedy Method *(assume each step costs polynomial time), is A a* hard *problem?*

Problem

If a problem A can be solved exactly using Dynamic Programming*, is A a* hard *problem?*

Problem

If a problem A can be solved exactly using Linear Programming*, is A a* hard *problem?*

Problem

If a problem A can be solved in polynomial time using a randomized algorithm*, is A a* hard *problem?*

Claim

Running time cannot be estimated from the approach you are using

Some Typical Hard Problems

1. Satisfiability
2. Traveling Salesman Problem
3. Euler and Rudrata
4. Cuts and Bisections
5. Integer Linear Programming
6. Three Dimensional Matching
7. Independent Set, Vertex Cover, and Cliques
8. Longest Path
9. Knapsack and Subset Sum

Satisfiability — **SAT**

Definition

Literal: a Boolean variable x or $\bar{x}$

Definition

Disjunction: logical *or*, denoted $\wedge$

Definition

Clause: e.g., Boolean formula in conjunctive normal form (CNF)

$$(x \wedge y \wedge z)(x \wedge \bar{y})(y \wedge \bar{z})(z \wedge \bar{x})(\bar{x} \wedge \bar{y} \wedge \bar{z})$$

Definition

Satisfying truth assignment. An assignment of `false` or `true` to each variable so that every clause it to evaluate is `true`

Lemma

For formulas with n variables, we can find the answer in time 2^n. In a particular case, such assignment may not exist.

Proof.

? □

Satisfiability — **SAT**

Definition

Literal: a Boolean variable x or $\bar{x}$

Definition

Disjunction: logical *or*, denoted $\wedge$

Definition

Clause: e.g., Boolean formula in conjunctive normal form (CNF)

$$(x \wedge y \wedge z)(x \wedge \bar{y})(y \wedge \bar{z})(z \wedge \bar{x})(\bar{x} \wedge \bar{y} \wedge \bar{z})$$

Definition

Satisfying truth assignment. An assignment of `false` or `true` to each variable so that every clause it to evaluate is `true`

Lemma

In a particular case — Horn formula, *a satisfying truth assignment, if one exists, can be found by (?) in polynomial time*

Lemma

In a particular case (each clause has only two literals), **SAT** *can be solved in polynomial time (linear, quadratic, etc?) by (?) algorithm*

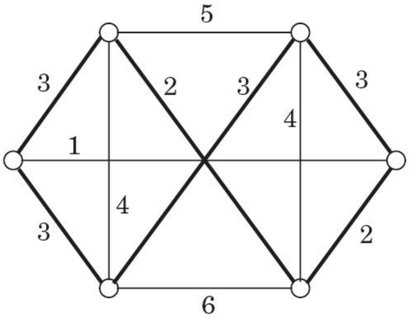
The Traveling Salesman Problem — **TSP**

Definition

TSP. We are given n vertices $1, 2, \ldots, n$ and all $(n \cdot (n-1))/2$ distances between them, as well as a budget b. We are asked to find a *tour*, a cycle that passes through every vertex exactly once, of total cost b or less — or to report that no such tour exists

$$d_{\tau(1),\tau(2)} + d_{\tau(2),\tau(3)} + \ldots + d_{\tau(n),\tau(1)} \leq b$$

The optimal traveling salesman tour, shown in bold, has length 18

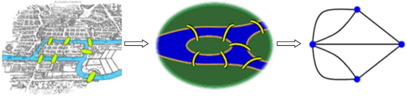


Euler and Rudrata

Definition

Euler. Look for a path that goes through each edge exactly once — When can a graph be drawn without lifting the pencil from the paper?

from http://en.wikipedia.org/wiki/Euler



Theorem

If an only if (a.) the graph is connected and (b.) every vertex, with the possible exception of two vertices (the start and final vertices of the walk), has even degree.

Proof.

?

Rudrata

Definition

Rudrata. Can one visit all the squares of the chessboard, without repeating any square, in one long walk that ends at the starting square and at each step makes a legal knight move?

Definition

Rudrata Cycle. Given a graph, find a cycle that visits each *vertex* exactly once — or report that no such cycle exists.

Cuts and Bisections

Definition

Cut. A set of edges whose removal leaves a graph disconnected

Theorem

MIN-CUT can be solved in polynomial time

Proof.

? □

Definition

Balanced cut. Given a graph with n vertices and a budget b, partition the vertices into two sets S and T such that $|S|, |T| \geq \frac{n}{3}$ and such that there are at most b edges between S and T

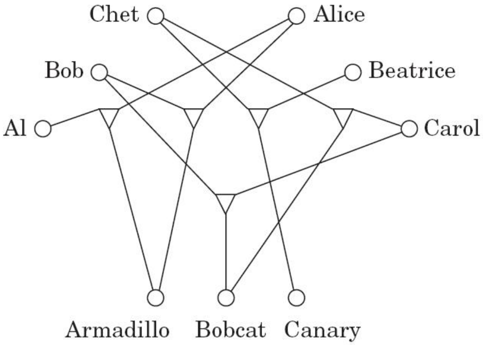
Integer Linear Programming

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} a_j \cdot x_j + \sum_{i=1}^{m} \sum_{j=1}^{n} b_{ij} y_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{n} y_{ij} = 1, \quad \forall i = 1, 2, \dots, m \\ & y_{ij} \le x_j, \quad i = 1, 2, \dots, m, \; j = 1, 2, \dots, n \\ & y_{ij}, x_j \in \mathbb{N}^+ \end{aligned}$$

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} a_j \cdot x_j + \sum_{i=1}^{m} \sum_{j=1}^{n} b_{ij} y_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{n} y_{ij} = 1, \quad \forall i = 1, 2, \dots, m \\ & y_{ij} \le x_j, \quad i = 1, 2, \dots, m, \; j = 1, 2, \dots, n \\ & y_{ij}, x_j \in \{0, 1\} \end{aligned}$$

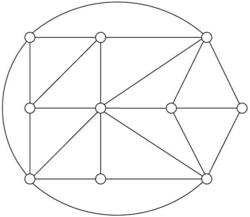
Three Dimensional Matching

Figure 8.4 A more elaborate matchmaking scenario. Each triple is shown as a triangular-shaped node joining boy, girl, and pet.



Independent Set, Vertex Cover, and Clique

What is the size of the largest independent set in this graph?



Definition

Independent Set. Find g vertices that are independent, i.e., no two of them have an edge between them

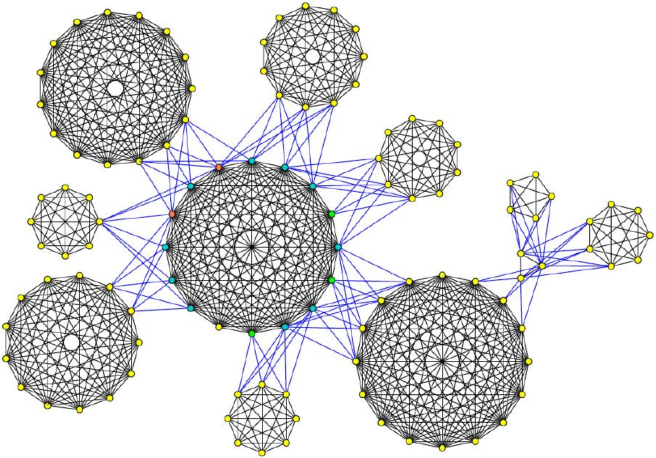
Definition

Vertex Cover. Find b vertices that cover every edge

Definition

Clique. Find a set of g vertices such that all possible edges between them are present

Longest Path



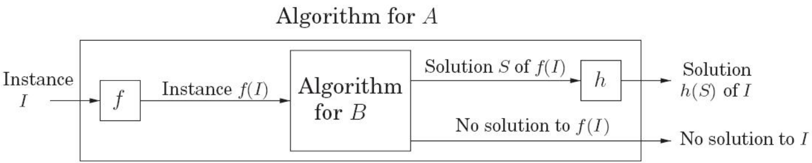
http://www.jaist.ac.jp/ s-teramo/

Reduction

Definition

Problem A polynomial reduces to problem B if arbitrary instances of problem A can be solved using

1. Polynomial number of standard computational steps, plus
2. Polynomial number of calls to oracle that solves problem B



P, NP and NP-Complete

Definition

Search problems. Any proposed solution can be quickly (in polynomial time of the input size) checked for correctness

Definition

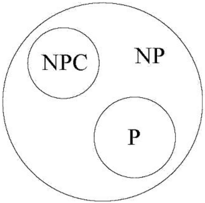
P. The class of all search problems that can be solved in polynomial time

Definition

NP. The class of all search problems

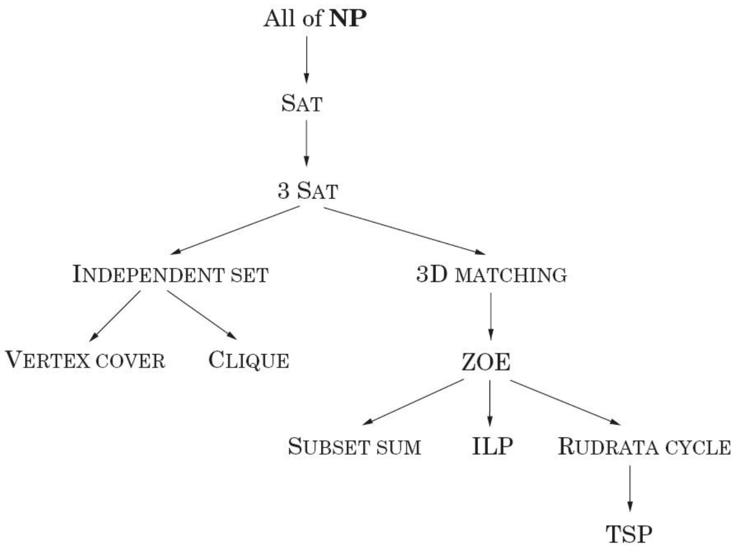
Definition

NP-complete. A problem is NP-complete if all other search problems reduce to it. (A hardest search NP problem.)



Reduction

Figure 8.7 Reductions between search problems.



3SAT $\rightarrow$ Independent Set

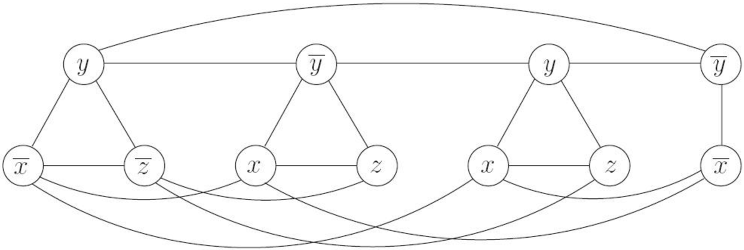
3SAT $\longrightarrow$ Independent Set

1. $$(\bar{x} \vee y \vee \bar{z})(x \vee \bar{y} \vee z)(x \vee y \wedge z)(\bar{x} \vee \bar{y})$$

3SAT $\longrightarrow$ Independent Set

1. $$(\bar{x} \vee y \vee \bar{z})(x \vee \bar{y} \vee z)(x \vee y \wedge z)(\bar{x} \vee \bar{y})$$

2. **Figure 8.8** The graph corresponding to $(\overline{x} \vee y \vee \overline{z})\ (x \vee \overline{y} \vee z)\ (x \vee y \vee z)\ (\overline{x} \vee \overline{y})$

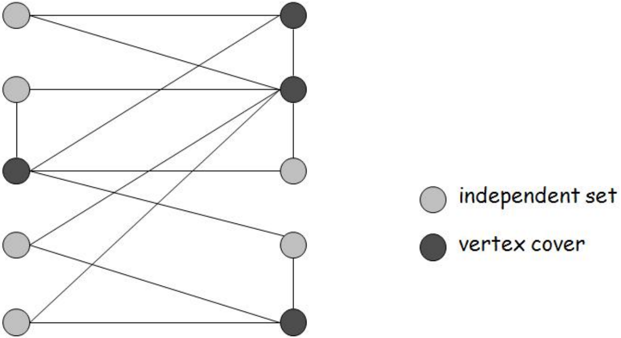


SAT $\rightarrow$ 3SAT

$$(a_1 \vee a_2 \vee \ldots \vee a_k) \rightarrow (a_1 \vee a_2 \vee y_1)(\bar{y}_1 \vee a_3 \vee y_2)(\bar{y}_2 \vee a_4 \vee y_3) \cdots (\bar{y}_{k-3} \vee a_{k-1} \vee a_k)$$

$$\left\{ \begin{array}{c} (a_1 \vee a_2 \vee \cdots \vee a_k) \\ \text{is satisfied} \end{array} \right\} \Longleftrightarrow \left\{ \begin{array}{c} \text{there is a setting of the } y_i\text{'s for which} \\ (a_1 \vee a_2 \vee y_1)\ (\overline{y}_1 \vee a_3 \vee y_2)\ \cdots\ (\overline{y}_{k-3} \vee a_{k-1} \vee a_k) \\ \text{are all satisfied} \end{array} \right\}$$

Independent Set $\rightarrow$ Vertex Cover, Clique



from Wayne's slides on "Algorithm Design"

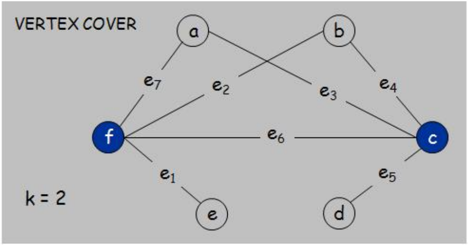
Theorem

Define the complement of a graph $G = (V, E)$ to be $\bar{G} = (V, \bar{E})$, where $\bar{E}$ contains precisely those unordered pairs of vertices that are not in E. A set of nodes S is an independent set of G if and only if S is a clique of $\bar{G}$

Proof.

?

Set Cover → Vertex Cover



SET COVER

$U = \{ 1, 2, 3, 4, 5, 6, 7 \}$
$k = 2$
$S_a = \{3, 7\}$ $S_b = \{2, 4\}$
$S_c = \{3, 4, 5, 6\}$ $S_d = \{5\}$
$S_e = \{1\}$ $S_f = \{1, 2, 6, 7\}$

from Wayne's slides on "Algorithm Design"

Beyond NP-hard

Theorem

Any problem in NP $\rightarrow$ SAT

Proof.

? □

Theorem

There exists algorithms running in exponential time for NP problems

```
function paradox(z: file)

    1: if terminates(z, z)
         go to 1;
```

Theorem

Some problems do not have algorithms

For example, find out x, y, z to satisfy

$$x^3yz + 2y^4z^2 - 7xy^5z = 6.$$