Aalto University
School of Science

CS-E4530 Computational Complexity Theory

Lecture 8: **NP**-Complete Problems I: Variants of Satisfiability; Packing, Covering and Partioning in Graphs

Aalto University
School of Science
Department of Computer Science

Spring 2020

Topics

- Characterising $\mathbf{NP}$
- Variants of satisfiability
- Graphs: packing, covering and partitioning

(C. Papadimitriou: *Computational Complexity*, Chapters 9.1–9-3)

9.1 Characterising $\mathbf{NP}$

- The complexity class $\mathbf{NP}$ can be characterised also without any reference to nondeterministic Turing machines.
- In light of the alternative characterisation $\mathbf{NP}$ can be seen as the class of problems having *succinct certificates*.

Definition (9.1)

1. A relation $R \subseteq \Sigma^* \times \Sigma^*$ is *polynomially decidable* if there is a deterministic TM deciding the language $\{x;y \mid (x,y) \in R\}$ in polynomial time.
2. A relation R is *polynomially balanced* if $(x,y) \in R$ implies $|y| \leq |x|^k$ for some $k \geq 1$.

Proposition (9.1)

Let $L \subseteq \Sigma^*$ be a language. Then $L \in \mathbf{NP}$ iff there is a polynomially balanced and polynomially decidable relation R such that $L = \{x \in \Sigma^* \mid (x,y) \in R$ for some $y \in \Sigma^*\}$.

Proof

($\Leftarrow$) Assume there is such a relation R. Then L is decided by a NTM that on input x (i) guesses a y of length at most $|x|^k$ and (ii) uses the machine for R to decide in polynomial time whether $(x,y) \in R$.

($\Rightarrow$) Assume that $L \in \mathbf{NP}$, i.e. there is a NTM N deciding L in time $|x|^k$ for some k.

Define a relation R as follows: $(x,y) \in R$ iff y is the encoding of an accepting computation of N on input x.

Now R is polynomially

- balanced (each computation of N is polynomially bounded) and
- decidable (since it can be checked in linear time whether y encodes an accepting computation of N on x).

As N decides L, $L = \{x \mid (x,y) \in R$ for some $y\}$.

Succinct certificates

A problem is in $\mathbf{NP}$ if any "yes" instance x of the problem has at least one *succinct certificate*, or polynomial witness, y. $\mathbf{NP}$ contains a huge number of practically important, natural computational problems:

- A typical problem is to construct a mathematical object satisfying certain specifications (path, solution of equations, routing, VLSI layout,...). This is the certificate.
- The decision version of the problem is to determine whether at least one such object exists for the input. Basic requirements:
  - The object is not very large compared to the input.
  - The specifications of the object are simple enough so that they can be checked in polynomial time.

Boundary between $\mathbf{NP}$ and $\mathbf{P}$

- Most problems arising in computational practice are in $\mathbf{NP}$.
- Computational complexity theory provides tools to study which problems in $\mathbf{NP}$ belong to $\mathbf{P}$ and which (probably) do not.
- $\mathbf{NP}$-completeness is a basic tool in this respect:
  Showing that a problem is $\mathbf{NP}$-complete implies that the problem is among the least likely to be in $\mathbf{P}$.
  (If an $\mathbf{NP}$-complete problem is in $\mathbf{P}$, then $\mathbf{NP} = \mathbf{P}$.)
- A quote from Papadimitriou (p. 183):
  "There is nothing wrong with trying to prove that $\mathbf{P} = \mathbf{NP}$ by developing a polynomial-time algorithm for an $\mathbf{NP}$-complete problem.
  The point is that without an $\mathbf{NP}$-completeness proof we would be trying the same thing *without knowing it!*"

$\mathbf{NP}$-completeness and algorithm design techniques

When a problem is known to be $\mathbf{NP}$-complete, further efforts can be directed to:

(i) Polynomially solvable special cases

(ii) Approximation algorithms

(iii) Average case performance

(iv) Randomised algorithms

(v) (Exponential) algorithms that are practical for small instances

(vi) Local search methods

9.2 Variants of Satisfiability

- $\mathbf{NP}$-complete problems typically remain difficult even when restricted to quite simple inputs. But often they also have interesting special cases in $\mathbf{P}$.
- Hence it is relevant to find the borderlines, for a given problem, between subproblems that are in $\mathbf{P}$ and that are $\mathbf{NP}$-complete.
- A basic technique to find difficult subproblems of a given $\mathbf{NP}$-complete problem A is to look at the set of instances produced by a reduction R from some other $\mathbf{NP}$-complete problem B to A.
- Next we consider variants of SAT such as

  $$3\text{SAT},\ 2\text{SAT},\ \text{MAX2SAT, and NAESAT}$$

  and analyse their computational complexities.

kSAT problems

Definition (9.2)

kSAT, where $k \geq 1$ is an integer, is the set of Boolean expressions $\phi \in$ SAT (in CNF) whose every clause has exactly k literals.

Proposition (9.2)

3SAT is $\mathbf{NP}$*-complete.*

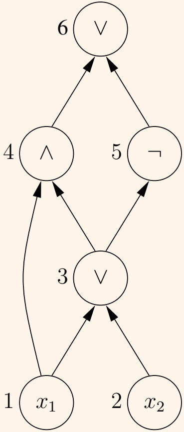
Proof

- 3SAT is in $\mathbf{NP}$ as a special case of SAT which is in $\mathbf{NP}$.
- CIRCUIT SAT was shown to be $\mathbf{NP}$-complete and a reduction from CIRCUIT SAT to SAT has already been presented.
- Consider now the clauses in the reduction. They have all at most 3 literals. Each clause with one or two literals can be modified to an equivalent clause with exactly 3 literals by duplicating literals.
- Hence, we can reduce CIRCUIT SAT to 3SAT.

Example

3SAT is **NP**-complete: reduction from CIRCUIT SAT to 3SAT

Circuit C:



The corresponding CNF formula $R(C)$:

$(g_6 \vee g_6 \vee g_6) \wedge$

$(\neg g_6 \vee g_4 \vee g_5) \wedge (g_6 \vee \neg g_4 \vee \neg g_4) \wedge (g_6 \vee \neg g_5 \vee \neg g_5) \wedge$

$(\neg g_5 \vee \neg g_3 \vee \neg g_3) \wedge (g_5 \vee g_3 \vee g_3)$

$(g_4 \vee \neg g_1 \vee \neg g_3) \wedge (\neg g_4 \vee g_1 \vee g_1) \wedge (\neg g_4 \vee g_3 \vee g_3) \wedge$

$(\neg g_3 \vee g_1 \vee g_2) \wedge (g_3 \vee \neg g_1 \vee \neg g_1) \wedge (g_3 \vee \neg g_2 \vee \neg g_2) \wedge$

$(g_2 \vee \neg x_2 \vee \neg x_2) \wedge (\neg g_2 \vee x_2 \vee x_2) \wedge$

$(g_1 \vee \neg x_1 \vee \neg x_1) \wedge (\neg g_1 \vee x_1 \vee x_1)$

Narrowing $\mathbf{NP}$-complete languages

- An $\mathbf{NP}$-complete languages can sometimes be narrowed down by *transformations* which eliminate certain features of the language but still preserve $\mathbf{NP}$-completeness.
- The following result is a typical example.

Proposition (9.3)

3SAT remains $\mathbf{NP}$-complete even when constrained to Boolean expressions ϕ in which each variable appears at most three times and each literal at most twice.

Proof

This is shown by a reduction where any instance ϕ of 3SAT is rewritten to eliminate the forbidden features.

- Consider a variable x appearing $k \geq 3$ times in ϕ.
  - (i) Replace the first occurrence of x in ϕ by x_1, the second by x_2, and so on where $x_1, \ldots, x_k$ are new variables.
  - (ii) Add clauses $(\neg x_1 \vee x_2), (\neg x_2 \vee x_3), \ldots, (\neg x_k \vee x_1)$ to ϕ.
- Let ϕ' be the expression ϕ modified systematically in this way.
- Clearly ϕ' has the desired syntactic properties.
- Also ϕ is satisfiable iff ϕ' is satisfiable:
  For each x appearing $k > 3$ times in ϕ, the truth values of $x_1, \ldots, x_k$ are the same in each truth assignment satisfying ϕ'.

Example

Original CNF formula ϕ for 3SAT:

$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge$
$(\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (x_1 \vee x_3 \vee \neg x_4)$

The "sparse" CNF formula $R(\phi)$:

$(x_{1,1} \vee x_2 \vee x_{3,1}) \wedge (x_{1,2} \vee x_{3,2} \vee x_4) \wedge$
$(\neg x_{1,3} \vee \neg x_2 \vee \neg x_4) \wedge (x_{1,4} \vee x_{3,3} \vee \neg x_4)$
$(\neg x_{1,1} \vee x_{1,2}) \wedge (\neg x_{1,2} \vee x_{1,3}) \wedge$
$(\neg x_{1,3} \vee x_{1,4}) \wedge (\neg x_{1,4} \vee x_{1,1}) \wedge$
$(\neg x_{3,1} \vee x_{3,2}) \wedge (\neg x_{3,2} \vee x_{3,3}) \wedge$
$(\neg x_{3,3} \vee x_{3,1})$

Boundary between $\mathbf{P}$ and $\mathbf{NP}$-completeness

- The boundary is between 2SAT and 3SAT.
- Every instance ϕ of 2SAT can be decided by a polynomial time algorithm, based on reachability in a graph associated with ϕ.

Definition (9.4)

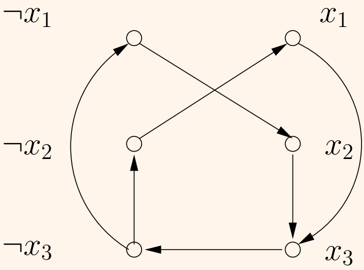
Let ϕ be an instance of 2SAT.
Define a graph $G(\phi)$ as follows:
– The vertices of $G(\phi)$ correspond to the variables of ϕ and their negations.
– For every clause $\alpha \vee \beta$ in ϕ, there are arcs $(\overline{\alpha}, \beta)$ and $(\overline{\beta}, \alpha)$ in $G(\phi)$.

Theorem (9.4)

Let ϕ be an instance of 2SAT.
Then ϕ is unsatisfiable iff there is a variable x such that there are paths from x to $\neg x$ and from $\neg x$ to x in $G(\phi)$.

Example

- Consider the formula
  $\phi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_3 \vee \neg x_3)$
- The graph $G(\phi)$:



- ϕ is unsatisfiable as there is a path from x_3 to $\neg x_3$ and from $\neg x_3$ to x_3 in $G(\phi)$.

The complexity of 2SAT—cont'd

Corollary (9.5)

2SAT is in $\mathbf{NL}$ $(\subseteq \mathbf{P})$.

Proof

Since $\mathbf{NL}$ is closed under complement, it is sufficient to show that 2SAT COMPLEMENT is in $\mathbf{NL}$.

The reachability condition of the preceding theorem can be tested in logarithmic space non-deterministically by guessing a variable x and paths from x to $\neg x$ and back.

The complexity of 2SAT—cont'd

MAX2SAT is a generalisation of 2SAT:

INSTANCE: a Boolean expression ϕ in CNF (having at most two literals per clause) and an integer bound K.

QUESTION: Is there a truth assignment satisfying at least K clauses?

Theorem (9.6)

MAX2SAT is $\mathbf{NP}$-complete.

(Proof on pp. 186–187 in the Papadimitriou book.)

Not-all-equal SAT (NAESAT)

The set NAESAT $\subseteq$ 3SAT comprises those instances $\phi \in$ 3SAT where for some truth assignment the three literals in each clause of ϕ do not have the same truth value. (I.e. satisfaction by three **true**'s is not accepted; and of course not by three **false**'s either.)

Theorem (9.7)

NAESAT is $\mathbf{NP}$*-complete.*

Proof

- CIRCUIT SAT was shown to be $\mathbf{NP}$-complete and a reduction R from CIRCUIT SAT to SAT has already been presented such that for a circuit C, $C \in$ CIRCUIT SAT iff $R(C) \in$ SAT.
- For all one- and two-literal clauses in the resulting set of clauses $R(C)$, add the same literal, say z, to make them 3-literal clauses.

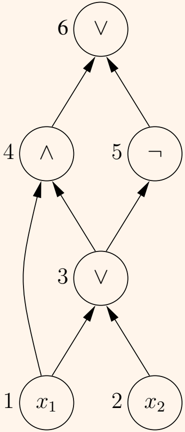
Claim: For the resulting Boolean expression $R_z(C)$ in 3CNF it holds:

$$C \in \text{CIRCUIT SAT } \textit{iff } R_z(C) \in \text{NAESAT}.$$

Example

Reduction from CIRCUIT SAT to NAESAT:

Circuit C:



The corresponding expression $R_z(C)$:

$(g_6 \lor z \lor z) \land$
$(\neg g_6 \lor g_4 \lor g_5) \land (g_6 \lor \neg g_4 \lor z) \land (g_6 \lor \neg g_5 \lor z) \land$
$(\neg g_5 \lor \neg g_3 \lor z) \land (g_5 \lor g_3 \lor z)$
$(g_4 \lor \neg g_1 \lor \neg g_3) \land (\neg g_4 \lor g_1 \lor z) \land (\neg g_4 \lor g_3 \lor z) \land$
$(\neg g_3 \lor g_1 \lor g_2) \land (g_3 \lor \neg g_1 \lor z) \land (g_3 \lor \neg g_2 \lor z) \land$
$(g_2 \lor \neg x_2 \lor z) \land (\neg g_2 \lor x_2 \lor z) \land$
$(g_1 \lor \neg x_1 \lor z) \land (\neg g_1 \lor x_1 \lor z)$

Proof (cont'd)

($\Rightarrow$) If C is satisfiable, then there is a truth assignment T satisfying $R(C)$. Let us then extend T for $R_z(C)$ by assigning $T(z) = \mathbf{false}$. Let us check that in no clause of $R_z(C)$ are all literals **true** in the extended assignment T. (They also cannot be all **false**.)

(i) Clauses for **true**/**false**/NOT/variable gates contain z that is **false**.

(ii) For AND gates the clauses are: $(\neg g \vee h \vee z)$, $(\neg g \vee h' \vee z)$, $(g \vee \neg h \vee \neg h')$. In the first two z is **false**, and in the third not all three literals can be **true**, as then the first two clauses would not be satisfied.

(iii) The case of OR gates is similar.

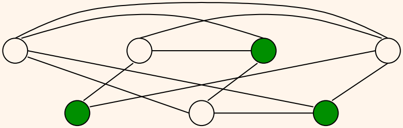
($\Leftarrow$) If a truth assignment T satisfies $R_z(C)$ in the sense of NAESAT, so does the complementary truth assignment $\overline{T}$. Since z is **false** in either T or $\overline{T}$, it follows that the unadorned expression $R(C)$ is satisfied by T or $\overline{T}$. Thus, C is satisfiable. $\square$

9.3 Graphs: Packing, Covering and Partitioning

- In this section, we will consider only undirected graphs $G = (V, E)$ and their properties.
- For instance, consider the problem of finding an *independent* subset I of V, i.e., a set $I \subseteq V$ such that for all $i, j \in I$, $\{i, j\} \notin E$.

Example

A graph with an independent set of size 3:



Definition (9.5)

INDEPENDENT SET:

INSTANCE: An undirected graph $G = (V, E)$ and an integer K.

QUESTION: Is there an independent set $I \subseteq V$ with $|I| = K$?

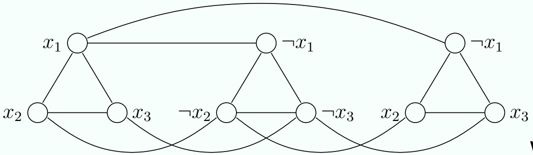
Theorem (9.8)

INDEPENDENT SET is $\mathbf{NP}$*-complete.*

Proof

Reduction from 3SAT, see Papadimitriou pp. 188–190 for details.

Example: $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$ is reduced to



with $K = 3$.

The subclass of graphs used in the reduction implies the following:

Corollary (9.9)

4-DEGREE INDEPENDENT SET is $\mathbf{NP}$*-complete.*

Graph problems: CLIQUE and VERTEX COVER

- The problems in graph theory are often closely related, in ways which suggest almost trivial reductions between problems.
- We now show that independent sets are closely related to cliques and vertex covers.

Definition (9.6)

CLIQUE:

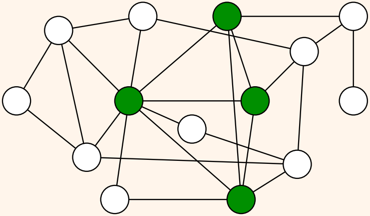
INSTANCE: An undirected graph $G = (V, E)$ and an integer K.

QUESTION: Is there a clique $C \subseteq V$ with $|C| = K$?

(A set $C \subseteq V$ is a clique iff for any two vertices $i, j \in C$, $\{i, j\} \in E$.)

Example

A graph with a clique of size 4:



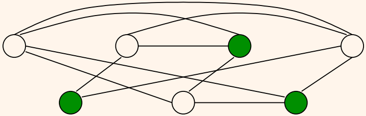
Observation: A set $I \subseteq V$ of vertices is an independent set of $G = (V, E)$ iff it is a clique of the *complement* graph $\overline{G} = (V, (V \times V) - E)$.

This leads to a trivial reduction from INDEPENDENT SET to CLIQUE:

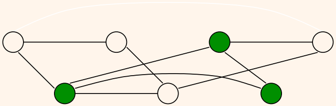
Example

Reduction from INDEPENDENT SET to CLIQUE illustrated:

$G; K$



$R(G;K) = \bar{G}; K$



Theorem (9.10)

CLIQUE is $\mathbf{NP}$*-complete.*

Definition (9.7)

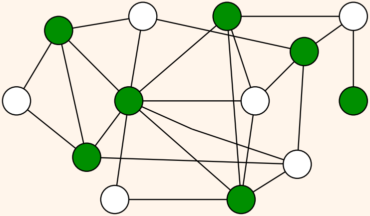
VERTEX COVER:

INSTANCE: An undirected graph $G = (V,E)$ and an integer B.

QUESTION: Is there a set $C \subseteq V$ with $|C| \leq B$ such that for every $\{i,j\} \in E$, either $i \in C$ or $j \in C$ (or both)?

Example

A graph with a vertex cover of size 7:



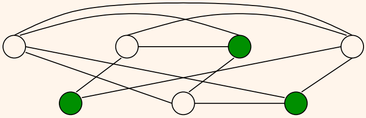
Observation: A set $I \subseteq V$ of vertices is an independent set of G iff $V - I$ is a vertex cover of G.

This leads to a simple reduction from INDEPENDENT SET to CLIQUE:

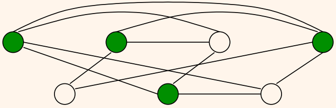
Example

Reduction from INDEPENDENT SET to VERTEX COVER illustrated:

$G; K$



$R(G; K) = G; |V| - |K|$



Theorem (9.11)

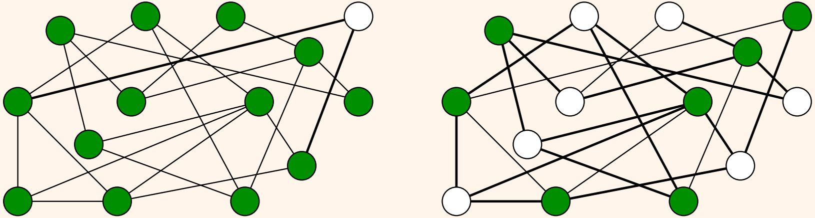
VERTEX COVER is $\mathbf{NP}$*-complete.*

Graph problems: MIN CUT and MAX CUT

- A *cut* in an undirected graph $G = (V,E)$ is a partition of the vertices into two nonempty sets S and $V - S$.
- The *size of a cut* is the number of edges between S and $V - S$.

Example

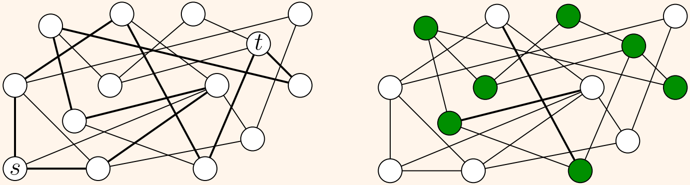
A graph and two cuts (of sizes 2 and 17, resp.):



- The problem of finding a cut with the smallest size is in $\mathbf{P}$:
  - (i) The size of the smallest cut that separates two given vertices s and t equals the maximum flow from s to t. ("Max-Flow/Min-Cut Thm".)
  - (ii) Minimum cut: find the maximum flow between a fixed s and all other vertices and choose the smallest value found.

Example

A maximum flow and cut of size 2:



- However, the problem of deciding whether there is a cut of a size at least K (MAX CUT) is much harder:

Theorem (9.12)

MAX CUT is $\mathbf{NP}$-complete.

Proof

The $\mathbf{NP}$-completeness of MAX CUT is shown for graphs with multiple edges between vertices by a reduction from NAESAT.

- For a conjunction of clauses $\phi = C_1 \wedge \ldots \wedge C_m$, we construct a graph $G = (V, E)$ so that

  G has a cut of size $5m$ iff ϕ is satisfied in the sense of NAESAT.

- The vertices of G are $x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n$ where $x_1, \ldots, x_n$ are the variables in ϕ.
- The edges in G include a triangle $[\alpha, \beta, \gamma]$ for each clause $\alpha \vee \beta \vee \gamma$ and n_i copies of the edge $\{x_i, \neg x_i\}$ where n_i is the number of occurrences of x_i or $\neg x_i$ in the clauses.
- Now a cut $(S, V - S)$ of size $5m$ in G corresponds to a truth assignment satisfying ϕ in the sense of NAESAT.

Example

Consider the conjunction of clauses ϕ:

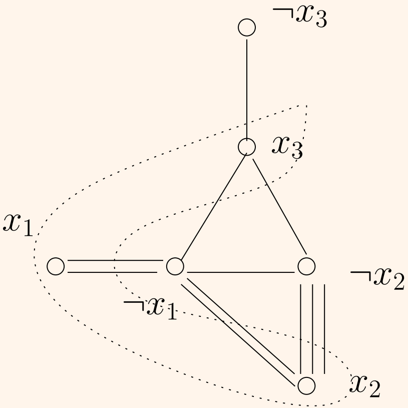
$$(\neg x_1 \vee x_2 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

This is satisfied in the sense of NAESAT iff the graph G on the right obtained as the result of the reduction has a cut of size 5*2=10.

For instance,

$$(\{x_1, x_2, x_3\}, \{\neg x_1, \neg x_2, \neg x_3\})$$

is a cut of size 10. It corresponds to a truth assignment $T(x_1) = T(x_2) = T(x_3) = \textbf{true}$ that satisfies ϕ in the sense of NAESAT.



Correctness of the reduction

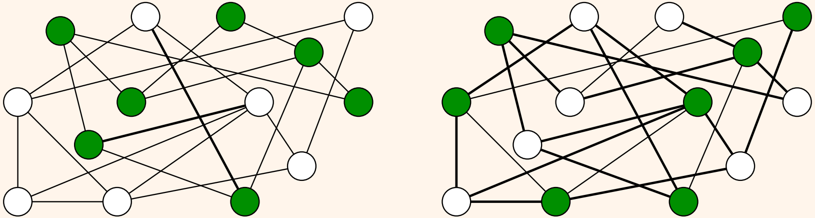
- It is easy to see that a satisfying truth assignment (in the sense of NAESAT) gives rise to a cut of size $5m$.
- Conversely, suppose there is a cut $(S, V - S)$ of size $5m$ or more.
- All variables can be assumed separate from their negations:
  If both $x_i, \neg x_i$ are on the same side, they contribute at most $2n_i$ edges to the cut (where n_i is the number of occurrences of x_i or $\neg x_i$ in the clauses).
  Hence, moving the one with fewer neighbours to the other side of the cut does not decrease the size of the cut.
- Let S be the set of true literals and $V - S$ those false.
- The total number of edges in the cut joining opposite literals is $3m$. The remaining $2m$ are coming from triangles meaning that all m triangles are cut, i.e. ϕ is satisfied in the sense of NAESAT. $\square$

Graph problems: MAX BISECTION

- In many applications of graph partitioning, the sizes of S and $V - S$ cannot be arbitrarily small or large.
- MAX BISECTION is the problem of determining whether there is a cut $(S, V - S)$ with size of K or more such that $|S| = |V - S|$.

Example

Bisections with cut sizes of 2 and 17, respectively:



Theorem (9.13)

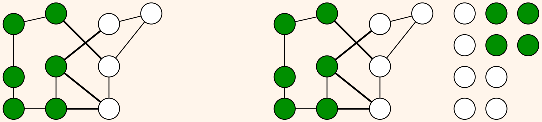
MAX BISECTION is $\mathbf{NP}$*-complete.*

Proof

MAX CUT can be reduced to MAX BISECTION by simply adding $|V|$ disconnected new vertices to G. Then every cut of G can be made a bisection by appropriately splitting the new vertices: Graph $G = (V, E)$ has a cut $(S, V - S)$ with size of K or more iff the modified graph has a cut with size of K or more with $|S| = |V - S|$.

Example

Reducing MAX CUT to MAX BISECTION:



Graph problems: BISECTION WIDTH

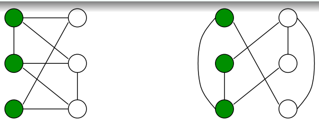
- The corresponding minimisation problem, i.e. MIN CUT with the bisection requirement, is $\mathbf{NP}$-complete, too. (Recall that MIN CUT $\in \mathbf{P}$).
- BISECTION WIDTH: is there a bisection of size K or less?

Theorem (9.14)

BISECTION WIDTH is $\mathbf{NP}$-complete.

Proof

A reduction from MAX BISECTION. A graph $G = (V, E)$ where $|V| = 2n$ for some n has a bisection of size K or more iff the complement graph $\overline{G}$ has a bisection of size $n^2 - K$ or less.



General guidelines for establishing $\mathbf{NP}$-completeness

Designing an $\mathbf{NP}$-completeness proof for a given problem Q:

- Work on small instances of Q to develop gadgets/primitives.
- Look at known $\mathbf{NP}$-complete problems.
- Design a reduction R **from** *a known* $\mathbf{NP}$*-complete problem* **to** Q.
- Typical ingredients of a reduction:

  choices + consistency + constraints.

- The key question is how to express these in Q?