NP-Complete graph problems - KombSøg 2017

Anders H Pedersen

May 2017

The graph problems and reductions below were introduced by Thomas Dueholm Hansen, at the lecture on NP-Complete graph problems may 3 2017.

1 Graph problems

1.1 INDEPENDENT SET

Given: Graph G = (V,E), target K.
Question: Does there exist $I \subseteq V$ such that $|I| \geq K$ and for all $u, v \in I$ we have $(u, v) \notin E$?

1.2 CLIQUE

Given: Graph G = (V,E), target K.
Question: Does there exist $C \subseteq V$ such that $|C| \geq K$ and for all $u, v \in C$ we have $(u, v) \in E$?

1.3 VERTEX COVER

Given: Graph G = (V,E), budget B.
Question: Does there exist $C \subseteq V$ such that $|C| \leq B$ and for all $(u, v) \in E$ we have $(u \in C \vee v \in C)$?

1.4 MAX CUT

Given: Graph G = (V,E), target K.
Question: Does there exist a cut $(S, V \setminus S)$ of size at least K?

1.5 MAX BISECTION

Given: Graph G = (V,E), target K.
Question: Does there exist a cut $(S, V \setminus S)$ of size at least K, such that $|S| = |V \setminus S|$?

1.6 BISECTION WIDTH

Given: Graph G = (V,E), target K.
Question: Does there exist a cut $(S, V \setminus S)$ of size at most K, such that $|S| = |V \setminus S|$?

1.7 HAMILTONIAN PATH

Given: Graph G = (V,E).
Question: Does G have a path that visits every vertex exactly once?

1.8 TSP

Given: Distance matrix D, target t.
Question: Is there a tour of length at most t that visits every node in the graph defined by D exactly once?

2 Reductions

As always, in a reduction $L_1 \leq L_2$, we describe a polynomial time computable function r, such that $\forall x : x \in L_1 \iff r(x) \in L_2$. We then argue that it is indeed polynomial, and that both directions of the bi-implication holds.

2.1 3SAT $\leq$ INDEPENDENT SET

Theorem 9.4: INDEPENDENT SET is NP-Complete

We prove this by reducing from 3SAT, which we know is NP-Complete.

For this reduction we need a gadget, the triangle. The logic behind this is that, if a graph contains a triangle, then at most one of the nodes can be in the independent set. We restrict the class of graphs we consider, to graphs whose nodes can be partitioned in m disjoint triangles. This ensures that an independent set can contain at most m nodes.

For each of the m clauses in our input CNF formula, we create a triangle where the nodes are labeled with the literals of the clause. Next, we add an edge between two nodes in different triangles if and only if the nodes correspond to opposite literals. Adding these edges between opposite literals, ensures that we cannot pick both (as it would not be an independent set). The reduction is completed by setting the independent set goal K = m.

Example:
Given a 3CNF formula: $f = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$
We construct the graph shown in figure 1.

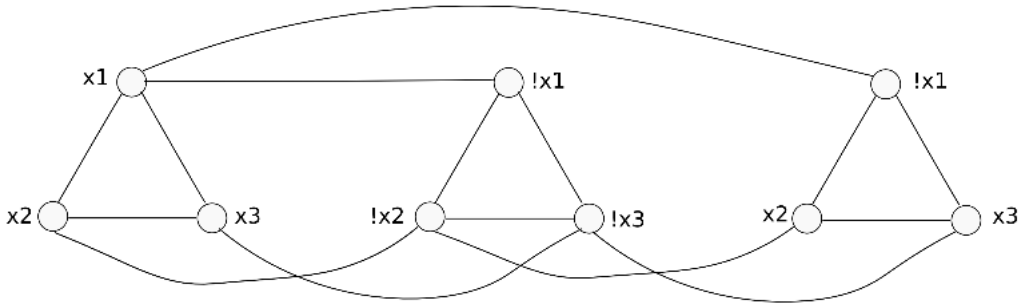

Figure 1: Graph r(f) corresponding to the formula f

Analysis
For each clause, we just create a triangle. Then we add at most $3 * (m - 1)$ edges between the triangles - so the reduction is clearly polynomial.

$\Longrightarrow$:
Given a satisfying assignment for the CNF formula f from above, we must show that the graph, r(f), has an indenpendent set of size K=m. The assignment: $x_1 = 0, x_2 = 1, x_3 = 0$ satisfies the formula, and for our indenpendent set of size K, we pick a node in each triangle which make the corresponding clause true. For the truth assignment we chose above, this means x_2 from triangle 1, $\neg x_1$ in triangle 2, $\neg x_1$ in triangle 3. (Note that this is not the only valid set since some clauses are satisfied by more than one literals).

$\Longleftarrow$:
Given an independent set of size K of r(f), we need to show that there is a satisfying assignment for f. Since it is of size K, it must have one vertex from each triangle, and it cannot contain a variable and its negation. Now, for each vertex in the independent set, we look at the label and assign a truth value to the corresponding variable, so the literal of the label become true. If a variable is not restricted to a truth value by any vertex label, we just assign it some value, since all clauses are already satisfied by the assignments to the other variables.

Read more in papadimitriou page 188-190.

2.2 INDEPENDENT SET $\leq$ CLIQUE

The independent set problem asks for a maximum set of vertices of size K. So does the CLIQUE problem. While the independent set problem asks for a set S of vertices where there is no edge between any pair $u, v \in S$, the clique problem ask for the "opposite": a set S where any pair $u, v \in S$ DOES have an edge between them. So the reduction is trivial - we obtain a clique of size K when taking the complementary graph of an independent set of size K.

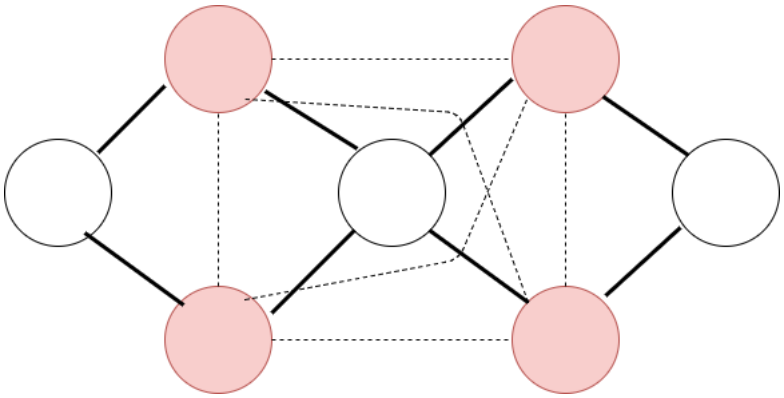

Figure 2: The red nodes are the maximum independent set I - the dotted lines completes the complementary graph of I. A clique of same size.

Read more in papadimitriou page 190.

2.3 INDEPENDENT SET $\leq$ VERTEX COVER

While independent set is a maximization problem, vertex cover is a minimization problem. For a graph G = (V,E) and a max independent set $I \subseteq V$, the minimum vertex cover set is just $V - I$.

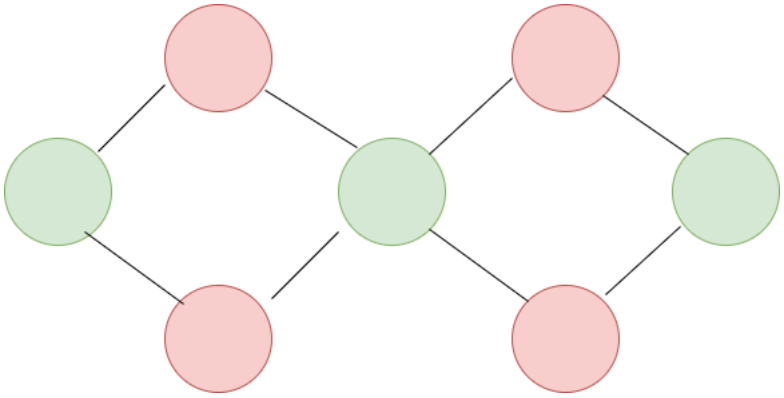

Figure 3: The red node forms the maximum independent set. The green form the minimal vertex cover.

Read more in papadimitriou page 190.

2.4 NAESAT $\leq$ MAX CUT

Theorem 9.5: MAX CUT is NP-Complete

We prove this by reducing from NAESAT, which we know is NP-complete.

We construct a graph G = (V,E), and a goal K such that there is a way to separate the nodes og G in to two sets S and V-S with K or more edges going from one set to the other, if and only if there is a truth assignment for our NAESAT formula f which satisfies f. We allow more than one edge between two vertices, and each such edge contributes one to the cut.

Our input formula has m clauses and n variables, and we constuct a graph with 2n nodes - one for each variable and it's negation. For each clause, we add edges to our graph to form triangles. If two literals of a clause are the same, for example $(x \vee y \vee y)$, (which is equivalent to $(x \vee y)$), we just add two edges between the nodes representing the two destinct literals. The point of the triangles and the two edges between same literals, is that for both constructs, the cut is always two. Now we add an edge between variables and their negation, and set our target K = n + 2m. **The n is for cutting the horizontal edges, and 2m is because we need to cut each triangle (each clause) - if we don't, then it is possible for all literals in a single clause to be true (and then it's not naesat))**. This completes the reduction.

Example:
Given a NAESAT formula: $f = (x_1 \vee x_2) \wedge (x_1 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$
We can construct the graph shown in figure 4. The green edges come from the first clause, red the second clause, and blue the third clause.

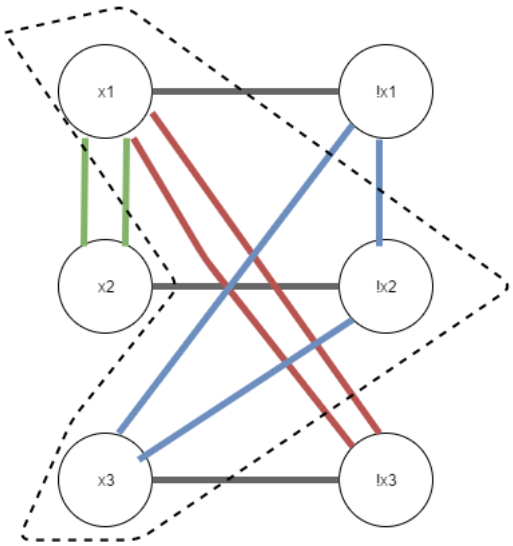

Figure 4: Graph r(f) corresponding to the formula f

Analysis:
For each variable we add two nodes and one edge between them. For each clause we add at most 3 edges. So the reduction is polynomial.

$\Longrightarrow$:
Given a satisfying naesat assignment for f, we must show that r(f) has a cut of size at least K. $x_1 = 1, x_2 = 0, x_3 = 1$ is a satisfying assignment for f. We have 3 variables and 3 clauses, so the target K is $2 * 3 + 3 = 9$. The cut of 9 is obtained by cutting the vertices within the dotted line in the example above.

$\Longleftarrow$:
Given a cut of size at least K of r(f), we must show that there is a satisfying naesat assignment for x. Here we just take the truth values of the nodes in our cut set.

Read more in papadimitriou page 191-192.

2.5 3SAT $\leq$ HAMILTONIAN PATH

2.6 HAMILTONIAN PATH $\leq$ TSP