Foundations of Computer Science
Logic, models, and computations

Chapter: Some NP**-complete problems**

Course CSC_41012_EP

of l'Ecole Polytechnique

Olivier Bournez

bournez@lix.polytechnique.fr

Version of August 20, 2024

Some NP-complete problems

Now that we have established the NP-completeness of a few problems, we are going to prove that a very huge number of problems are NP-complete.

The book [Garey & Johnson, 1979] listed more than 300 NP-complete problems in 1972. We do not have the ambition of presenting so many problems, but we will list some famous NP-complete problems: Our main purpose is actually to provide some examples of proofs of NP-completeness.

1 Some NP-complete problems

1.1 Around SAT

Recall the following theorem proved in previous chapter.

Definition 1 (3-SAT)

Input: *A set of variables* $\{x_1, \cdots, x_n\}$ *and a formula* $F = C_1 \wedge C_2 \cdots \wedge C_\ell$ *with* $C_i = y_{i,1} \vee y_{i,2} \vee y_{i,3}$, *where for every* i, j, $y_{i,j}$ *is either* x_k, *or* $\neg x_k$ *for one of the* x_k.

Answer: *Decide whether* F *is satisfiable: that is, decide if there exist* $x_1, \cdots, x_n \in \{0,1\}^n$ *such that* F *evaluates to true with this value of its variables* $x_1, \cdots, x_n$.

Theorem 1 *The problem* 3-SAT *is* NP*-complete.*

Remark 1 *The problem* $2-$SAT, *where we would consider clauses with two literals, is in* P.

Definition 2 (NAESAT)

Input: *A set of variables* $\{x_1, \cdots, x_n\}$ *and a set of clauses* $y_{i,1} \vee \cdots \vee y_{i,k_i}$, *where for all* i, j, $y_{i,j}$ *is either* x_k, *or* $\neg x_k$ *for some of the* x_k.

Answer: *Decide if there is some assignment of the variables* $x_i \in \{0,1\}$ *in such a way that every clause contains at least one true literal and at least one false*

literal (that is, for every i, there is a j and a k with $y_{i,j} = 1$ and $y_{i,k} = 0$).

Theorem 2 *The problem* NAESAT *is* NP*-complete.*

Proof: The problem is in NP since, given some assignment of the variables, it is easy to check in polynomial time that the property holds for these values of the variables.

We will reduce SAT to NAESAT. Let F a formula of SAT on the variables $\{x_1, \cdots, x_n\}$. We add a unique distinct variable z and we construct the clauses for NAESAT by replacing each clause $C_i = y_{i,1} \vee \cdots \vee y_{i,k}$ of F by $C'_i = y_{i,1} \vee \cdots \vee y_{i,k} \vee z$.

This transformation can be realized in polynomial time.

If the given instance of SAT is satisfiable, the same assignment of variables fixing for z the value 0 is a valid assignment for NAESAT.

Conversely, suppose that the constructed instance of NAESAT is satisfiable. If the truth value of z in the corresponding assignment is 0, then the values of the variables x_i in the assignment give some valid assignment for the original formula F (for the instance of SAT). If, on the contrary z values 1, then change all the values of all the variables in the assignment. The assignment remains valid for NAESAT since at least one literal by clause in the initial assignment values 0, and hence values now 1, whereas z values 0. We have built an assignment in which z values 0, and by previous case the initial instance of SAT is satisfiable.

We have indeed proved the equivalence between satisfiability of F and the corresponding instance of NAESAT. Hence NAESAT is NP-complete. □

By using the same reduction for instances 3SAT, we get that NAE4SAT is NP-complete: NAE4SAT is NAESAT restricted to formulas with 4 literals in each clause. We can actually prove, that this holds for the version with three literals.

Corollary 1 NAE3SAT *is* NP*-complete.*

Proof: We will reduce NAE4SAT to NAE3SAT. Let $C = x \vee y \vee z \vee t$ be a clause with 4 literals. We introduce a new variable u_C, and we construct the two clauses $C_1 = x \vee y \vee \neg u_C$ and $C_2 = z \vee t \vee u_C$. Doing so for all the clauses, we clearly construct an instance F' of NAE3SAT in polynomial time.

Suppose that F' is some positive instance of NAE3SAT, and consider the assignment of the corresponding truth value. If $u_C = 0$, then x or y is 0, and z or t is 1, so $x \vee y \vee z \vee t$ has at least a literal 1 and at least one literal 0; Similarly, if $u_C = 1$; So F is a positive instance of NAE4SAT.

Conversely, if F is a positive instance of NAE4SAT, consider the corresponding truth assignment. In $x \vee y \vee z \vee t$, if x and y are both set to 1, set u_C to 1; otherwise if x and y are both set to 0, set u_C to 0; otherwise, set u_C to the suitable truth value for the clause $u_C \vee z \vee t$. That produces an assignment that proves that F' is a positive instance of NAE3SAT.

Once again, NAE3SAT is in NP from definition, as a value for the variable is clearly a certificate that can be checked in polynomial time. □

1.2 Around INDEPENDANT SET

Definition 3 (INDEPENDANT SET)

Input: *An (undirected) graph $G = (V, E)$ and some integer k.*

Answer: *Decide whether there exists $V' \subset V$, with $|V'| = k$, such that $u, v \in V' \Rightarrow (u, v) \notin E$.*

Theorem 3 *The problem* INDEPENDANT SET *is* NP*-complete.*

Remark 2 *Such an independent set V' is sometimes also called a* stable set *(or* stable*).*

Proof: INDEPENDANT SET is indeed in NP, since giving V' provides a certificate that can be easily checked in polynomial time.

We reduce the problem 3-SAT to INDEPENDANT SET, that is to say, given some formula F of type 3-SAT, we construct in polynomial time a graph G in such a way that the existence of a stable set in G is equivalent to the existence of a truth assignment that satisfies F.

Let $F = \bigwedge_{1 \leq j \leq k} (x_{1j} \vee x_{2j} \vee x_{3j})$. We construct a graph G with $3k$ vertices, one for each occurrence of a literal in a clause.

- For every variable x_i of 3-SAT, G has an edge between every vertex associated to a literal x_i and every vertex associated to a literal $\neg x_i$ (and so an independent set of G corresponds to a truth assignment of a subset of variables);
- For every clause C, we associate a triangle: for example for a clause of F of the form $C = (x_1 \vee \neg x_2 \vee x_3)$, then G has the edges $(x_1, \neg x_2)$, $(\neg x_2, x_3)$, (x_3, x_1) (by doing, an independent set of G must contain at most one of the vertices associated to clause C).

Let k be the number of clauses in F. One proves that F is satisfiable if and only if G has an independent set of size k.

Indeed, if F is satisfiable, consider an assignment of the variables that satisfies F. For every clause C of F, select y_C a literal of C that is set to true by the assignment: that defines k vertices defining an independent set of G.

Conversely, if G has an independent set of size k, then it has necessarily a vertex in every triangle. This vertex corresponds to a literal that makes the associated clause true, and this forms an assignment of the variables that is consistent by construction of the edges.

The reduction is clearly polynomial. □

Two classical problems are related to INDEPENDANT SET.

Definition 4 (CLIQUE)

Input: *An (undirected) graph $G = (V, E)$ and some integer k.*

Answer: *Decide if there exists $V' \subset V$, with $|V'| = k$, such that $u, v \in V' \Rightarrow (u, v) \in E$.*

Theorem 4 *The problem* CLIQUE *is* NP*-complete.*

Proof: The reduction from INDEPENDANT SET consists in going to the complementary on the edges. Indeed, it is sufficient to observe that a graph $G = (V, E)$ has an independent set of size k if and only if complementary graph $\overline{G} = (V, \overline{E})$ (where $\overline{E} = \{(u, v) | (u, v) \notin E\}$) has a clique of size k. □

Definition 5 (VERTEX COVER)

Input: *An (undirected) graph $G = (V, E)$ and some integer k.*

Answer: *Decide if there exists $V' \subset V$, with $|V'| = k$, such that every edge of G has at least one of its extremity in V'.*

Theorem 5 *The problem* VERTEX COVER *is* NP*-complete.*

Proof: The reduction from INDEPENDANT SET consists in considering the complementary on the vertices. □

Definition 6 (MAXIMAL CUT)

Input: *An (undirected) graph $G = (V, E)$ and some integer k.*

Answer: *Decide if there exists a partition $V = V_1 \cup V_2$ such that the number of edges between V_1 and V_2 is at least k.*

Theorem 6 *The problem* MAXIMAL CUT *is* NP*-complete.*

Proof: We reduce NAE3SAT to MAXIMAL CUT. Suppose an instance of NAE3SAT is given, in which we can suppose without loss of generality that every clause does not contain simultaneously a variable and its complementary. Replacing $u \vee v$ by $((u \vee v \vee w) \wedge (u \vee v \vee \neg w))$ if needed, we can suppose that every clause contains exactly 3 literals. Furthermore, if we have $(u \vee v \vee w)$ and $(u \vee v \vee z)$, we can, by introducing two variables t_1 and t_2 and by proceeding as we did to reduce NAE4SAT to NAE3SAT, rewrite these two clauses as $(u \vee t_1 \vee t_2) \wedge (v \vee w \vee \neg t_1) \wedge (v \vee z \vee \neg t_2)$. In other words, we can suppose that two given clauses have at most one variable in common.

We denote by $x_1, \cdots, x_n$ the variables of the formula F.

We will construct a graph $G = (V, E)$ in the following way: G has $2n$ vertices and every variable u of F is corresponding to two vertices u and $\neg u$. G has an edge between every couple of vertices (u, v) such that u and v appear in the same clause, and an edge between the vertices u and $\neg u$ for every variable u.

The reductions in the first paragraph of the proof permit to state that to every clause corresponds a triangle and that two of these triangles have distinct edges.

If we denote by n the number of variables and by m the number of clauses, the graph G has $2n$ vertices and $3m + n$ edges. It is easy to see that the number of edges in a cut corresponding to a valid NAE3SAT assignment is $2m + n$: The edge between u and $\neg u$ for every variable u, and two edges of the triangle uvw for every clause $u \vee v \vee w$.

Conversely, every cut of G has at most $2m + n$ edges, since a cut can not include more than the number of edges for a given triangle associated to a clause. Consequently, a cut of value $2m + n$ provides immediately a valid NAE3SAT assignment.

In other words, solving an instance of NAE3SAT is equivalent to solve MAXIMAL CUT on $(G, 2m + n)$.

The reduction is polynomial. Now MAXIMAL CUT is in NP since giving V_1 is a valid certificate that can be checked in polynomial time. □

1.3 Around HAMILTONIAN CIRCUIT

The problem HAMILTONIAN CIRCUIT is often at the source of the proof of NP-completeness of properties related to paths in graphs.

Definition 7 (HAMILTONIAN CIRCUIT)

Input: *An (undirected) graph $G = (V, E)$.*

Answer: *Decide if there exists a* Hamiltonian circuit *that is to say a path of G that goes exactly once through every vertex and that comes back to its starting point.*

Theorem 7 *The problem* HAMILTONIAN CIRCUIT *is* NP*-complete.*

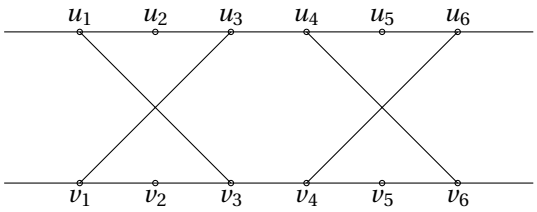
Proof: We will prove that fact by reducing VERTEX COVER to this problem.

The idea consists, starting from an instance of VERTEX COVER, in constructing a graph in which every initial edge will be replaced by a "pattern" that admits exactly two Hamiltonian paths, that is to say paths that visit exactly once every vertex. One of this path will correspond to the case where the corresponding vertex belongs to the cover, the other where it does not belong.

There will remain to tell how to glue all these patterns so that a global Hamiltonian circuit corresponds exactly to a union of Hamiltonian paths through every pattern, and to ensure that one obtains the good result to problem VERTEX COVER by solving HAMILTONIAN CIRCUIT in this graph.

We now go to the details: Denote $(G = (V, E), k)$ the considered instance of VERTEX COVER.

The pattern that we will use is the following:



To obtain a traversal of this pattern that goes exactly once through every vertex, only two solutions are possible: either a traversal in two steps, first $u_1u_2u_3u_4u_5u_6$ then later $v_1v_2v_3v_4v_5v_6$ (in one direction or the other); or a traversal in only one step $u_1u_2u_3v_1v_2v_3v_4v_5v_6u_4u_5u_6$ (or the same but switching the role of the u's and of the v's).

To every edge (u,v) of the graph G, we associate a pattern of this type, by putting in correspondence the vertex u_i's at side u and the vertex v_i's at side v. We then link together all the sides of all the patterns corresponding to a same vertex. We construct consequently a chain associated to a given vertex, with still two free "outputs". We then link again these two "outputs" to k new vertices $s_1,\cdots,s_k$. We denote by H the graph constructed in that way in the remaining part of this proof.

Suppose now that a cover of the initial graph of size k, whose vertices are $\{g_1,\cdots,g_k\}$ is given. One can then construct a Hamiltonian circuit of H in the following way:

- starts from s_1;
- follow the chain g_1 in the following way. When one traverses an edge (g_1,h), if h is also in the cover, simply cross the side g_1; otherwise, cross the two sides simultaneously;
- once the chain g_1 is finished, come back to s_2 and restart by g_2 and so on.

It is clear that every vertex s_k is traversed exactly once.

Consider a vertex h of the pattern corresponding to an edge (u,v). One can always suppose that u is in the cover, say $u=g_1$. It follows that if h is on the same side as u, h will be reached at least once. We see, according to second item, that it will not be reached later on. If h is on the same side as v, and if v is not in the cover, h is visited along the traversal of u. If $v=g_i$, h is crossed by the traversal of the chain corresponding to g_i and at this moment only. We hence have indeed a Hamiltonian circuit.

Conversely, suppose that we have a Hamiltonian circuit of H. The construction of our pattern implies that, coming from vertex s_i, one traverses completely a chain u and then one goes to a vertex s_j. One then traverses k chains; the k corresponding vertices form a cover. Indeed, if (u,v) is an edge, the corresponding pattern is traversed by the Hamiltonian path; But it can be only a traversal of a loop corresponding to one of the extremities of the edge.

Finally, the reduction is trivially polynomial. HAMILTONIAN CIRCUIT is hence NP-complete. □

As in previous section, we can then deduce the NP-completeness of many variants.

Definition 8 (TRAVELING SALESMAN)

Input: *A couple (n, M), where M is a matrix $n \times n$ of integers and some integer k.*

Answer: *Decide if there exists some permutation π of $[1, 2, \cdots, n]$ such that*

$$\sum_{1 \leq i \leq n} M_{\pi(i)\pi(i+1)} \leq k.$$

(here indices are taken modulo n so that it makes sense)

Corollary 2 *The problem* TRAVELING SALESMAN *is* NP*-complete.*

Remark 3 *This problems has its name, since it can be seen as establishing the planning of visits of a traveling salesman that must visit n town, whose distances are given by matrix M, in less than k kilometers.*

Proof:

We reduce HAMILTONIAN CIRCUIT to TRAVELING SALESMAN. To do so, given a graph $G = (V, E)$, consider $V = \{x_1, \cdots, x_n\}$. We consider then the matrix M $n \times n$ of integers such that

$$M_{i,j} = \begin{cases} 1 & \text{if } (i, j) \in E; \\ 2 & \text{otherwise.} \end{cases}$$

We then claim that HAMILTONIAN CIRCUIT(V, E) is true if and only if TRAVELING SALESMAN(n, M, n). Indeed:

If there exists a Hamiltonian circuit in G, one can then indeed construct the permutation π as description the order of traversal of the vertices of graph G: By construction, the sum of the distances of the edges on this circuit will value n.

Conversely, given some permutation with this property, the fact that the n terms of the sum value at least 1 implies that they are all equal to 1, and so that the edges $(\pi(i), \pi(i+1))$ exist in the graph G: we hence have a Hamiltonian circuit. □

Definition 9 (LONGEST CIRCUIT)

Input: *An (undirected) graph $G = (V, E)$, with a distance on each edge, and some integer r.*

Answer: *Decide if there exists a circuit G that does not visit twice the same vertex whose length is $\geq r$.*

Corollary 3 *The problem* LONGEST CIRCUIT *is* NP*-complete.*

Proof: We construct a reduction from HAMILTONIAN CIRCUIT: to do so, to every graph for HAMILTONIAN CIRCUIT, we associate the length 1 to every edge. Finding a Hamiltonian cycle is then trivially identical to finding a circuit of length $\geq n$ in the graph. □

1.4 Around 3-COLORABILITY

We have defined what is called a *(well) colouring* of a graph in the previous chapter. The smallest integer k such that a graph can be (well) colored is called the *chromatic number* of the graph.

It is known that planar graphs are always colorable with 4 colors.

We recall the following result established in previous chapter.

Definition 10 (3-COLORABILITY)

Input: *An (undirected) graph $G = (V, E)$.*

Answer: *Decide if there exists a colouring of the graph that uses at most 3 colors.*

Theorem 8 *The problem* 3-COLORABILITY *is* NP*-complete.*

1.5 Around SUBSET SUM

Definition 11 (SUBSET SUM)

Input: *A finite sequence of integers $x_1, x_2, \ldots, x_n$ and some integer t.*

Answer: *Decide if there exists $E \subset \{1, 2, \ldots, n\}$ such that $\sum_{i \in E} x_i = t$.*

Theorem 9 *The problem* SUBSET SUM *is* NP*-complete.*

Proof: SUBSET SUM is in NP since giving E is a certificate that can be checked in polynomial time.

We use a reduction from VERTEX COVER.

Suppose that a graph $G = (V, E)$ is given in which one wants to determine if there exist a vertex cover of size k. Number the vertices and the edges. Let $B = (b_{ij})$ the incident matrix vertex-edges, that is to say $b_{ij} = 1$ if edge i is incident to vertex j, $b_{ij} = 0$ otherwise.

Consider $b \geq 4$. We will construct a sequence F of integers: For every edge i, we add integer b^i to F. For every vertex j we add integer a_j to F, where $a_j = b^m + \sum_{i=0}^{m-1} b_{i,j} b^i$.

We consider then

$$t = kb^m + \sum_{i=0}^{m-1} 2b^i. \tag{1}$$

Consider a cover S of G of cardinality k. Construct the subsequence made of the a_j's such that $j \in S$, and of the b^i's such that exactly one of the two extremities of edge i is in S. Then the sum of the elements of this subsequence is t: Indeed, we sum k times the term b^m and each edge i with two extremities in S contributes to b^i for each of its two extremities by the part of the sum related to the a_j's, and each

edge i with one extremity in S contributes to b^i once by the part of the sum related to the a_j's, and once by the integer b^i.

Conversely, suppose that we have a subsequence of sum t. We split the subsequence into X_1 made of the elements $x_i \geq m$, and into X_2 made of the elements $x_i < b^m$. To every element of X_1, is associated some vertex j: We take S to be the set of such vertices associated to elements of X_1. Since $kb^m \leq t < (k+1)b^m$, necessarily the size of S is k.

It remains to show that S is a cover. As in every sum of elements of F, there are at most three terms b^i for $i < m$, and as $b \geq 4$, no carry can be produced in the addition (except possibly for the coefficient of b^m that can exceed $b-1$). Consequently, the number of occurrences of term b^i can be read on equation (1), and it must necessarily be 2 for $i < m$.

That means that each edge i must necessarily have at least one of its extremity in S, since otherwise the number of occurrences of term b^i would be 0 or 1.

We have indeed reduced VERTEX COVER to SUBSET SUM.

Indeed, it is easy to see that the reduction is done in polynomial time, and hence we have proved the theorem. □

We can deduce:

Definition 12 (KNAPSACK)

Input: *A set of weights $a_1, \cdots, a_n$, a set of values $v_1, \cdots, v_n$, a weight limit A, and some integer V.*

Answer: *Decide if there exists $E' \subset \{1, 2, \cdots, n\}$ such that $\sum_{i \in E'} a_i \leq A$ and $\sum_{i \in E'} v_i \geq V$.*

Corollary 4 *The problem* KNAPSACK *is* NP*-complete.*

Proof: From SUBSET SUM: Given $E = \{e_1, \cdots, e_n\}$ and t an instance of SUBSET SUM, one considers $v_i = a_i = e_i$, and $V = A = t$. □

Definition 13 (PARTITION)

Input: *A finite sequence of integers $x_1 x_2 \ldots x_n$.*

Answer: *Decide if there exists $E \subset \{1, 2, \ldots, n\}$ such that $\sum_{i \in E} x_i = \sum_{i \notin E} x_i$.*

Theorem 10 *The problem* PARTITION *is* NP*-complete.*

Proof: We will reduce SUBSET SUM to PARTITION. Let $(x_1 x_2 \ldots x_n, t)$ be an instance of SUBSET SUM. We set $S = \sum_{1 \leq i \leq n} x_i$. Changing t in $S - t$ if needed (which is equivalent to change the obtained set in its complement), we can suppose that $2t \leq S$.

A first natural idea would consist in adding the element $u = S - 2t$ to the sequence. The result of a partition would then be two subsequences of sum $S - t$. One

of the two would contain the element $S-2t$, and hence by suppressing the latter, we would find a subsequence of sum t. Unfortunately, this reasoning fails if $S-2t$ is already in the sequence.

Instead of doing so, we take the number $X = 2S$ and $X' = S+2t$, and we apply PARTITION to the sequence $x_1, x_2, \ldots, x_n, X, X'$. There exists a partition of this sequence if and only if there exists a subsequence of $x_1, x_2, \ldots, x_n$ of sum t. Indeed, if there exists a partition of $x_1, x_2, \ldots, x_n, X, X'$, there exists two subsets complementary of sum $2S+t$. Each of this two subsets must contain either X, or X', since otherwise its sum would exceed $2S+t$; So one of the two subsets contains X and not X'. By suppressing X in it, we must then obtain a subsequence F of $x_1, x_2, \ldots, x_n$ of total sum t. Conversely, given such a subsequence F of total sum t, then the subsequence made of F and X, and its complement, constitutes a partition of $x_1, x_2, \ldots x_n, X, X'$. We have indeed reduced SUBSET SUM to PARTITION.

It remains to justify that the reduction is indeed polynomial. The main part of the reduction is the computation of A and A', and this is indeed polynomial in the size of the inputs (the addition of k numbers of n bits can be done in time $\mathcal{O}\left(k \log n\right)$). □

2 Exercises

2.1 Polynomial variants

Exercise 1 *A graph $G = (V, E)$ is said* Eulerian *if there exists a cycle that goes through every edge of G exactly once.*

Prove that a connected graph is Eulerian if and only if all its vertices are of degree two.

Propose a polynomial algorithm to determine if a graph is Eulerian.

Exercise 2 *We consider* 2SAT.

1. *Propose an evaluation t that satisfies $\phi_1(u_1,u_2,u_3) = (u_2 \vee \neg u_3) \wedge (\neg u_2 \vee u_3) \wedge (u_2 \vee \neg u_1)$.*
2. *What happens for $\phi_2(u_1,u_2) = (u_2 \vee u_3) \wedge (\neg u_2 \vee u_3) \wedge (\neg u_3 \vee u_1) \wedge (\neg u_3 \vee \neg u_1)$?*
3. *Prove that $u \vee v = (\neg u \Rightarrow v) \wedge (\neg v \Rightarrow u)$.*

   Starting from an instance of 2SAT *we construct a directed graph $G_\phi = (V,E)$ with*

   - *a vertex for every literal;*
   - *and an arc for every implication (by transforming every clause into two implications).*
4. *Draw the graphs G_{ϕ_1} and G_{ϕ_2}.*
5. *Prove that there exists a variable u such that G_ϕ contains a cycle between u towards $\neg u$ in G, if and only if ϕ is not satisfiable.*
6. *Prove that* 2-SAT *can be solved in polynomial time.*

Exercise 3 *(solution on page 235) [Knights of the round table] Given n knights, and knowing all the pairs of fierce enemies among them, is it possible to position them in polynomial time around some round table in such a way that no pair of fierce enemy are sitting near each other.*

2.2 NP-completeness

Exercise 4 *(solution on page 235) A* Hamiltonian path *is a path that goes through every vertex of the graph exactly once.*
Prove that the following problem is NP*-complete:*

Input: *A (undirected) graph G of n vertices, two distinct vertices u and v of G.*

Answer: *Decide if G contains a Hamiltonian path whose extremities are vertices u and v.*

Exercise 5 *(solution on page 236) Prove that the following problem is* NP*-complete.*

Input: *An (undirected) graph G with n vertices, two distinct vertices u and v of G.*

Answer: *Decide if G contains a path of length $n/2$ between u and v.*

Exercise 6 *(solution on page 237) Prove that the following problem is* NP*-complete.*

Input: *A graph $G = (V, E)$, and some integer k.*

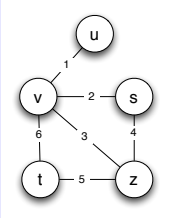
Answer: *Decide if there exists a tree covering all the vertices of G with at least k leaves.*

Exercise 7 *(solution on page 237) Let $G = (V, E)$ be a graph.*

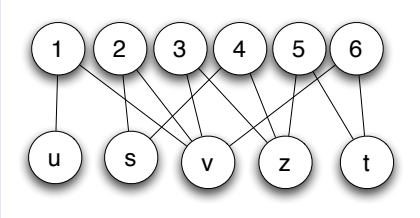
- *A* vertex cover *S of graph G is a subset of vertices such that all edges of G are incident to at least a vertex of S.*
- *A* dominating set *C of graph G is a subset of vertices such that every vertex is either in C or neighbour of a vertex of C.*

Let $G = (V, E)$ be a connected graph. We are going to construct a graph $G' = (V', E')$ from G such that

- $V' = V \cup E$;
- $E' = E \cup \{(v, a) | v \in V, a \in E, v$ *is a extremity of a in* $G\}$



Graph G



Graph G' (without the edges of G)

1. *Prove that if S is a vertex cover of G, then S is a dominating set of G'.*
2. *Prove that if S' is a dominating set of G', then there exists some vertex cover $S \subseteq V$ of graph G of cardinal less or equal to S'.*
3. *Express the problem of minimization of the dominating set as a decision problem.*
4. *Prove that this problem is in* NP.
5. *Prove that the problem is* NP*-complete.*

Exercise 8 *(solution on page 237) We will focus on the problem of k-centre: Given a set of towns whose distances are given, select k towns in order to put some warehouses in order to minimize the maximal distance from a town to the closest warehouse. Such a set of k town is called a k-center.*

The associated decision problem is the following:

Input: *A complete graph $K = (V, E)$ having a weight function w on the edges, and some strictly positive integers k and b.*

Answer: *Decide if exists a set S of vertices such that $|S| = k$ and such that every vertex v of V satisfies the following condition*

$$min\{w(v,u) : u \in S\} \leq b.$$

1. *Prove that k-*CENTRE *is in* NP.
2. *Prove that k-*CENTRE *is* NP*-complete, knowing that* DOMINANT *is* NP*-complete.*

3 Bibliographic Notes

Suggested readings The book [Garey & Johnson, 1979] provides a list of more than 300 NP-complete problems, and is rather pleasant to read. One can find updates of lists of complete problems on the web.

Bibliography This chapter is reproduced, and adapted from course book [Cori et al., 2010] from course INF550 of École Polytechnique.

Index

Bibliography

[Cori et al., 2010] Cori, R., Hanrot, G., Kenyon, C., & Steyaert, J.-M. (2010). Conception et analyse d'algorithmes. Cours de l'Ecole Polytechnique, Cours de l'Ecole Polytechnique.

[Garey & Johnson, 1979] Garey, M. R. & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.