Integer Programming

Instance: A set v of integer variables, a set of inequalities over these variables, a function $f(v)$ to maximize, and integer B.

Question: Does there exist an assignment of integers to v such that all inequalities are true and $f(v) \geq B$?

Example:

$v_1 \geq 1, \ v_2 \geq 0$

v1 + v2 ≤ 3

$f(v) = 2v_2 \ ; \ B = 3$

Is Integer Programming in NP?

Theorem: Integer Programming is in NP

Given a set of values for each variable, we can verify that all inequalities are satisfies.

Is Integer Programming NP-Hard?

Theorem: Integer Programming is NP-Hard

Proof: By reduction from Satisfiability

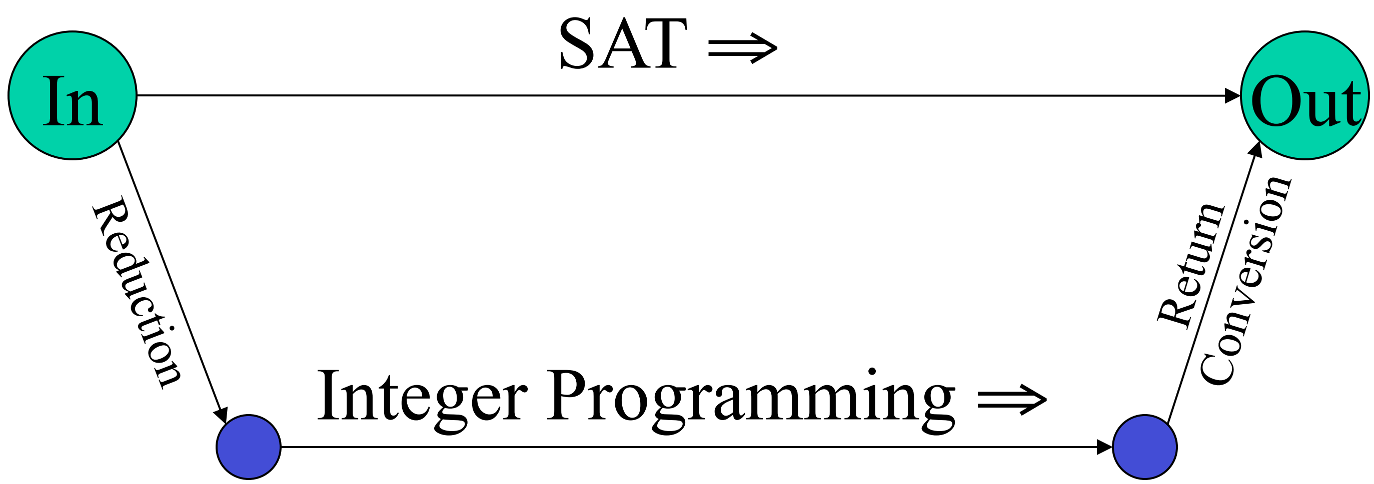
Any SAT instance has boolean variables and clauses. Our Integer programming problem will have twice as many variables, one for each variable and its compliment, as well as the following inequalities:

$0 \leq v_i \leq 1$ and $0 \leq \neg v_i \leq 1$

$1 \leq v_i + \neg v_i \leq 1$

for each clause C = $\{v_1, \neg v_2, \ldots v_i\}$: $v_1 + \neg v_2 + \ldots + v_i \geq 1$

Reducing to Integer Programming



We must show that:

1. Any SAT problem has a solution in IP.

In any SAT solution, a TRUE literal corresponds to a 1 in IP since, if the expression is SATISFIED, at least one literal per clause is TRUE, so the inequality sum is $>$ 1.

2. Any IP solution gives a SAT solution.

Given a solution to this IP instance, all variables will be 0 or 1. Set the literals corresponding to 1 as TRUE and 0 as FALSE. No boolean variable and its complement will both be true, so it is a legal assignment with also must satisfy the clauses.

Things to Notice

1. The reduction preserved the structure of the problem. Note that reducing the problem did not *solve* it - it just put the problem into a different format.

2. The IP instances that can result are a small subset of possible IP instances, but since some of them are hard, the problem in general must be hard.

More Things to Notice

3. The transformation captures the essence of why IP is hard - it has nothing to do with big coefficients or big ranges on variables; restricting to 0/1 is enough. A reduction tells us a lot about a problem.

4. It is not obvious that IP is in NP, since the numbers assigned to the variables may be too large to write in polynomial time - don't be too hasty! Couldn't maximizing a function could drive some unbounded variables to extreme values?

The Independent Set Problem

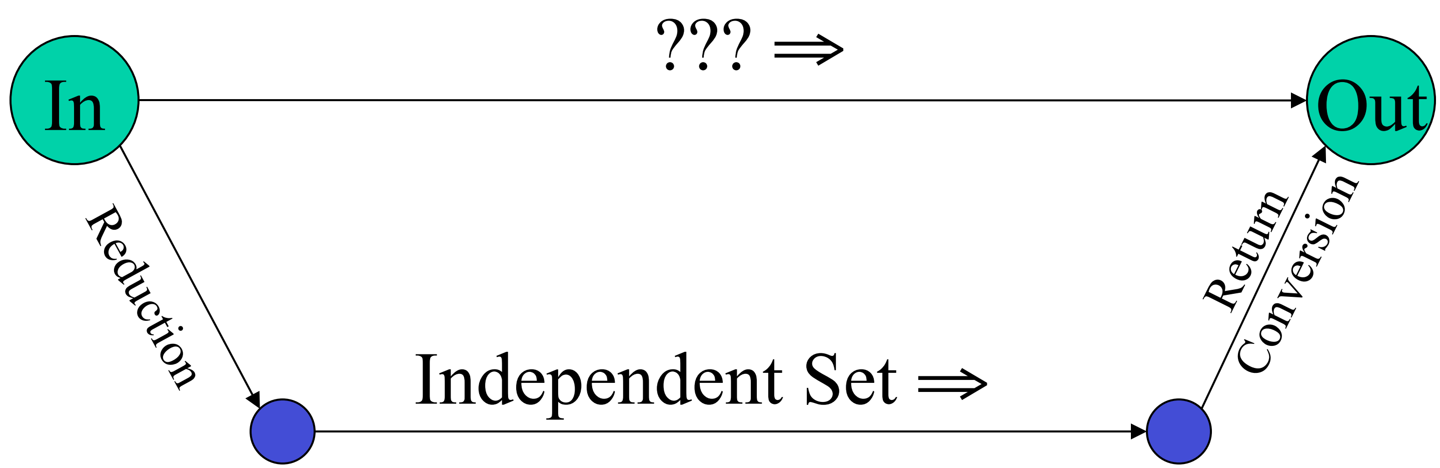
Problem: Given a graph $G = (V, E)$ and an integer k, is there a subset S of at least k vertices such that no $e \in E$ connects two vertices that are both in S ?

Theorem: Independent Set is NP-complete.

Proof: Independent Set is in NP - given any subset of vertices, we can count them, and show that no vertices are connected.

How can we prove that it is also a hard problem?

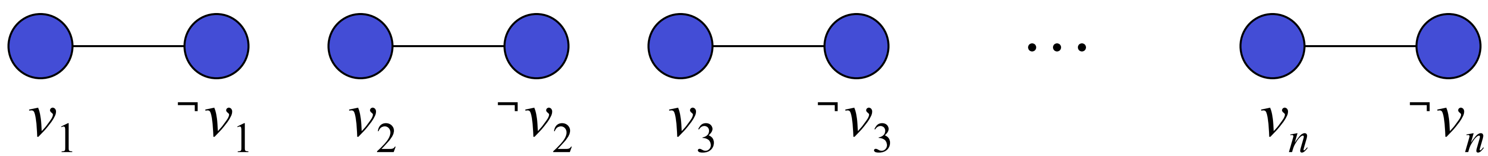
Reducing to Independent Set



What problem should we choose to reduce?

Reducing 3-SAT to Independent Set

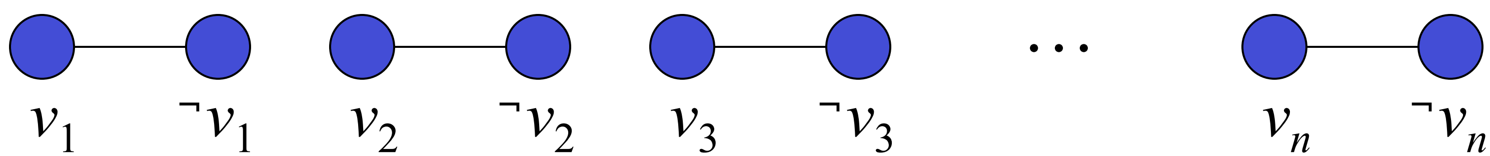
For each variable, we can create two vertices:



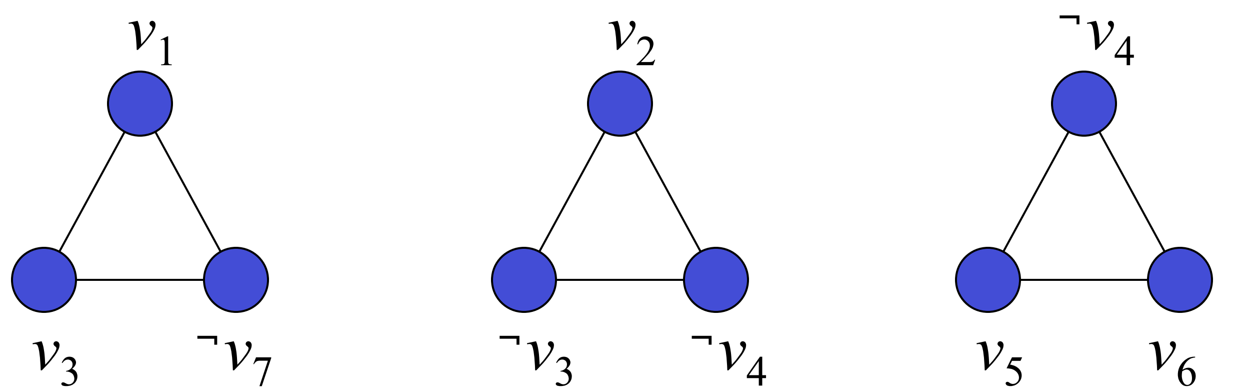
If we connect a variable and its negation, we can be sure that only *one* of them is in the set. In all, we must have n vertices in S to be sure all variables are assigned.

This will handle the binary true-false values; how can we also make sure that all of the clauses are fulfilled?

Including *Clauses* in the Reduction



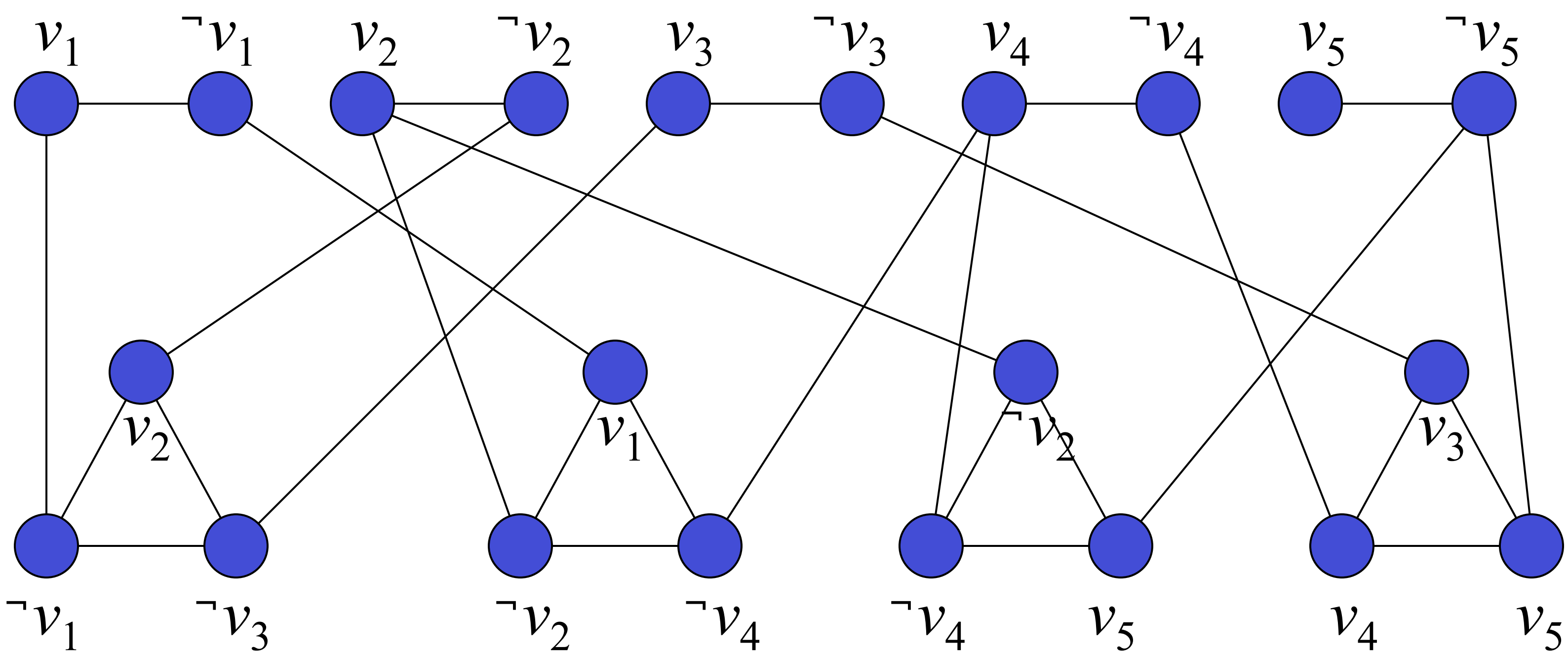
We can consider the clauses as triangles:



Each clause has at least one true value. On the other hand, at most one vertex in a triangle can be in the independent set. So how do we tie these together?

Tying it all together...

$$C = \{\neg v_1, v_2, \neg v_3\} \;,\; \{v_1, \neg v_2, \neg v_4\} \;,$$
$$\{\neg v_2, \neg v_4, v_5\} \;,\; \{v_3, v_4, v_5\}$$



Minimum Vertex Cover: Decision Problem

Problem: Given a graph $G = (V, E)$ and an integer k, is there a subset of at most k vertices such that every $e \in E$ has at least one vertex in the subset?

Theorem: Vertex cover is NP-complete.

Proof: VC is in NP - given a subset of vertices, we can count them, and show that each edge is covered.

What problem should we choose to reduce?

Maximum Clique

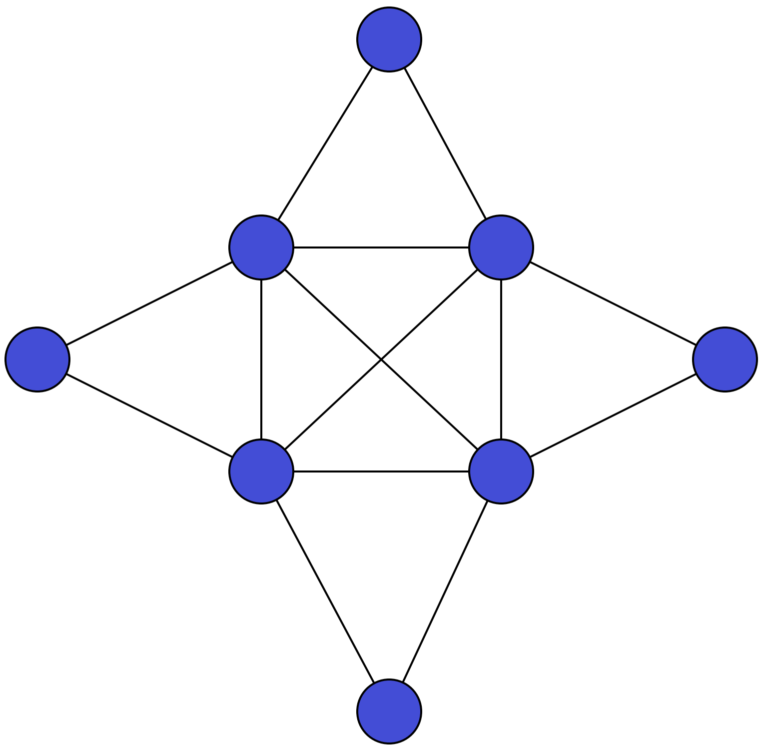
Problem: Given a graph $G = (V, E)$ and an integer k, is there a subset S of at least k vertices, such that for every $v_1, v_2 \in V$, there exists an $e \in E$ connecting v_1 and v_2?

Theorem: Maximum Clique is NP-complete.

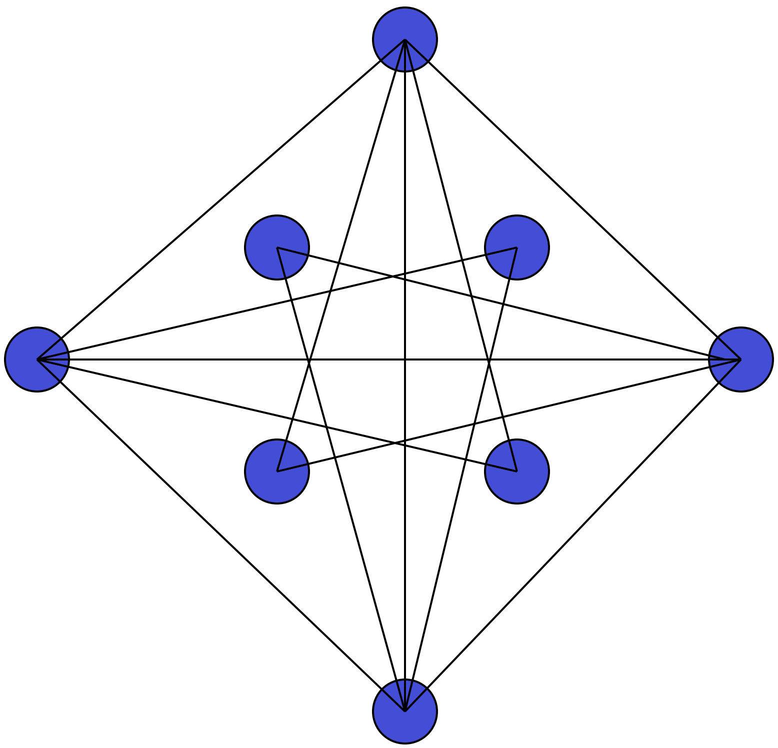
Proof: MC is in NP - given a subset of vertices, we can count them, and show that pair is connected.

What problem should we choose to reduce this time?

Transforming a Graph



G = (V, E)



G′ = (V, E′)

The Set Cover Problem

In the U.S. navy, the SEALS are each specially trained in a wide variety of skills so that small teams can handle a multitude of missions.

If there are *k* different skills needed for a mission, and *n* SEAL members that can be assigned to the team, find the smallest team that will cover all of the required skills.

Andersen knows **hand-to-hand**, **first aid**, and **camouflage**
Butler knows **hand-to-hand** and **snares**
Cunningham knows **hand-to-hand**
Douglas knows **hand-to-hand**, **sniping**, **diplomacy**, and **snares**
Eckers knows **first-aid**, **sniping**, and **diplomacy**

Minimum Set Cover

Problem: Given a set S of subsets $\{S_1, S_2, \ldots, S_m\}$ out of a universal set $U = \{u_1, u_2, \ldots, u_n\}$ and an integer k, is it possible to choose only k subsets of S such that the union of these subsets is U.

Theorem: Minimum Set Cover is NP-complete.

Proof: MSC is in NP - given a subset of sets, we can count them, and show that all elements of U are included.

What problem should we choose to reduce this time?