NP-COMPLETE REDUCTIONS

David Kauchak
CS 140 – Fall 2024

1

Admin

Assignment 11 out today (last one!)

Midterm 3 Thursday

- Extra mentor hours:
  - Tuesday 6:30-8:30pm
  - Wednesday 4:30-6:30pm
- A few sample problems posted

No class next Tuesday

Review on Tuesday after break

2

P problems

P = problems with a polynomial runtime solution

Also, called "tractable" problems

(Basically, all of the problems in this class)

3

NP problems

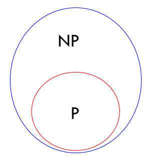
NP is the set of problems that can be *verified* in polynomial time

A problem can be verified in polynomial time if you can check that a given solution is correct in polynomial time

(NP is an abbreviation for non-deterministic polynomial time)

4

P and NP



Big-O allowed us to group algorithms by run-time

Today, we're talking about sets of problems grouped by how easy they are to solve

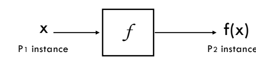
P = problems with a polynomial runtime solution (tractable)

5

Reduction function

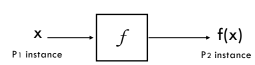
Given two problems P_1 and P_2 a *reduction function*, *f(x)*, is a function that transforms a problem instance *x* of type P_1 to a problem instance of type P_2

such that: a solution to *x* exists for P_1 iff a solution for *f(x)* exists for P_2

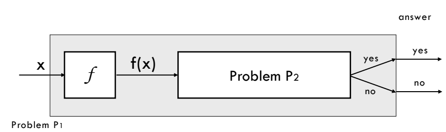


6

Reduction function



Allows us to solve P_1 problems if we have a solver for P_2



7

Proving NP-completeness

Given a problem NEW to show it is NP-Complete

1. Show that NEW is in NP
   a. Provide a verifier
   b. Show that the verifier runs in polynomial time
2. Show that NEW is NP-Hard (i.e., all NP-complete problems are reducible to NEW in polynomial time)
   a. Describe a reduction function *f* from a known NP-Complete problem to NEW
   b. Show that *f* runs in polynomial time
   c. Show that a solution exists to the NP-Complete problem IFF a solution exists *to the NEW problem generate by f*

8

Proving NP-completeness

Show that a solution exists to the NP-Complete problem IFF a solution exists *to the NEW problem generate by f*

- Assume we have an NP-Complete problem instance that has a solution, show that the NEW problem instance generated by *f* has a solution

  yes $\dashrightarrow$ yes

- Assume we have a problem instance of NEW *generated by f* that has a solution, show that we can derive a solution to the NP-Complete problem instance

  yes $\dashleftarrow$ yes

Other ways of proving the IFF, but this is often the easiest

9

NP-complete: 3-SAT

A boolean formula is in *n-conjunctive normal form* (*n*-CNF) if:

- it is expressed as an AND of clauses
- where each clause is an OR of no more than *n* variables

$$(a \vee \neg a \vee \neg b) \wedge (c \vee b \vee d) \wedge (\neg a \vee \neg c \vee \neg d)$$

3-SAT: Given a 3-CNF boolean formula, is it satisfiable?

3-SAT is an NP-complete problem

10

NP-complete: SAT

Given a boolean formula of *n* boolean variables joined by *m* connectives (AND, OR or NOT) is there a setting of the variables such that the boolean formula evaluate to true?

$$(a \wedge b) \vee (\neg a \wedge \neg b)$$

$$((\neg(b \vee \neg c) \wedge a) \vee (a \wedge b \wedge c)) \wedge c \wedge \neg b$$

Is SAT an NP-complete problem?

11

NP-complete: SAT

Given a boolean formula of *n* boolean variables joined by *m* connectives (AND, OR or NOT) is there a setting of the variables such that the boolean formula evaluate to true?

$$((\neg(b \vee \neg c) \wedge a) \vee (a \wedge b \wedge c)) \wedge c \wedge \neg b$$

1. Show that SAT is in NP
   a. Provide a verifier
   b. Show that the verifier runs in polynomial time
2. Show that NEW is NP-Hard (i.e., all NP-complete problems are reducible to NEW in polynomial time)
   a. Describe a reduction function *f* from a known NP-Complete problem to SAT
   b. Show that *f* runs in polynomial time
   c. Show that a solution exists to the NP-Complete problem IFF a solution exists *to the SAT problem generate by f*

12

NP-Complete: SAT

1. Show that SAT is in NP
   a. Provide a verifier
   b. Show that the verifier runs in polynomial time

Verifier: A solution consists of an assignment of the variables
- If clause is a single variable:
  - return the value of the variable
- otherwise
  - for each clause:
    - call the verifier recursively
    - compute a running solution

polynomial run-time?

13

NP-Complete: SAT

Verifier: A solution consists of an assignment of the variables
- If clause is a single variable:
  - return the value of the variable
- otherwise
  - for each clause:
    - call the verifier recursively   linear time
    - compute a running solution

- at most a linear number of recursive calls (each call makes the problem smaller and no overlap)
- overall polynomial time

14

NP-Complete: SAT

2. Show that all NP-complete problems are reducible to SAT in polynomial time
   a. Describe a reduction function *f* from a known NP-Complete problem to SAT
   b. Show that *f* runs in polynomial time
   c. Show that a solution exists to the NP-Complete problem IFF a solution exists *to the SAT problem generate by f*

Reduce 3-SAT to SAT:
- Given an instance of 3-SAT, turn it into an instance of SAT

Reduction function:
- DONE ☺

- Runs in constant time! (or linear if you have to copy the problem)

15

NP-Complete: SAT

Show that a solution exists to the NP-Complete problem IFF a solution exists *to the NEW problem generate by f*

- Assume we have an NP-Complete problem instance that has a solution, show that the NEW problem instance generated by *f* has a solution
- Assume we have a problem instance of NEW *generated by f* that has a solution, show that we can derive a solution to the NP-Complete problem instance

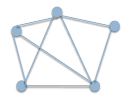
- Assume we have a 3-SAT problem with a solution:
  - Because 3-SAT problems are a subset of SAT problems, then the SAT problem will also have a solution
- Assume we have a problem instance generated by our reduction with a solution:
  - Our reduction function simply does a copy, so it is already a 3-SAT problem
  - Therefore the variable assignment found by our SAT-solver will also be a solution to the original 3-SAT problem

16

CLIQUE

A *clique* in an undirected graph G = (V, E) is a subset V' ⊆ V of vertices that are fully connected, i.e. every vertex in V' is connected to every other vertex in V'

CLIQUE problem: Does G contain a clique of size k?



Is there a clique of size 4 in this graph?

17

HALF-CLIQUE

Given a graph G, does the graph contain a clique containing exactly half the vertices?

Is HALF-CLIQUE an NP-complete problem?

18

Is Half-Clique NP-Complete?

1. Show that NEW is in NP
   a. Provide a verifier
   b. Show that the verifier runs in polynomial time
2. Show that all NP-complete problems are reducible to NEW in polynomial time
   a. Describe a reduction function *f* from a known NP-Complete problem to NEW
   b. Show that *f* runs in polynomial time
   c. Show that a solution exists to the NP-Complete problem IFF a solution exists *to the NEW problem generate by f*

Given a graph G, does the graph contain a clique containing exactly half the vertices?

19

HALF-CLIQUE

1. Show that HALF-CLIQUE is in NP
   a. Provide a verifier
   b. Show that the verifier runs in polynomial time

Verifier: A solution consists of the set of vertices in V'
- check that |V '| = |V|/2
- for all pairs of u, v ∈ V'
  - there exists an edge (u,v) ∈ E

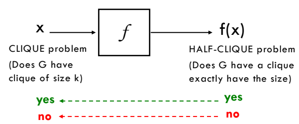
- Check for edge existence in O(V) (assuming adjacency list)
- $O(V^2)$ checks
- $O(V^3)$ overall, which is polynomial

20

HALF-CLIQUE

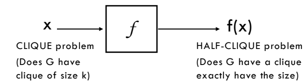
2. Show that all NP-complete problems are reducible to HALF-CLIQUE in polynomial time
   a. Describe a reduction function *f* from a known NP-Complete problem to HALF-CLIQUE
   b. Show that *f* runs in polynomial time
   c. Show that a solution exists to the NP-Complete problem IFF a solution exists *to the HALF-CLIQUE problem generate by f*

Reduce CLIQUE to HALF-CLIQUE:
Given a problem instance of CLIQUE, turn it into a problem instance of HALF-CLIQUE



21

HALF-CLIQUE



Three cases:
1. k = |V|/2

2. k < |V|/2

3. k > |V|/2

22

HALF-CLIQUE

Reduce CLIQUE to HALF-CLIQUE:
Given an instance of CLIQUE, turn it into an instance of HALF-CLIQUE

It's already a half-clique problem

$f(G, k)$

```
if ⌈|V|⌉/2 = k
        return G
elseif k < ⌈|V|⌉/2
        return G plus (|V| − 2k) nodes which are fully connected
        and are connected to every node in V
else
        return G plus 2k − |V| nodes which have no edges
```

23

HALF-CLIQUE

Reduce CLIQUE to HALF-CLIQUE:
Given an instance of CLIQUE, turn it into an instance of HALF-CLIQUE

We're looking for a clique that is smaller than half, so add an artificial clique to the graph and connect it up to all vertices

$f(G, k)$

```
if ⌈|V|⌉/2 = k
        return G
elseif k < ⌈|V|⌉/2
        return G plus (|V| − 2k) nodes which are fully connected
        and are connected to every node in V
else
        return G plus 2k − |V| nodes which have no edges
```

24

HALF-CLIQUE

Reduce CLIQUE to HALF-CLIQUE:
Given an instance of CLIQUE, turn it into an instance of HALF-CLIQUE

We're looking for a clique that is bigger than half, so add vertices until k = |V|/2

$f(G, k)$
if $\lceil |V| \rceil/2 = k$
  return G
elseif $k < \lceil |V| \rceil/2$
  return G plus $(|V| - 2k)$ nodes which are fully connected and are connected to every node in V
else
  return G plus $2k - |V|$ nodes which have no edges

25

HALF-CLIQUE

Reduce CLIQUE to HALF-CLIQUE:
Given an instance of CLIQUE, turn it into an instance of HALF-CLIQUE

$f(G, k)$
if $\lceil |V| \rceil/2 = k$
  return G
elseif $k < \lceil |V| \rceil/2$
  return G plus $(|V| - 2k)$ nodes which are fully connected and are connected to every node in V
else
  return G plus $2k - |V|$ nodes which have no edges

Runtime: From the construction we can see that it is polynomial time

26

Reduction proof

Show that a solution exists to the NP-Complete problem IFF a solution exists *to the NEW problem generate by f*

- Assume we have an NP-Complete problem instance that has a solution, show that the NEW problem instance generated by *f* has a solution
  yes ----------------------→ yes
- Assume we have a problem instance of NEW *generated by f* that has a solution, show that we can derive a solution to the NP-Complete problem instance
  yes ←---------------------- yes

$f(G, k)$
if $\lceil |V| \rceil/2 = k$
  return G
elseif $k < \lceil |V| \rceil/2$
  return G plus $(|V| - 2k)$ nodes which are fully connected and are connected to every node in V
else
  return G plus $2k - |V|$ nodes which have no edges

27

Reduction proof

Given a graph G that has a CLIQUE of size k, show that f(G,k) has a solution to HALF-CLIQUE

If k = |V|/2:
- the graph is unmodified
- f(G,k) has a clique that is half the size

28

Reduction proof

Given a graph G that has a CLIQUE of size k, show that f(G,k) has a solution to HALF-CLIQUE

If k < |V|/2:

- we added a clique of |V|- 2k fully connected nodes
- there are |V| + |V| - 2k = 2(|V|-k) nodes in f(G)
- there is a clique in the original graph of size k
- plus our added clique of |V|-2k
- k + |V|-2k = |V|-k, which is half the size of f(G)

29

Reduction proof

Given a graph G that has a CLIQUE of size k, show that f(G,k) has a solution to HALF-CLIQUE

If k >|V|/2:

- we added 2k - |V| unconnected vertices
- f(G) contains |V| + 2k - |V| = 2k vertices
- Since the original graph had a clique of size k vertices, the new graph will have a half-clique

30

Reduction proof

Given a graph f(G) that has a CLIQUE of half the elements, show that G has a clique of size k

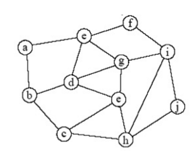
Key: f(G) was constructed by your reduction function

Use a similar argument to what we used in the other direction

31

Independent-Set

Given a graph G = (V, E) is there a subset V'⊆ V of vertices of size |V '| = k that are independent, i.e. for any pair of vertices u, v ∈ V' there exists no edge between any of these vertices

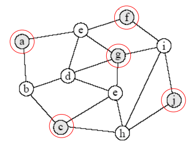


Does the graph contain an independent set of size 5?

32

Independent-Set

Given a graph G = (V, E) is there a subset V'⊆ V of vertices of size |V '| = k that are independent, i.e. for any pair of vertices u, v ∈ V' there exists no edge between any of these vertices



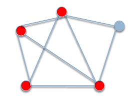
Independent-Set is NP-Complete

33

CLIQUE revisited

A *clique* in an undirected graph G = (V, E) is a subset V' ⊆ V of vertices that are fully connected, i.e. every vertex in V' is connected to every other vertex in V'

CLIQUE problem: Does G contain a clique of size k?



Is CLIQUE NP-Complete?

34

Is CLIQUE NP-Complete?

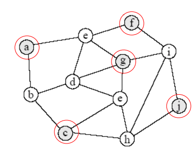
1. Show that CLIQUE is in NP
   a. Provide a verifier
   b. Show that the verifier runs in polynomial time
2. Show that all NP-complete problems are reducible to CLIQUE in polynomial time
   a. Describe a reduction function *f* from a known NP-Complete problem to CLIQUE
   b. Show that *f* runs in polynomial time
   c. Show that a solution exists to the NP-Complete problem IFF a solution exists *to the CLIQUE problem generate by f*

Given a graph G, does the graph contain a clique of size k?

35

Independent-Set

Given a graph G = (V, E) is there a subset V'⊆ V of vertices of size |V '| = k that are independent, i.e. for any pair of vertices u, v ∈ V' there exists no edge between any of these vertices. Is there an independent set of size k?



Reduce Independent-Set to CLIQUE

36

Independent-Set to Clique

Given a graph G = (V, E) is there a subset V'⊆ V of vertices of size |V '| = k that are independent, i.e. for any pair of vertices u, v ∈ V' there exists no edge between any of these vertices

Both are selecting vertices

Independent set wants vertices where NONE are connected

Clique wants vertices where ALL are connected

How can we convert a NONE problem to an ALL problem?

37

Independent-Set to Clique

Given a graph G = (V, E), the complement of that graph G' = (V, E) is the graph constructed by remove all edges E and including all edges not in E

For example, for adjacency matrix this is flipping all of the bits

```
f(G)
    return G'
```

38

Reduction proof

Show that a solution exists to the NP-Complete problem IFF a solution exists *to the NEW problem generate by f*

- Assume we have an Independent-Set problem instance that has a solution, show that the Clique problem instance generated by *f* has a solution

  yes ----------------------→ yes

- Assume we have a problem instance of Clique *generated by f* that has a solution, show that we can derive a solution to Independent-Set problem instance

  yes ←---------------------- yes

```
f(G)
    return G'
```

39

Proof

Given a graph G that has an independent set of size k, show that f(G) has a clique of size k

- By definition, the independent set has no edges between any vertices
- These will all be edges in f(G) and therefore they will form a clique of size k

40

Proof

Given f(G) that has clique of size k, show that G has an independent set of size k

- By definition, the clique will have an edge between every vertex
- None of these vertices will therefore be connected in G, so we have an independent set

41

Concrete example

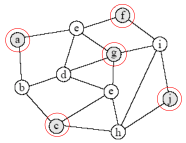
In class is slightly different than what you'd write

I've provided a concrete example of a NP completeness proof on the course webpage

42

Independent-Set revisited

Given a graph G = (V, E) is there a subset V'⊆ V of vertices of size |V '| = k that are independent, i.e. for any pair of vertices u, v ∈ V' there exists no edge between any of these vertices

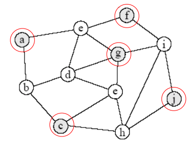


Is Independent-Set NP-Complete?

43

Independent-Set revisited

Given a graph G = (V, E) is there a subset V'⊆ V of vertices of size |V '| = k that are independent, i.e. for any pair of vertices u, v ∈ V' there exists no edge between any of these vertices



Reduce 3-SAT to Independent-Set

44

3-SAT to Independent-Set

Given a 3-CNF formula, convert it into a graph

$$(a \lor \neg b \lor c) \land (c \lor b \lor d) \land (\neg a \lor \neg c \lor \neg d)$$

For the boolean formula in 3-SAT to be satisfied, at least one of the literals in each clause must be true

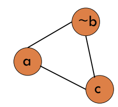
In addition, we must make sure that we enforce a literal and its complement must not both be true.

45

3-SAT to Independent-Set

Given a 3-CNF formula, convert into a graph

For each clause, e.g. *(a OR ~b OR c)* create a clique containing vertices representing these literals



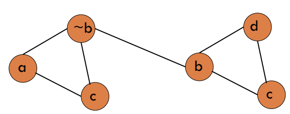
- for the Independent-Set problem to be satisfied it can only select one variable
- to make sure that all clauses are satisfied, we set k = number of clauses

46

3-SAT to Independent-Set

Given a 3-CNF formula, convert into a graph

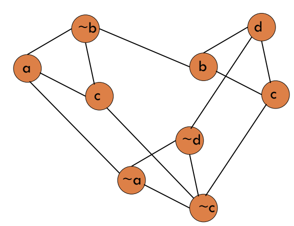
To enforce that only one variable and its complement can be set we connect each vertex representing x to each vertex representing its complement ~x



47

3-SAT to Independent-Set

$$(a \lor \neg b \lor c) \land (c \lor b \lor d) \land (\neg a \lor \neg c \lor \neg d)$$



48

Proof

Given a 3-SAT problem with k clauses and a valid truth assignment, show that f(3-SAT) has an independent set of size k. (Assume you know the solution to the 3-SAT problem and show how to get the solution to the independent set problem)

51

Proof

Given a 3-SAT problem with k clauses and a valid truth assignment, show that f(3-SAT) has an independent set of size k. (Assume you know the solution to the 3-SAT problem and show how to get the solution to the independent set problem)

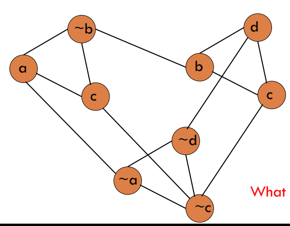
Since each clause is an OR of variables, at least one of the three must be true for the entire formula to be true. Therefore each 3-clique in the graph will have at least one node that can be selected

52

Proof

$(a \lor \neg b \lor c) \land (c \lor b \lor d) \land (\neg a \lor \neg c \lor \neg d)$

a = T
b = T
c = F
d = F



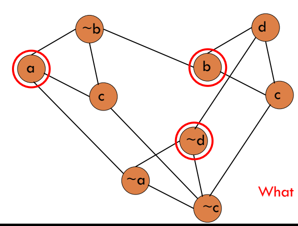
What independent set?

53

Proof

$(a \lor \neg b \lor c) \land (c \lor b \lor d) \land (\neg a \lor \neg c \lor \neg d)$

a = T
b = T
c = F
d = F



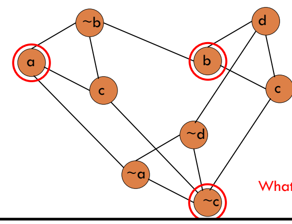
What independent set?

54

Proof

$(a \lor \neg b \lor c) \land (c \lor b \lor d) \land (\neg a \lor \neg c \lor \neg d)$

a = T
b = T
c = F
d = F



What independent set?

55

Proof

Given a graph with an independent set S of k vertices, show there exists a truth assignment satisfying the boolean formula

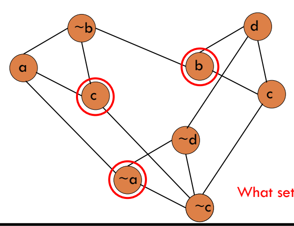
- For any variable x_i, S cannot contain both x_i and $\neg x_i$ since they are connected by an edge
- For each vertex in S, we assign it a true value and all others false. Since S has only k vertices, it must have one vertex per clause

56

Proof

$(a \lor \neg b \lor c) \land (c \lor b \lor d) \land (\neg a \lor \neg c \lor \neg d)$

a =
b =
c =
d =



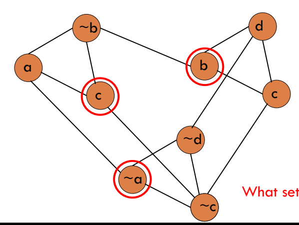
What setting of variables?

57

Proof

$(a \lor \neg b \lor c) \land (c \lor b \lor d) \land (\neg a \lor \neg c \lor \neg d)$

a = F
b = T
c = T
d = T/F



What setting of variables?

58

More NP-Complete problems

SUBSET-SUM:
- Given a set S of positive integers, is there some subset S'⊆ S whose elements sum to t.

TRAVELING-SALESMAN:
- Given a weighted graph G, does the graph contain a hamiltonian cycle of length k or less?

VERTEX-COVER:
- Given a graph G = (V, E), is there a subset V'⊆V such that if (u,v)∈E then u∈V' or v∈V'?

59

Our known NP-Complete problems

We can reduce any of these problems to a new problem in an NP-completeness proof

- SAT, 3-SAT
- CLIQUE, HALF-CLIQUE
- INDEPENDENT-SET
- HAMILTONIAN-CYCLE
- TRAVELING-SALESMAN
- VERTEX-COVER
- SUBSET-SUM

60

Search vs. Exists

All the problems we've looked at asked decision questions:
- Is there a hamiltonian cycle?
- Does the graph have a clique of size k?
- Does the graph has an independent set of size k?
- …

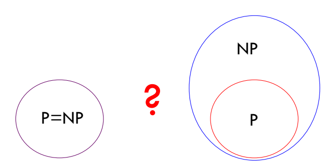
For many of the problems with a k in them, we really want to know what the largest/smallest one is
- What is the largest clique in the graph?
- What is the shortest path that visits all the vertices exactly once?

Why don't we care?

61

P vs. NP

The big question:



Someone finds a polynomial time solution to one of the NP-Complete problems

NP-Complete problems are somehow harder and distinct

62

Solving NP-Complete problems

https://www.math.uwaterloo.ca/tsp/

https://www.math.uwaterloo.ca/tsp/world/

63

Handout

64

Is Half-Clique NP-Complete?

1. Show that Half-Clique is in NP
   a. Provide a verifier
   b. Show that the verifier runs in polynomial time

Given a graph G, does the graph contain a clique containing exactly half the vertices?

65

Is Half-Clique NP-Complete?

2. Show that all NP-complete problems are reducible to Half-Clique in polynomial time
   a. Describe a reduction function *f* from a known NP-Complete problem to Half-Clique
   b. Show that *f* runs in polynomial time
   c. Show that a solution exists to the NP-Complete problem IFF a solution exists *to the NEW problem generate by f*

Given a graph G, does the graph contain a clique containing exactly half the vertices?

66