Analysis of Algorithms, I CSOR W4231.002

Eleni Drinea Computer Science Department

Columbia University

Tuesday, April 12, 2016



2 Representative \mathcal{NP} -complete problems

3 Minimum-weight Set Cover



2 Representative *NP*-complete problems

3 Minimum-weight Set Cover

Complexity classes $\mathcal{P}, \mathcal{NP}$ and \mathcal{NP} -complete

Definition 1.

We define \mathcal{P} to be the set of problems that can be solved by polynomial-time algorithms.

Definition 2.

We define \mathcal{NP} to be the set of decision problems that have an efficient certifier.

Fact 3.

 $\mathcal{P}\subseteq\mathcal{NP}$

Definition 4.

A problem X(D) is \mathcal{NP} -complete if

- 1. $X(D) \in \mathcal{NP}$ and
- 2. for all $Y \in \mathcal{NP}, Y \leq_P X$.

Why we should care whether a problem is \mathcal{NP} -complete

If a problem is \mathcal{NP} -complete, we need to stop looking for efficient algorithms for the general problem.

Instead we have a number of options, such as

- 1. approximation algorithms
 - ▶ mathematically rigorous basis to study heuristics
 - distinguish between various optimization problems in terms of how well they can be approximated
- 2. work on interesting special cases
- 3. study the average performance of the algorithm
- $4. \ heuristics$

Suppose we had an \mathcal{NP} -complete problem X.

To show that another problem Y is \mathcal{NP} -complete, we use transitivity of reductions. So we "only" need show that

- 1. $Y \in \mathcal{NP}$
- 2. $X \leq_P Y$

The first \mathcal{NP} -complete problem

Theorem 5 (Cook-Levin).

Circuit SAT is \mathcal{NP} -complete.

SAT: Given a formula ϕ in CNF with n variables and m clauses, is ϕ satisfiable?

3SAT: Given a formula ϕ in CNF with *n* variables and *m* clauses such that each clause has exactly 3 literals, is ϕ satisfiable?

Circuit-SAT: Given a boolean combinatorial circuit C, is there an assignment of truth values to its inputs that causes the output to evaluate to 1?

Lemma 6.

Circuit-SAT $\leq_P SAT$, SAT $\leq_P 3SAT$ and $3SAT \leq_P IS(D)$

So far, we have stated (with or without proofs) that

- Circuit-SAT is \mathcal{NP} -complete
- Circuit-SAT \leq_P SAT
- ▶ SAT \leq_P 3SAT
- $\Rightarrow~SAT~{\rm and}~SSAT~{\rm are}~\mathcal{NP}\text{-complete}.$

Is IS(D) as "hard" as SAT?

So far, we have stated (with or without proofs) that

- Circuit-SAT is \mathcal{NP} -complete
- Circuit-SAT \leq_P SAT
- ▶ SAT \leq_P 3SAT
- \Rightarrow SAT and 3SAT are \mathcal{NP} -complete.

Claim 1.

IS(D) is \mathcal{NP} -complete.

Proof.

Reduction from 3SAT.

Given an arbitrary instance formula ϕ of **3SAT**, we need to transform it into a graph G and an integer k, so that

- 1. The transformation is completed in polynomial time.
- 2. The instance (G, k) is a yes instance of IS(D) if and only if

 ϕ is a **yes** instance of **3SAT**.

Given an arbitrary instance formula ϕ of **3SAT**, we need to transform it into a graph G and an integer k, so that

- 1. The transformation is completed in polynomial time.
- 2. G has an independent set of size at least k if and only if ϕ is satisfiable

Example: given

 $\phi = (x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor \neg x_3) \land (x_1 \lor \neg x_2 \lor \neg x_3)$

 $\operatorname{construct}$

(G,k)

Given an arbitrary instance formula ϕ of **3SAT**, we need to transform it into a graph G and an integer k, so that

- 1. The transformation is completed in polynomial time.
- G has an independent set of size at least k
 if and only if

 ϕ is satisfiable.

Remark 1.

- ► Heart of reduction X ≤_P Y: understand why some small instance of Y makes it difficult.
- ▶ For IS(D), such an instance is a triangle: it's not clear which of its vertices to add to our independent set.

When reducing from **3SAT**, we often use **gadgets**. Gadgets are constructions that ensure:

- 1. Consistency of truth values in a truth assignment: once x_i is assigned a truth value, we must henceforth consistently use it under this truth value.
- 2. Clause constraints: since ϕ is in CNF, we must provide a way to satisfy every clause. Equivalently, we must exhibit at least one literal that is set to 1 in every clause.

In effect, these gadgets will allow us to derive a valid and satisfying truth assignment for ϕ when the transformed instance is a **yes** instance of our problem, so we can prove equivalence of the two instances.

Clause constraint gadget: for every clause, introduce a triangle where a node is labelled by a literal in the clause.

Example: $\phi = (x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor \neg x_3) \land (x_1 \lor \neg x_2 \lor \neg x_3)$

- Hence our graph G consists of m isolated triangles.
- The max independent set in this graph has size m: pick one vertex from every triangle. So we will set k = m.

Goal: derive a truth assignment from our independent set S. **Idea:** when a node from a triangle is added to S, set the corresponding literal to 1.

Consistency gadgets

- 2. Is this truth assignment consistent?
 - Suppose x_1 was picked from the first triangle.
 - Can still pick $\overline{x_1}$ from the second triangle!
 - But then we are setting x_1 to both 1 and 0.
 - \Rightarrow This is obviously **not** a valid truth assignment!

Consistency of truth assignment: must ensure that we cannot add a node labelled x_i and a node labelled $\overline{x_i}$ to our independent set.

Consistency gadgets

- 2. Is this truth assignment consistent?
 - Suppose x_1 was picked from the first triangle.
 - Can still pick $\overline{x_1}$ from the second triangle!
 - But then we are setting x_1 to both 1 and 0.
 - \Rightarrow This is obviously **not** a valid truth assignment!

Consistency of truth assignment: must ensure that we cannot add a node labelled x_i and a node labelled $\overline{x_i}$ to our independent set.

Consistency gadget: add edges between all occurrences of x_i and $\overline{x_i}$, for every *i*, in *G*.

Constructed instance (G, k) of IS(D)

Example: given the formula ϕ below (n=m=3)

$$\varphi = (x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor \neg x_3) \land (x_1 \lor \neg x_2 \lor \neg x_3),$$

the derived graph G is as follows:



Remark: the construction requires time polynomial in the size of ϕ .

We need to show that

 ϕ is satisfiable $\label{eq:general} \begin{array}{c} f \mbox{ an only if} \\ G \mbox{ has an independent set of size at least } m \end{array}$

Proof of equivalence, reverse direction

- Suppose that G has an independent set S of size m.
- Then every triangle contributes one node to S.
- ► Define the following truth assignment
 - ▶ Set the literal corresponding to that node to 1.
 - ► Any variables left unset by this assignment may be set to 0 or 1 arbitrarily.

$$\varphi = (\mathsf{x}_1 \lor \mathsf{x}_2 \lor \mathsf{x}_3) \land (\neg \mathsf{x}_1 \lor \mathsf{x}_2 \lor \neg \mathsf{x}_3) \land (\mathsf{x}_1 \lor \neg \mathsf{x}_2 \lor \neg \mathsf{x}_3)$$



Independent set $S = \{X_1, X_2, X_1\}$

Derived truth assignment: X1=1, X2=1, X3=0

Proof of equivalence, reverse direction

- Suppose that G has an independent set S of size m.
- Then every triangle contributes one node to S.
- ▶ Define the following truth assignment
 - ▶ Set the literal corresponding to that node to 1.
 - ► Any variables left unset by this assignment may be set to 0 or 1 arbitrarily.

We need to show that this truth assignment

- 1. is valid
- 2. satisfies ϕ

Proof of equivalence, reverse direction

- Suppose that G has an independent set S of size m.
- Then every triangle contributes one node to S.
- ► Define the following truth assignment
 - ▶ Set the literal corresponding to that node to 1.
 - ► Any variables left unset by this assignment may be set to 0 or 1 arbitrarily.

We need to show that this truth assignment

- 1. is valid: by construction, $x_i, \overline{x_i}$ cannot **both** appear in S.
- 2. satisfies ϕ : since every triangle contributes one node to S, every clause has a true literal, thus every clause is satisfied.

Proof of equivalence, forward direction

- ▶ Now suppose there is a satisfying truth assignment for ϕ .
- ▶ Then there is (at least) one true literal in every clause.
- Construct an independent set S as follows:
 From every triangle, add to S a node labelled by such a literal; hence S has size m.

We claim that S thus constructed is indeed an independent set.

Proof of equivalence, forward direction

- ▶ Now suppose there is a satisfying truth assignment for ϕ .
- ▶ Then there is (at least) one true literal in every clause.
- Construct an independent set S as follows:
 From every triangle, add to S a node labelled by such a literal; hence S has size m.

We claim that S thus constructed is indeed an independent set.

- 1. S would not be an independent set *if* there was an edge between any two nodes in it.
- 2. Since all nodes in S belong to *different* triangles, an edge implies that the two nodes are labelled by opposite literals.
- 3. Impossible: all literals in S evaluate to 1.

Common pitfalls when showing \mathcal{NP} -completeness

- 1. Carry out the reduction in the wrong direction
- 2. Reduce from a problem not known to be \mathcal{NP} -complete
- 3. Exponential-time transformations
 - Subsets, permutations
- 4. Neglect to carefully prove both directions of equivalence of the original and the derived instances; that is, x is a **yes** instance of X if and only if y = R(x) is a **yes** instance of Y
- 5. Neglect to show that the problem is in \mathcal{NP}

Suggestions

- You should think carefully which problem is most suitable to reduce from
- ▶ In absence of other ideas, reduce from 3SAT

1 Review of last lecture **IS(D)** \leq_P **3SAT**

2 Representative \mathcal{NP} -complete problems

3 Minimum-weight Set Cover

The Traveling Salesman Problem (TSP)

Tour: a *simple* cycle that visits *every* vertex exactly once.

Definition 7 (TSP(D)).

Given n cities $\{1, \ldots, n\}$, a set of non-negative distances d_{ij} between every pair of cities and a budget B, is there a tour of length $\leq B$?

Equivalently, is there a permutation π such that

- 1. $\pi(1) = \pi(n+1) = 1$; that is, we start and end at city 1
- 2. the total distance travelled satisfies

$$\sum_{i=1}^n d_{\pi(i)\pi(i+1)} \le B$$

Application: Google street view car

Example instance of TSP



Depending on the distances, TSP instances may be

- Asymmetric: $d_{ij} \neq d_{ji}$
- Symmetric: $d_{ij} = d_{ji}$
- Metric: satisfy the triangle inequality $d_{ij} \leq d_{ik} + d_{kj}$
- ► Euclidean: e.g., cities are in \mathcal{R}^2 hence city *i* corresponds to point (x_i, y_i) ; then $d_{ij} = \sqrt{(x_i x_j)^2 + (y_i y_j)^2}$

Hamiltonian Cycle: Given a graph G = (V, E), is there a simple cycle that visits every vertex exactly once?

Claim 2.

Hamiltonian Cycle is \mathcal{NP} -complete.

Proof: Reduction from **3SAT** (e.g., see your textbook).

Claim 3.

TSP(D) is \mathcal{NP} -complete.

Proof: reduction from Hamiltonian Cycle.

Proof of Claim 3 (Hamiltonian Cycle \leq_P TSP(D))

- 1. Start from an arbitrary instance of Hamiltonian Cycle, that is, an undirected graph G = (V, E).
- 2. Construct the following instance (G' = (V', E', w), B) of TSP(D): G' is a *complete* weighted graph with V' = V such that for every edge $e \in E'$,

$$w_e = \begin{cases} 1, & \text{if } e \in E \\ 2, & \text{otherwise} \end{cases}$$

3. Set the budget B = n.

This completes the reduction transformation.

Equivalence of the instances is straightforward:

- If G has a hamiltonian cycle, that cycle is a tour of length n in G'.
- ► If G' has a tour of length n, it must consist of edges of weight 1 (why?); thus all these edges appear in G.

- ► Claim 2 also holds for directed Hamiltonian cycle. An exact analog of the proof of Claim 3 then shows that asymmetric TSP is *NP*-complete.
- It is possible to reduce Hamiltonian cycle to Euclidean TSP, thus showing that even Euclidean TSP is \mathcal{NP} -complete.
- However, these problems are not similar in terms of how well they can be approximated: it is possible to provide very good approximate solutions to Euclidean TSP, which is not the case for Symmetric TSP.

- ▶ Set Packing: given a set U of n elements, a collection S_1, S_2, \ldots, S_m of subsets of U, and a number k, is there a collection of at least k subsets such that no two of them intersect?
- ▶ 3D-Matching: Given disjoint sets B, G, H, each of size n, and a set of triples $T \subseteq B \times G \times H$, is there a set of n triples in T, no two of which have an element in common? Reduction from 3SAT.

- Subset sum: Given natural numbers w₁,..., w_n and a (large) target weight W, is there a subset of w₁,..., w_n that adds up exactly to W?
 Applications: cryptography, scheduling
- ▶ Minimum-weight solution to linear equations: Given a system of linear equations in n variables with integer constants, and an integer $B \le n$, does it have a rational solution with at most B non-zero entries?

Applications: coding theory, signal processing

Similar problems with very different complexities

\mathcal{NP}	\mathcal{P}
max cut	min cut
longest path	shortest path
3D matching	matching
Hamiltonian cycle	Euler cycle
3-colorability	2-colorability
3-SAT	2-SAT
LCS of n sequences	LCS of 2 sequences

More on \mathcal{NP} -completeness:

- Computers and Intractability: A guide to the theory of NP-completeness, by Garey and Johnson
- ► Computational Complexity, by C. Papadimitriou



2 Representative *NP*-complete problems



Input

- a set $E = \{e_1, e_2, \dots, e_n\}$ of n elements
- ▶ a collection of subsets of these elements S_1, S_2, \ldots, S_m , where each $S_j \subseteq E$
- a non-negative weight w_j for every subset S_j

Output

A minimum-weight collection of subsets that cover all of E.

In symbols: find an $I \subseteq \{1, ..., m\}$ such that $\bigcup_{i \in I} S_i = E$ and $\sum_{i \in I} w_i$ is minimum.

(Unweighted Set Cover: $w_j = 1$ for all j)

Example instance of Set Cover



n = 8 ground elements, m = 6 subsets with weights $w_1 = w_2 = w_3 = w_4 = 1, w_5 = w_6 = 1 + \epsilon$ $(0 < \epsilon < 1/2)$ Goal: detect features of viruses that do not occur in typical applications

- Ground elements: computer viruses $(n \approx 150)$
- ▶ Sets: labelled by some three-byte sequence occurring in these viruses but not occurring in typical computer applications ($m \approx 21000$); each set consisted of all the viruses that contained the three-byte sequence
- ▶ Goal: output a small number of such sequences (much smaller than 150) that *cover* all known viruses

Claim 4.

Set-Cover(D) is \mathcal{NP} -complete.

Proof.

Reduction from VC(D).

- Let $E = \{e_1, \ldots, e_m\}$ be the set of edges in the graph
 - ▶ These are the ground elements we are trying to *cover*.
- Let S_j be the set of edges (ground elements) that are *covered* by vertex *i*.
 - A vertex j covers all edges adjacent to it.
- Set $w_j = 1$ for all $1 \le j \le n$.

Equivalence of instances: input graph has a vertex cover of size k if and only if E can be covered by k sets.