CS 401

Polynomial Reduction

Xiaorui Sun

Computational Complexity

Goal: Classify problems according to the amount of computational resources used by the best algorithms that solve them

Here we focus on time complexity

Polynomial Time Reduction

Def $A \leq_P B$: if there is an algorithm for problem A using a 'black box' (subroutine) that solve problem B s.t.,

- Algorithm uses only a polynomial number of steps
- Makes only a polynomial number of calls to a subroutine for **B**

Example

```
Algorithm for A:
Int i=0, i'=0;
Int j=0, j'=0;
…..
i=i+j;
(computation on i, j, i', j')
…..
Int x = B(i, j)
Int y = B(i', j')
…..
(compute z based on x and y)
Return z
```

Polynomial Time Reduction

Def $A \leq_P B$: if there is an algorithm for problem A using a 'black box' (subroutine) that solve problem B s.t.,

- Algorithm uses only a polynomial number of steps
- Makes only a polynomial number of calls to a subroutine for **B**

Question: Is the following polynomial time reduction correct?

Interval Scheduling $\leq_P$ Max Independent Set

- Yes. Without the blackbox of max independent set, we still have a polynomial time algorithm for interval scheduling.
- If problem A can be solved in polynomial time, then $A \leq_P B$ holds for any problem B

Polynomial Time Reduction

Def A $\leq_P$ B: if there is an algorithm for problem A using a 'black box' (subroutine) that solve problem B s.t.,

- Algorithm uses only a polynomial number of steps
- Makes only a polynomial number of calls to a subroutine for **B**

So,

B is Polynomial time solvable $\Rightarrow$ A is Polynomial time solvable

Conversely,

No efficient Algorithm for A $\Rightarrow$ No efficient Algorithm for B

In words,

- Problem A is polynomial-time reducible to problem B
- B is as hard as A (it can be even harder)
- Informally, A is a special case of B

Polynomial Time Reduction

Purpose. Classify problems according to relative difficulty.

Design algorithms. If $A \leq_P B$ and B can be solved in polynomial-time, then A can also be solved in polynomial time.

Establish intractability. If $A \leq_P B$ and A cannot be solved in polynomial-time, then B cannot be solved in polynomial time.

Establish equivalence. If $A \leq_P B$ and $B \leq_P A$, we use notation $A \equiv_P B$.

up to cost of reduction

Polynomial Time Reduction

Basic reduction strategies

- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

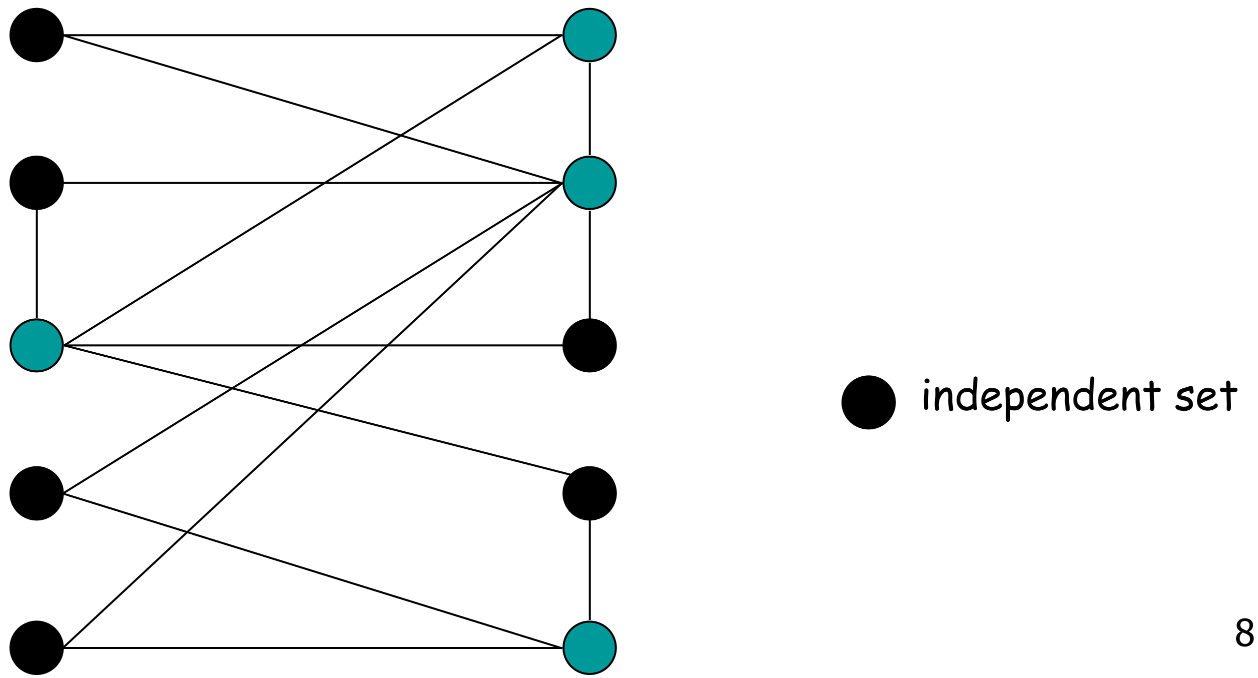
More advanced technique, read KT 8.2

Example 1: Vertex Cover $\equiv_p$ Indep Set

INDEPENDENT SET: Given a graph G = (V, E) and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \geq k$, and for each edge at most one of its endpoints is in S?

Ex. Is there an independent set of size ≥ 6? Yes.

Ex. Is there an independent set of size ≥ 7? No.

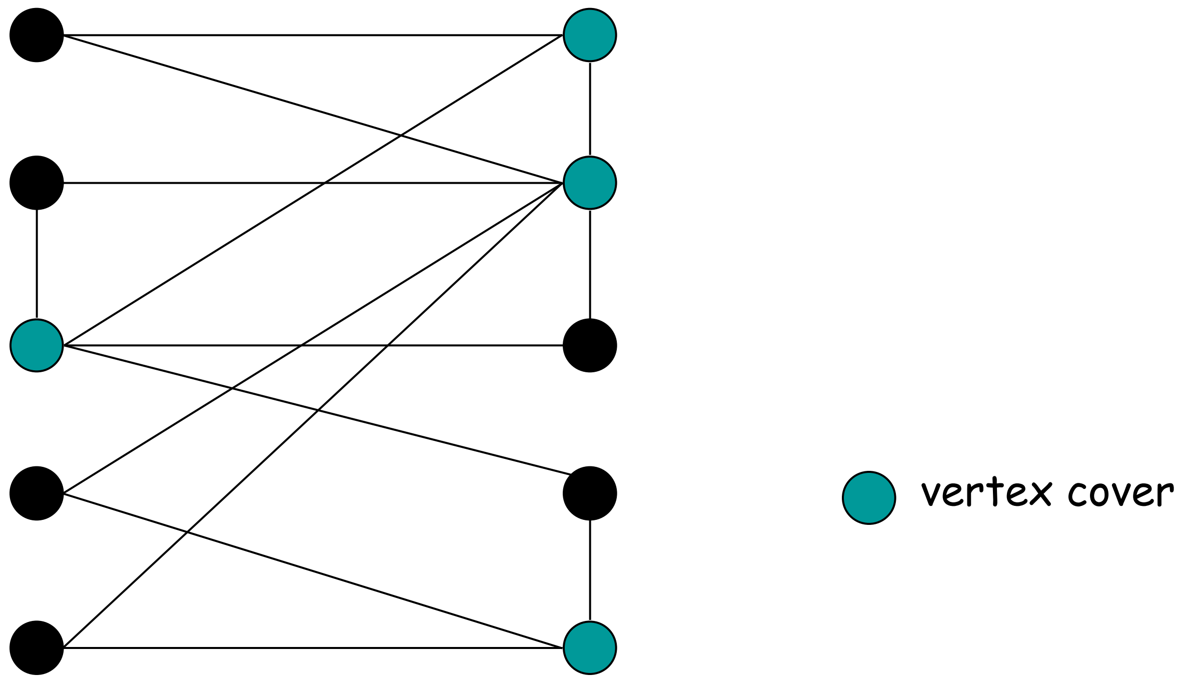


Example 1: Vertex Cover $\equiv_p$ Indep Set

VERTEX COVER: Given a graph G = (V, E) and an integer k, is there a subset of vertices S $\subseteq$ V such that |S| $\leq$ k, and for each edge, at least one of its endpoints is in S?

Ex. Is there a vertex cover of size $\leq$ 4? Yes.

Ex. Is there a vertex cover of size $\leq$ 3? No.



Example 1: Vertex Cover $\equiv_p$ Indep Set

Claim: For any graph $G = (V, E)$, S is an independent set iff $V - S$ is a vertex cover

Pf: =>

Let S be a independent set of G

Then, S has at most one endpoint of every edge of G

So, $V - S$ has at least one endpoint of every edge of G

So, $V - S$ is a vertex cover.

<= Suppose $V - S$ is a vertex cover

Then, there is no edge between vertices of S (otherwise, $V - S$ is not a vertex cover)

So, S is an independent set.

Polynomial Time Reduction

Basic reduction strategies

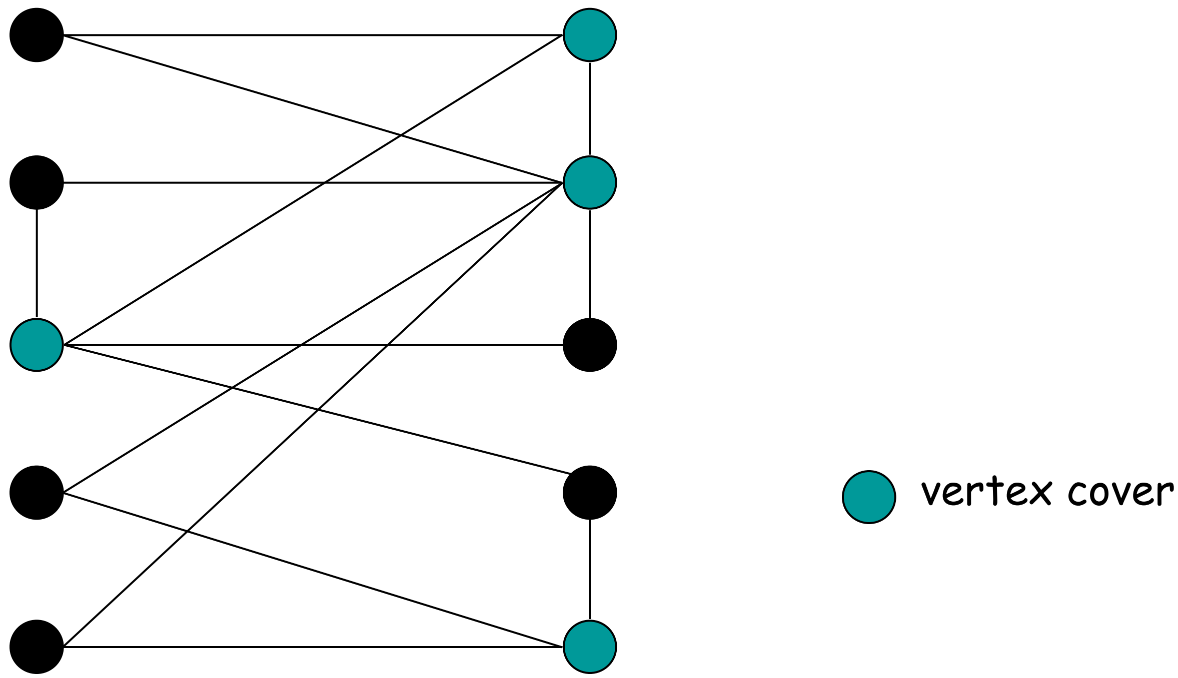
- Reduction by simple equivalence.
- Reduction from special case to general case.
- Reduction by encoding with gadgets.

Example 2: Vertex Cover $\leq_p$ Set Cover

VERTEX COVER: Given a graph G = (V, E) and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge, at least one of its endpoints is in S?

Ex. Is there a vertex cover of size ≤ 4? Yes.

Ex. Is there a vertex cover of size ≤ 3? No.



Example 2: Vertex Cover $\leq_p$ Set Cover

SET COVER: Given a set U of elements, a collection S_1, S_2, . . . , S_m of subsets of U, and an integer k, does there exist a collection of $\leq$ k of these sets whose union is equal to U?

Ex:

U = { 1, 2, 3, 4, 5, 6, 7 }
k = 2

S_1 = {3, 7}	S_4 = {2, 4}
S_2 = {3, 4, 5, 6}	S_5 = {5}
S_3 = {1}	S_6 = {1, 2, 6, 7}

Example 2: Vertex Cover $\leq_p$ Set Cover

Claim: VERTEX-COVER $\leq_P$ SET-COVER.

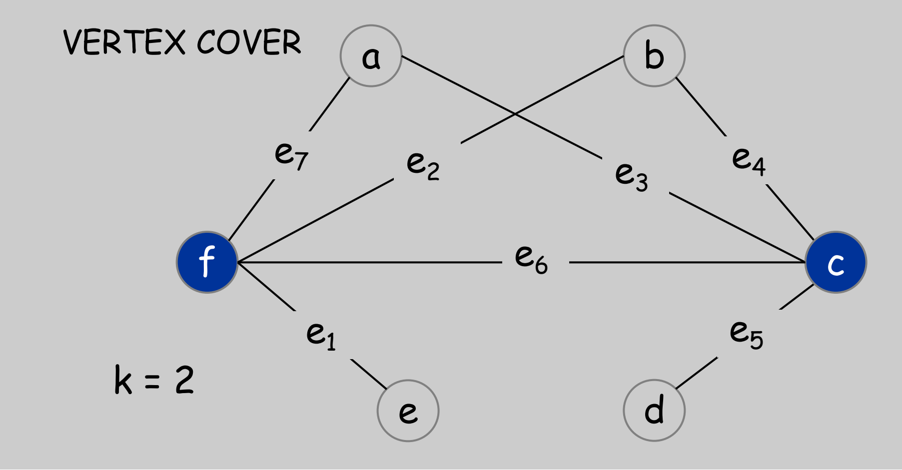
Pf: Given a VERTEX-COVER instance G = (V, E), k, we construct a set cover instance whose size equals the size of the vertex cover instance.

Construction:

Create SET-COVER instance:

- k = k, U = E, S_v = {e $\in$ E : e incident to v }

Set-cover of size $\leq$ k iff vertex cover of size $\leq$ k. ▪



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}